Human Computation

Luis von Ahn

CMU-CS-05-193

December 7, 2005

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Manuel Blum, Chair
Takeo Kanade
Michael Reiter
Josh Benaloh, Microsoft Research
Jitendra Malik, University of California, Berkeley

This work was partially supported by the National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982 (The Aladdin Center), by a Microsoft Research Fellowship, and by generous gifts from Google, Inc. The views and conclusions contained in this document are those of the author and should not be interpreted as representing official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: CAPTCHA, the ESP Game, Peekaboom, Verbosity, Phetch, human computation, automated Turing tests, games with a purpose.

Abstract

Tasks like image recognition are trivial for humans, but continue to challenge even the most sophisticated computer programs. This thesis introduces a paradigm for utilizing human processing power to solve problems that computers cannot yet solve. Traditional approaches to solving such problems focus on improving software. I advocate a novel approach: constructively channel human brainpower using computer games. For example, the ESP Game, introduced in this thesis, is an enjoyable online game — many people play over 40 hours a week — and when people play, they help label images on the Web with descriptive keywords. These keywords can be used to significantly improve the accuracy of image search. People play the game not because they want to help, but because they enjoy it.

I introduce three other examples of "games with a purpose": Peekaboom, which helps determine the location of objects in images, Phetch, which collects paragraph descriptions of arbitrary images to help accessibility of the Web, and Verbosity, which collects "common-sense" knowledge.

In addition, I introduce CAPTCHAs, automated tests that humans can pass but computer programs cannot. CAPTCHAs take advantage of human processing power in order to differentiate humans from computers, an ability that has important applications in practice.

The results of this thesis are currently in use by hundreds of Web sites and companies around the world, and some of the games presented here have been played by over 100,000 people. Practical applications of this work include improvements in problems such as: image search, adult-content filtering, spam, common-sense reasoning, computer vision, accessibility, and security in general.

Acknowledgements

First, I would like to thank my advisor Manuel Blum for his unconditional support and advice. This thesis would not have been possible without him and without the freedom and inspiration he has given me over the last 5 years. I could not have asked for a better advisor or for better advice.

I would also like to thank Lenore Blum for her tremendous support and advice. My fiancée, Laura Dabbish, has provided both emotional and intellectual support without which this thesis would not have been possible. I would like to thank my officemates and friends Aditya Akella and Shuchi Chawla, who managed to survive being next to me for 5 years. My undergraduate students Andrew Bortz, Shiry Ginosar, Mihir Kedia, Roy Liu, and Yinmeng Zhang have provided an immense amount of help; this thesis would also not have been possible without them. Susan Hrishenko has helped in many of the logistical and strategic issues revolving around the material in this thesis, and I am eternally grateful to her. My coauthors Nick Hopper and John Langford have provided many great ideas as well as constant friendship.

I would like to thank my thesis committee members, Josh Benaloh, Takeo Kanade, Jitendra Malik and Mike Reiter for their advice and comments. While at Microsoft, Josh Benaloh provided immense support and a number of great ideas that have inspired some of the work in this thesis.

I would also like to thank Avrim Blum, Alexei Efros, Alan Eustace, Brighten Godfrey, Scott Hudson, Steven Katz, Sara Kiesler, Udi Manber, Moni Naor, Lenore Ramm, Roni Rosenfeld, Chuck Rosenberg, Ryan Staake, David Stork, Latanya Sweeney, Shingo Uchihashi, Manuela Veloso and the anonymous CHI, Eurocrypt and Communications of the ACM reviewers for their insightful comments.

Contents

Chapter 1: Introduction

Construction of the Empire State Building: 7 million human-hours. The Panama Canal: 20 million human-hours. Estimated number of human-hours spent playing solitaire around the world in one year: *billions.* A problem with today's computer society? No, an opportunity.

What if this time and energy could be channeled into useful work? This thesis presents a general paradigm for doing exactly that: utilizing human processing power.

We focus on harnessing human time and energy for addressing problems that computers cannot yet tackle on their own. Although computers have advanced significantly in many respects over the last 50 years, they still do not possess the basic conceptual intelligence or perceptual capabilities that most humans take for granted. By leveraging human skills and abilities in a novel way, we hope to solve large-scale computational problems that computers cannot yet solve and begin to teach computers many of these human talents.

In this paradigm, we treat human brains as processors in a distributed system, each performing a small part of a massive computation. Unlike computer processors, however, humans require an incentive in order to become part of a collective computation. We propose online games as a means to encourage participation in the process.

We argue that games constitute a general mechanism for using brain power to solve open computational problems. Each problem requires the careful design of a game developed to be enjoyable and, at the same time, guarantee that game-play correctly solves instances of the problem. We argue that designing such games is much like designing computer algorithms: the game needs to be proven correct and enjoyable; its efficiency can be analyzed; more efficient games can supersede less efficient ones, etc. Instead of using a silicon processor, these "algorithms" run on a processor consisting of *ordinary humans interacting with computers* over the Internet.

We refer to these games as "human algorithm games."

An example of a human algorithm game is the ESP Game (Chapter 3). ESP is a popular online game: many people play over 40 hours a week. By playing, they provide meaningful, accurate keyword labels for images on the Web (an image with a dog, for instance, gets the label "dog"). These labels can be used, among other things, to improve the accuracy of image search. In only a few months of deployment, the ESP Game collected over 10 million image labels, and if it were deployed at a popular gaming site like *MSN* or *Yahoo! Games*, all images on the Web could be labeled in a matter of weeks. Rather than using computer

vision techniques that don't work well enough, the ESP Game constructively channels people to do the work as a form of entertainment.

In this thesis we introduce, along with the ESP Game, multiple examples of human algorithm games:

- **Peekaboom (Chapter 4)**, a game that helps to locate objects in images, and has been played for thousands of human-hours. The data produced by Peekaboom (freely available for research use) can be used to train computer vision algorithms.
- **Phetch (Chapter 5)**, a game that annotates images with descriptive paragraphs. These paragraphs can help, among other things, to improve accessibility of the Web by providing descriptions of images to the visually impaired.
- **Verbosty (Chapter 6)**, a game that collects "common-sense" knowledge that can be used to train reasoning algorithms.

In every case, people play the games not because they are interested in solving an instance of a computational problem, but because they wish to be entertained. In Chapter 7 we present a general paradigm for building such "games with a purpose."

Another instantiation of the general approach for utilizing human processing power introduced in this thesis is CAPTCHA (Chapter 2), an automated test that humans can pass but that current computer programs cannot. You have probably seen them at the bottom of Web registration forms: colorful images with distorted text in them. CAPTCHAs take advantage of the power of human cycles in order to differentiate people from computer. The ability to do this has important applications in practical security. In Chapter 2 we present CAPTCHAs based on a variety of human abilities such as understanding images and reading distorted text. The results of this work are used by Yahoo! and many other websites to ensure that only humans obtain free email accounts and otherwise gate access to free services.

The contribution of this thesis, therefore, is a general way of thinking about computational problems that takes human processing power into account. Whereas previous approaches to many problems have focused on improving software and algorithmic techniques, we argue that humans provide a viable, under-tapped resource that can aid in the solution of several important problems in practice.

Chapter 2: CAPTCHA*

In this Chapter we briefly discuss CAPTCHAs. You've probably seen them—colorful images with distorted text in them at the bottom of registration forms. CAPTCHAs are in use by Yahoo!, Hotmail, Paypal and most other popular web sites to prevent automated registrations, and they work because no computer program can currently read distorted text as well as humans can. What you probably don't know is that a CAPTCHA is something more general than an image with distorted text: it is a test, any test, that can be automatically generated, that most humans can pass, but that current computer programs can't pass. Notice the paradox: a CAPTCHA is a program that can generate and grade tests that it itself cannot pass (much like some professors).

CAPTCHA stands for "Completely Automated Public Turing Test to Tell Computers and Humans Apart." The P for Public means that the code and the data used by a CAPTCHA should be publicly available. This is not an open source requirement, but a security guarantee: it should be hard to write a computer program that can pass the tests generated by a CAPTCHA even if it's known exactly how the CAPTCHA works (the only hidden information is a small amount of randomness utilized to generate the tests). The T for "Turing Test to Tell" is because CAPTCHAs are like Turing Tests [62]. In the original Turing Test, a human judge was allowed to ask a series of questions to two players, one of which was a computer and the other a human. Both players pretended to be the human, and the judge had to distinguish between them. CAPTCHAs are similar to the Turing Test in that they distinguish humans from computers, but they differ in that the judge is now a computer. A CAPTCHA is an *Automated* Turing Test. Notice that we do not use the term "Reverse Turing Test" (or even worse, "RTT") because it can be misleading—"Reverse Turing Test" has been used to refer to a form of the Turing Test in which both players pretend to be a computer.

Applications

Although the goal of the original Turing Test was to serve as measure of progress for Artificial Intelligence—a computer would be said to be intelligent if it passed the Turing Test—making the judge be a computer allows CAPTCHAs to be useful for other practical applications:

- **Online Polls.** In November 1999, *slashdot.com* released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by

* This Chapter is adapted from work that appeared in *Communications of the ACM* [3].

using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting "bots." MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote.

- **Free Email Services.** Several companies (Yahoo!, Microsoft, etc.) offer free email services, most of which previously suffered from a specific type of attack: "bots" that signed up for thousands of email accounts every minute. This situation has been improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a CAPTCHA of our design to prevent bots from registering for accounts.
- **Search Engine Bots.** Some web sites don't want to be indexed by search engines. There is an *html* tag to prevent search engine bots from reading web pages, but the tag doesn't guarantee that bots won't read the pages; it only serves to say "no bots, please." Search engine bots, since they usually belong to large companies, respect web pages that don't want to allow them in. However, in order to truly *guarantee* that bots won't enter a web site, CAPTCHAs are needed.
- **Worms and Spam.** CAPTCHAs also offer a plausible solution against email worms and spam: only accept an email if you know there is a human behind the other computer. A few companies, such as *www.spamarrest.com* are already marketing this idea.
- **Preventing Dictionary Attacks.** Pinkas and Sander [49] have suggested using CAPTCHAs to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

Examples of CAPTCHAs

CAPTCHAs also differ from the original Turing Test in that they can be based on a variety of sensory abilities. The original Turing Test was conversational—the judge was only allowed to ask questions over a text terminal. In the case of a CAPTCHA, the computer judge can ask any question that can go over a network.

- **Gimpy and OCR-based CAPTCHAs.** GIMPY [5] is one of the many CAPTCHAs based on the difficulty of reading distorted text. GIMPY works as follows: it picks seven words out of a dictionary and renders a distorted image containing the words (as shown in Figure 1). GIMPY then presents a test to its user, which consists of the distorted image and the directions: "type three words appearing in the image." Given the types of distortions that GIMPY uses, most humans can read three words from the distorted image, but current computer programs can't. The majority of

CAPTCHAs used on the Web today are similar to gimpy in that they rely on the difficulty of Optical Character Recognition (i.e., the difficulty of reading distortet text).

Figure 1: Can you read 3 words in this image?

- **Bongo.** Another example of a CAPTCHA is the program we call BONGO [5]. BONGO is named after M.M. Bongard, who published a book of pattern recognition problems ([11]) in the 70s. BONGO asks the user to solve a visual pattern recognition problem. It displays two series of blocks, the LEFT and the RIGHT. The blocks in the LEFT series differ from those in the RIGHT, and the user must find the characteristic that sets them apart. A possible LEFT and RIGHT series are shown in Figure 2. After seeing the two series of blocks, the user is presented with a single block and is asked to determine whether this block belongs to the RIGHT series or to the LEFT. The user passes the test if he or she correctly determines the side to which the block belongs. Try it yourself: to which side does the isolated block belong in Figure 3? (Answer: RIGHT)

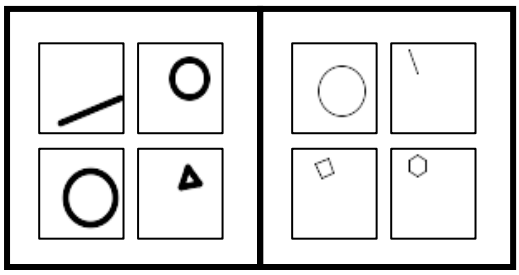

Figure 2: Everything on the left is drawn with thick lines, while everything on the right is drawn with thin lines.

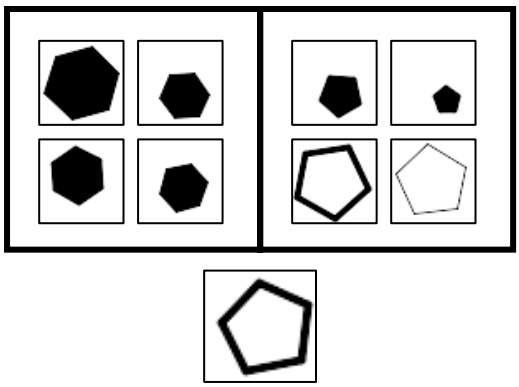

Figure 3: To which side does the block on the bottom belong?

- **Pix.** PIX [5] (see Figure 4) is a program that has a large database of labeled images. All of these images are pictures of concrete objects (a horse, a table, a house, a flower, etc). The program picks an object at random, finds 4 images of that object from its database, presents them to the user and then asks the question "what are these pictures of?" Current computer programs should not be able to answer this question, so PIX should be a CAPTCHA. However, PIX, as stated, is not a CAPTCHA: it is very easy to write a program that can answer the question "what are these pictures of?" Remember that all the code and data of a CAPTCHA should be publicly available; in particular, the image database that PIX uses should be public. Hence, writing a program that can answer the question "what are these pictures of?" is easy: search the database for the images presented and find their label. Fortunately, this can be fixed. One way for PIX to become a CAPTCHA is to randomly distort the images before presenting them to the user, so that computer programs can not easily search the database for the undistorted image.

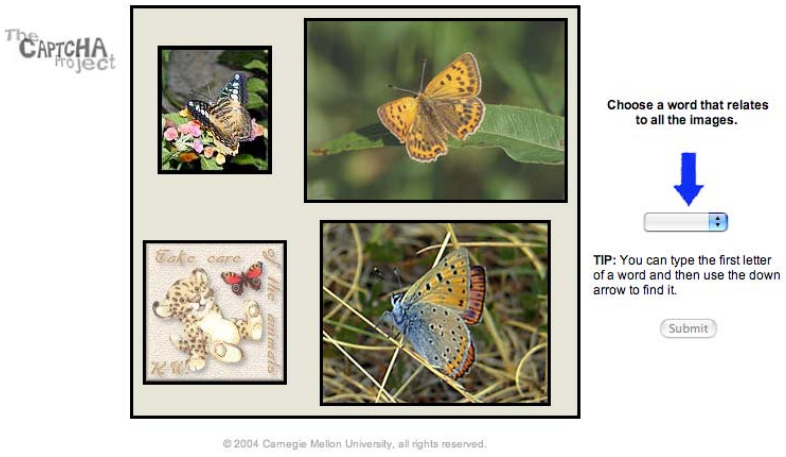

Figure 4. PIX. Choose a word that relates to all the images.

- **Sound-Based CAPTCHAs.** The final example that we mention is based on sound. The program picks a word or a sequence of numbers at random, renders the word or the numbers into a sound clip and distorts the sound clip; it then presents the distorted sound clip to its user and asks them to enter its contents. This CAPTCHA is based on the difference in ability between humans and computers in recognizing spoken language. Nancy Chan of the City University in Hong Kong was the first to implement a sound-based system of this type.

It is extremely important to have CAPTCHAs based on a variety of sensory abilities. All CAPTCHAs presented in this thesis, except for the sound-based CAPTCHA, rely on the user being able to see an image. However, there are many people who use the Web that are visually impaired. Thus, for accessibility purposes, CAPTCHAs based on sound are necessary.

Unfortunately, images and sound alone are not sufficient: there are people who use the Web that are both visually and hearing impaired. The construction of a CAPTCHA based on a text domain such as text understanding or generation is an important open problem for the project.

Lazy Cryptographers Doing AI

Modern cryptography has shown that open or intractable problems in number theory can be useful: an adversary cannot act maliciously unless they can solve an open problem (like factor a very large number). Similarly, CAPTCHAs show that open problems in AI can be useful: an adversary cannot vote thousands of times in online polls or obtain millions of free email accounts unless they can solve an open problem in Artificial Intelligence.

In the case of ordinary cryptography, it is assumed (for example) that the adversary cannot factor 1024-bit integers in any reasonable amount of time. In our case [3,16,34,44], we assume that the adversary cannot solve an Artificial Intelligence problem with higher accuracy than what's currently known to the AI community. This approach has the beneficial side effect of inducing security researchers, as well as otherwise malicious programmers, to advance the field of AI (much like computational number theory has been advanced since the advent of modern cryptography). This is how lazy cryptographers do AI.

A good example of this process is the recent progress in reading distorted text images driven by the CAPTCHA in use at Yahoo!. In response to the challenge provided by this test, Malik and Mori [42] have developed a program which can pass the test with probability roughly 0.8. Malik and Mori's algorithm represents significant progress in the general area of text recognition; it is encouraging to see such progress. A CAPTCHA implies a win-win situation: either the CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and a useful AI problem is solved!

The Flip Side of CAPTCHAs: Human Computation

CAPTCHAs take advantage of human processing power in order to differentiate humans from computers, and being able to do so has important applications in practice. But this Thesis is mostly concerned with the "flip side" of CAPTCHAs. There are many things that humans can easily do that computers cannot yet do. In the following chapters we show how to use human processing power in order to solve large-scale open problems in computer science.

Chapter 3: The ESP Game*

Images on the Web present a major technological challenge. There are millions of them, there are no guidelines about providing appropriate textual descriptions for them, and computer vision hasn't yet produced a program that can determine their contents in a widely useful way. However, accurate descriptions of images are required by several applications like image search engines and accessibility programs for the visually impaired. Current techniques to categorize images for these applications are insufficient in many ways, mostly because they assume that the contents of images on the Web are related to the text appearing in the page. This is insufficient because the text adjacent to the images is often scarce, and can be misleading or hard to process [14].

The only method currently available for obtaining precise image descriptions is manual labeling, which is tedious and thus extremely costly. But, what if people labeled images without realizing they were doing so? What if the experience was enjoyable? In this Chapter we introduce an interactive system in the form of a game with a unique property: the people who play the game label images for us. The labels are meaningful even if individual players attempt to disrupt them.

The labels generated by our game can be useful for a variety of applications. For accessibility purposes, visually impaired individuals surfing the Web need textual descriptions of images to be read out loud. For computer vision research, large databases of labeled images are needed as training sets for machine learning algorithms. For image search over the Web and inappropriate (e.g., pornographic) content filtering, proper labels could dramatically increase the accuracy of the current systems.

We believe our system makes a significant contribution, not only because of its valuable output, but also because of the way it addresses the image-labeling problem. Rather than making use of computer vision techniques, we take advantage of people's existing perceptual abilities and desire to be entertained.

Our goal is extremely ambitious: to label the majority of images on the World Wide Web. If our game is deployed at a popular gaming site like Yahoo! Games and if people play it as much as other online games, we estimate that most images on the Web can be properly labeled in a matter of weeks. As we show below, 5,000 people continuously playing the game could assign a label to all images indexed by Google [26] in 31 days.

* This Chapter is adapted from work that appeared in *CHI 2004* [2].

We stress that our method is not necessarily meant to compete against the other techniques available for handling images on the Web. The labels produced using our game can usually be combined with these techniques to provide a powerful solution.

The Open Mind Initiative

Our work is similar in spirit to the Open Mind Initiative (e.g., [54,55]), which is a worldwide effort to develop "intelligent" software. Open Mind collects information from regular Internet users (referred to as "netizens") and feeds it to machine learning algorithms. Volunteers participate by answering questions and teaching concepts to computer programs. Our method is similar to Open Mind in that we plan to use regular people on the Internet to label images for us. However, we put greater emphasis on our method being fun because of the scale of the problem that we want to solve. We don't expect volunteers to label all images on the Web for us: we expect all images to be labeled because people want to play our game.

General Description of the System

We call our system "the ESP game" for reasons that will become apparent as the description progresses. The game is played by two partners and is meant to be played online by a large number of pairs at once. Partners are randomly assigned from among all the people playing the game. Players are not told whom their partners are, nor are they allowed to communicate with their partners. The only thing partners have in common is an image they can both see.

From the player's perspective, the goal of the ESP game is to guess what their partner is typing on each image. Once both players have typed the same string, they move on to the next image (both player's don't have to type the string *at the same time,* but each must type the same string at some point while the image is on the screen). We call the process of typing the same string "agreeing on an image" (see Figure 5).

Player 1 guesses: purse
Player 1 guesses: bag
Player 1 guesses: brown

Success! Agreement on "purse"

Player 2 guesses: handbag

Player 2 guesses: purse
Success! Agreement on "purse"

Figure 5. Partners agreeing on an image. Neither of them can see the other's guesses.

Partners strive to agree on as many images as they can in 2.5 minutes. Every time two partners agree on an image, they get a certain number of points. If they agree on 15 images they get a large number of bonus points. The thermometer at the bottom of the screen (see Figure 6) indicates the number of images that the partners have agreed on. By providing players with points for each image and bonus points for completing a set of images, we reinforce their incremental success in the game and thus encourage them to continue playing. Players can also choose to pass or opt out on difficult images. If a player clicks the pass button, a message is generated on their partner's screen; a pair cannot pass on an image until both have hit the pass button.

Since the players can't communicate and don't know anything about each other, the easiest way for both to type the same string is by typing something related to the common image. Notice, however, that the game doesn't ask the players to describe the image: all they know is that they have to "think like each other" and type the same string (thus the name "ESP"). *It turns out that the string on which the two players agree is typically a good label for the image,* as we will discuss in our evaluation section.

Figure 6. The ESP Game. Players try to "agree" on as many images as they can in 2.5 minutes. The thermometer at the bottom measures how many images partners have agreed on.

Taboo Words

A key element of the game is the use of taboo words associated with each image, or words that the players are not allowed to enter as a guess (see Figure 6). These words will usually be related to the image and make the game harder because they can be words that players commonly use as guesses. Imagine if the taboo words for the image in Figure 5 were “purse”, “bag”, “brown” and “handbag”; how would you then agree on that image?

Taboo words are obtained from the game itself. The first time an image is used in the game, it will have no taboo words. If the image is ever used again, it will have one taboo word: the word that resulted from the previous agreement. The next time the image is used, it will have two taboo words, and so on. (The current implementation of the game displays up to six different taboo words.)

Players are not allowed to type an image’s taboo words, nor can they type singulars, plurals or phrases containing the taboo words. The rationale behind taboo words is that often the initial labels agreed upon for an image are the most general ones (like “man” or “picture”), and by ruling those out the players will enter guesses that are more specific. Additionally, taboo words guarantee that each image will get many *different* labels associated with it.

Labels and Good Label Threshold

The words that we use as labels are the ones that players agree on. Although there is additional information that could be utilized (i.e., all other guesses that the players enter), for the purposes of this thesis such information will be ignored. We use only words that players agree on to ensure the quality of the labels: agreement by a pair of independent players implies that the label is probably meaningful. In fact, since these labels come from different people, they have the potential of being more robust and descriptive than labels that an individual indexer would have assigned [46].

To increase the probability that a label for a particular image is meaningful, we can utilize a *good label threshold*. This means that before a label is attached to the image (or used as a taboo word), it must have been agreed upon by at least X number of pairs, where X is the threshold. The threshold can be lenient and extremely low (X=1, one pair agreeing makes a label acceptable) or strict and high (X=40, forty pairs must have agreed on that label before it is attached to the image and made a taboo word). Of course, the lower the good label threshold is, the faster we can assign labels to the entire Web.

When is an Image "Done"?

As a particular image passes through the ESP game multiple times, it will accumulate several labels that people have agreed upon. The question is, at what point is an image considered to have been completely labeled and thus no longer used in the game? Our answer to this question is to remove an image from the game when it is no longer enjoyable to guess its contents with a partner. This will occur when a particular image has acquired an extensive list of taboo words, such that pairs are unable to agree on new labels and consistently ask their partners to pass on the image. Repeated passing notifies the system that an image should no longer be used for the game at that point in time. (Repeated passing might also indicate that the image is too complex to be used in the game; in which case the image should also be removed.)

Fully labeled images should be re-inserted into the game when several months have passed because the meaning of the images may have changed due to maturation effects. The English language changes over time, as do other languages. We want to capture the labels appropriate to an image, and thus if the language referring to that image changes over time, so should our labels. In addition to changes in language, cultural changes may occur since a particular image has last been labeled. Thus a picture of something or someone that was labeled as "cool" or "great" six months prior may no longer be considered to be so. For example, an image of Michael Jackson twenty years ago might have been labeled as "amazing" whereas today it might be labeled as "guilty."

Implementation and Other Details

The current version of the game is implemented as a Java applet and can be played at http://www.espgame.org. The applet connects to a centralized *game server*, which is responsible for the following: pairing up the players, providing each pair with a set of 15 different images with their corresponding taboo words, comparing the players' guesses (currently, guesses can only be 13 characters long), and storing all the information. The game server starts a game every 30 seconds: when a new player logs in, it waits until the next 30-second boundary to pair them with another player and start their game. This is done to make sure that players get paired at random and cannot cheat by logging in at the same time as their friends.

The current implementation is complete except that only 350,000 images are available for playing (rather than all images on the Web). We currently use a good label threshold of X=1.

Pre-Recorded Game Play

Our implementation does not require two people to be playing at the same time: a single person can play with a pre-recorded set of actions as their "partner." This set of actions is recorded at an earlier time when

two other people were playing the game simultaneously. For each image, every guess of each partner was recorded, along with timing information. We refer to the set of pre-recorded actions as the "bot." Having pre-recorded game play is especially useful when the game is still gaining popularity. When there are few players, only a single person may be playing the game at a time.

Notice that pre-recorded game play does not necessarily stop the labeling process. If the single player and the bot agree on the label that was agreed on when the actions were recorded, we can increase our confidence regarding that label. If the single player and the bot match on another word, we get a brand new label.

Cheating

It is imperative that partners not be able to communicate with each other; otherwise agreeing on an image would be trivial. Similarly, players could cheat by being partnered with themselves or by a large group of them agreeing on a unified strategy (for instance, all the players could agree to type "a" on every image; this could be achieved by posting this strategy on a popular website). The current implementation has several mechanisms in place to counter such cheating.

Notice first that no form of cheating is very likely: the game is meant to be played by hundreds, if not thousands, of people at once, most of which will be in distributed locations. Since players are randomly paired, they will have no information about who their partner is, and they will have no way to previously agree on a strategy. The probability of two cheaters using the same strategy being paired together should be low.

That being said, several additional steps are taken to minimize cheating. First, IP addresses of players are recorded and must be different from that of their partner to further prevent players from being paired with themselves. Second, to counter global agreement of a strategy (e.g., "let's all type 'a' for every image"), we use pre-recorded game-play. Massive global agreement of a strategy can be easily detected by measuring the average time in which players are agreeing on images: a sharp decrease in this average time should indicate agreement on a strategy. If a massive agreement strategy is in place, having a majority of the "players" be bots acting out pre-recorded sets of actions will make cheating impossible. Once people realize that the massive agreement strategy doesn't work, they will probably stop using it and we can lessen the use of pre-recorded game play.

An alternative mechanism that can be implemented is to enforce taboo words across an entire session. A pair's answer to an image could become a taboo word for the duration of their session together. This, coupled with a good label threshold greater than one ($X > 1$) would also prevent global agreement of a strategy from corrupting our labels. If the strategy was to always type "a" on an image, it would only work

for the first image, as “a” would become a taboo word for the duration of the session. If the strategy was something more complicated, like “type ‘one’ for the first image, ‘two’ for the second, etc”, then the labels couldn’t be corrupted because of the good label threshold: in order for “one” to become a label for a certain image, the image would have to occur X times as the first image in games played by cheaters using the same strategy.

We also remark that some amount of cheating is acceptable for certain applications of our labels. In the case of image search, for instance, we expect to see an improvement over the current techniques even if some of the labels are meaningless. The current techniques, which associate most of the text on a website to each image, generate many inappropriate labels.

Selecting the Images

We believe that the choice of images used by the ESP game makes a difference in the player’s experience. The game could perhaps be less entertaining if all the images were chosen from a single site and were all extremely similar.

The most basic strategy for picking the images is to select them at random from the Web using a small amount of filtering. This is the strategy employed in the current implementation of the game, except for two minor differences. First, once an image is randomly chosen from the Web, we reintroduce it into the game several times until it is fully labeled. Second, rather than picking the images from the Web in an online fashion, we collected 350,000 images in advance and are waiting until those are fully labeled to start with the whole Web. The images were chosen using “Random Bounce Me” [50], a website that selects a page at random from the Google database [26]. “Random Bounce Me” was queried repeatedly, each time collecting all JPEG and GIF images in the random page, except for images that did not fit our criteria: blank images, images that consist of a single color, images that are smaller than 20 pixels on either dimension, and images with an aspect ratio greater than 4.5 or smaller than 1/4.5. This process was repeated until 350,000 images were collected. The images were then rescaled to fit the game applet. For each session of the game, we choose 15 *different* images from our set of 350,000.

Spelling

The game server is equipped with a 73,000-word English dictionary that alerts players when they have misspelled a word. It does so by displaying the misspelled word in yellow rather than in white in the “Your Guesses” area (Figure 6). This is useful when one of the players doesn’t know how to spell a word, or makes a typing mistake.

Extension: Context-Specific Labels

Presenting images randomly selected from the Web to a wide-ranging audience is likely to result in labels that are general. There might be more specific labels for some images, which could be obtained if the correct population of users was doing the labeling. For example, when presented with pictures of faculty members at a certain university, the average ESP game player might enter general words such as man, woman, person, etc. However, if the users playing the ESP game were all students at that university, they might input faculty member names.

In order to generate these kinds of specific labels for certain categories of images, we suggest the usage of "theme rooms" for the ESP game. These more specific theme rooms can be accessed by those who wish to play the ESP game using only certain types of images. Some players might want images from certain domains or with specific types of content (e.g., images of paintings). Images for these theme rooms can be obtained using either Web directories or the labels generated during the "general category" ESP game. The labels generated in such theme rooms are likely to be more specific and thus more appropriate for certain applications. In the "art" theme room, for instance, images of paintings could be labeled with the names of their creators and maybe even the year in which they were made.

Notice, however, that proper general labels will already provide a vast improvement for many applications. For the visually impaired, for example, knowing that an image has a man in it is better than not knowing anything about it. The current version of the game implements the "general category" ESP game.

Inappropriate Content

A fractions of images on the Web are inappropriate for children (e.g., pornography). This means that the "general category" ESP game may also be inappropriate for children. Our suggested solution to this problem uses theme rooms as described above: children would only be allowed to play the "children's version" of the game. This version would obtain its images from the general category ESP game. Only images that have obtained a certain number of labels can go to the children's version; all of the labels for these images must be "safe." To be more rigorous, we can combine this with text-based filtering. Images coming from web pages containing inappropriate words, etc., would not be allowed. We believe these strategies would prevent inappropriate images from reaching the children's version. Notice also that the actual percentage of freely accessible images on the Web that are pornographic is small (the exact number of such images is hard to estimate, and varies depending on the source). Our game will only display images that are freely accessible.

Evaluation

We present data supporting our claims that people will want to play the ESP game and that the labels it produces are useful. In general it is difficult to predict if a game will become popular. One approach, which we followed early on, is to ask participants a series of questions regarding how much they enjoyed playing the game. Our data were extremely positive, but we follow a different approach: we present usage statistics from arbitrary people playing our game online.

We also present evidence that the labels produced using the game are indeed useful descriptions of the images. It's not the case that players *must* input words describing the images: players are never asked to describe anything. We show, however, that players do input words describing the images. To do so, we present the results of searching for randomly chosen keywords and show that the proportion of appropriate images when searching using the labels generated by the game is extremely high. In addition, we present the results of a study that compares the labels generated using the game to labels generated by participants that were asked to describe the images.

Usage Statistics

At the time of writing this thesis, the ESP Game has been running over the Web for roughly 2 years, allowing independent users to sign up for accounts and play the game. The game was first posted on the website on August 9 of 2003 and the statistics here presented are for the four-month period ending on December 10. A total of 13,630 people played the game during this time, generating 1,271,451 labels for 293,760 different images. Over 80% of the people played in more than one occasion (by this we mean that moe than 80% of the people played on different dates). Furthermore, 33 people played more than 1,000 games (this is about 50 hours!).

We believe these numbers provide evidence of how much fun the game is: almost 1.3 million labels were collected with only 13,630 players, some of which spent over 50 hours playing the game!

Labeling Rate

The usage statistics also allowed us to determine the rate at which images are labeled. The average number of images labeled per minute was 3.89 (std. dev. = 0.69). At this rate, 5,000 people playing the ESP game 24 hours a day would label all images on Google (425,000,000) in 31 days. This would only associate one word to each image. In 6 months, 6 words could be associated to every image. Notice that this is a perfectly reasonable estimate: on a recent weekday afternoon, the authors found 107,000 people playing in Yahoo! Games [69], 115,000 in MSN's The Zone [38] and 121,000 in Pogo.com [22]. A typical game on these sites averages over 5,000 people playing at any one time.

The time it takes players to agree on an image depends on the number of taboo words associated with the image. Our calculation of the labeling rate, however, is independent of the number of taboo words: every session of the game has roughly the same number of images with 0 taboo words, the same number of images with 1 taboo word, etc.

Quality of the Labels

We give evidence that players do input appropriate labels for the images, even though they are only trying to maximize their score in the game. We show the results of three distinct evaluations. The first is a measure of precision when using the labels as search queries. The second compares the labels generated using the game to labels generated by expreimental participants asked to describe the images. The third consists of asking experimental participants whether the labels generated using the game were appropriate with respect to the images.

Search Precision

We performed an evaluation similar to that of [31]: we examined the results of searching for all images associated to particular labels. To do so, we chose 10 labels at random from the set of all labels collected using the game. We chose from labels that occurred in more than 8 images.

Figure 7 shows the first 14 images having the label "car" associated with them: all of them contain cars or parts of cars. Similar results were obtained for the other 9 randomly chosen labels: dog, man, woman, stamp, Witherspoon (as in "Reese Witherspoon"), smiling, Alias (the TV show), cartoon, and green.

100% of the images retrieved indeed made sense with respect to the test labels. In more technical terms, the *precision* of searching for images using our labels is extremely high. This should be surprising, given that the labels were collected not by asking players to enter search terms, but by recording their answers when trying to maximize their score in the ESP game.

Comparison to Labels Generated by Participants Asked to Describe the Images

To further determine whether the words that players agreed on were actually describing the image, we asked 15 participants to input word descriptions of images and we compared their descriptions to the labels generated by the game. The participants were between 20 and 25 years of age and had not played the game during the trial period.

Method. Twenty images were chosen at random out of the first 1,023 images that had more than 5 labels associated to them by the game (1,023 is the number of images that had more than 5 labels associated to them at the time this experiment was performed). All 15 participants were presented with each of the 20 images in randomized order. For each image, the participant was asked to do the following:

Please type the six individual words that you feel best describe the contents of this image. Type one word per line below; words should be less than 12 characters.

Results. The results indicate that indeed players of the ESP game were generating descriptions of the images. For all (100%) of the 20 images, at least 5 (83%) of the 6 labels produced by the game were covered by the participants (i.e., each of these labels was entered by at least one participant). Moreover, for all (100%) of the images, the three most common words entered by participants were contained among the labels produced by the game.

Figure 7. All 14 images that had the label "car" associated to them by the ESP game (some of them have been slightly cropped to fit the page better).

Manual Assessment of the Labels

In addition to the previous evaluations, we had 15 participants complete a questionnaire about the quality of the labels generated using the game. The participants were chosen as independent raters because they had not played the ESP game. None of the participants of this evaluation took part in the previous one and vice-versa. All participants were 20 to 25 years of age.

Method. Twenty images were chosen at random out of the first 1,023 images that had more than 5 labels associated to them by the game. All 15 participants were presented with each of the 20 images in randomized order. For each image the participant was shown the first six words that pairs agreed on for that image during the game, as shown in Figure 8. For each of the 20 image-word sets they were asked to answer the following questions:

1. How many of the words above would you use in describing this image to someone who couldn't see it.
2. How many of the words have **nothing** to do with the image (i.e., you don't understand why they are listed with this image)?

Dog
Leash
German
Shepard
Standing
Canine

Figure 8. An image with all its labels.

Results. For question 1, the mean was 5.105 words (std. dev. 1.0387), indicating that a majority (or 85%) of the words for each image would be useful in describing it. The mean for question 2 was 0.105 words (std. dev. 0.2529), indicating that for the most part subjects felt there were few (1.7%) if any labels that did not belong with each image.

Previous Techniques for Processing Images

To this point, we have presented a method for labeling images on the Web and we have presented evidence that it does indeed produce high-quality labels. There are a variety of other techniques for processing images on the Web, all of which are different in nature from ours. We now survey the different techniques and contrast them with our method.

Computer Vision

There has been considerable work in computer vision related to automatically labeling images. The most successful approaches *learn* from large databases of annotated images. Annotations typically refer to the contents of the image, and are fairly specific and comprehensive (and building these kinds of databases is expensive). Methods such as [9,10] cluster image representations and annotations to produce a joint distribution linking images and words. These methods can predict words for a given image by computing

the words that have a high posterior probability given the image. Other methods attempt to combine large semantic text models with annotated image structures [21]. Though impressive, such algorithms based on learning don't work very well in general settings and work only marginally well in restricted settings. For example, the work described in [21] only gave reasonable results for 80 out of their 371 vocabulary words (their evaluation consisted of searching for images using the vocabulary words, and only 80 of the words resulted in reasonable images).

A different line of research attempts to find specific objects in images. [53], for instance, introduced a method to locate human faces in still photographs. These algorithms are typically accurate, but have not been developed for a wide range of objects. Additionally, combining algorithms for detecting specific objects into a single general-purpose classifier is a non-trivial task.

The ESP game provides a possible solution to the image-labeling problem, but having a computer program that can label images remains a more important goal. One application of the ESP game is in the creation of such a program: the limitations of the current computer vision techniques partly arise from the lack of large databases of annotated images, which could be constructed using methods similar to our game. Work done by Takeo Kanade and Shingo Uchihashi [29,63] shows that such an approach is quite successful.

Text, Authority and Content-Based Image Retrieval

The World Wide Web contains millions of images and finding effective methods to search and retrieve from among them has been a prevalent line of research, both academically ([17,31]) and in industry ([1,26]). Text-based image retrieval systems such as [1] annotate images with text derived from the HTML documents that display them. The text can include the caption of the image (which is rarely present), text surrounding the image, the entire text of the containing page, the filename of the containing HTML document, and the filename of the image itself. More recent proposals such as [31,26] also make use of the link structure of the Web to assign "authority" values to the images. Images that come from more authoritative web pages (e.g., pages with higher PageRank [26]) are displayed before images coming from less authoritative pages. This improves the quality of the results by typically showing more relevant images first. Another possibility that has been explored involves combining text-based systems with computer vision techniques as in [17]. This approach allows different types of queries to be processed (e.g., similarity queries).

Most image search engines rely on the images being related to the text in the web page. This heuristic works in part because almost any simple query matches thousands of images on the Web — the results appear satisfactory even if the search engine finds only half of all the relevant images. However, not all queries are handled properly, and finding a particular image on the Web can be impossible. Figure 9, for

instance, illustrates an example of Google Image Search [26] returning a picture of a map of Chicago as the first result on the query "car."

Figure 9. First result from Google Images on the query "car"

(http://maps.uchicago.edu/directions/graphics/car.gif)

We argue that our game can improve the quality of image retrieval systems by providing meaningful labels that are independent of the content of the websites.

Inappropriate Content Filters

Inappropriate content filters such as [43] attempt to block certain images from being displayed. Typically these filters try to block pornographic sites from reaching children at home or employees in the workplace. Since computer vision techniques for this purpose are not highly accurate [24], content filters usually analyze the text inside web pages to determine whether they should be blocked.

Most filters are reasonably accurate, but have several flaws. First, they only work for a few languages and in most cases only work for pages in English. Second, they work poorly when the pages don't contain any "incriminating" text: a page with a nude image and nothing else in it would not be correctly identified. For this reason, in order to ensure that inappropriate content does not get posted, dating services and websites that allow users to post images have to hire people to look over every single picture to be posted. Third, content filters have to be constantly updated: imagine what happens when a new porn star named Thumbelina comes out; suddenly every search for "Thumbelina" will return pornography. Google Image Search [26] offers a content filter (called SafeSearch), which attempts to block all inappropriate images from being displayed in their search results. At the time of writing this thesis, a query for "interracial" returns several inappropriate images (and a more direct query like "wet tshirt" returns even more inappropriate results). We argue that having proper labels associated to each freely available image on the Web would improve content filtering technology.

Using Our Labels

This thesis is mostly concerned with obtaining appropriate labels for images, and not with how these labels should be used. In the case of image search, building the labels into the current systems is not difficult, since they can be thought of as HTML captions or text appearing right next to the image. This naïve

strategy would already signify an improvement over the current techniques, as there would be more useful data to work with. More intelligent techniques could be conceived, such as assigning a higher weight to labels coming from the ESP game as opposed to regular HTML captions, or a numerical weight based on the "good label threshold". However, arriving at an optimal strategy for using the labels is outside the scope of this thesis and is left as future work.

In the case of providing textual descriptions for the visually impaired, using the labels is slightly less trivial. Our game produces labels, not explanatory sentences. While keyword labels are perfect for certain applications such as image search, other applications such as accessibility would benefit more from explanatory sentences. Nevertheless, having meaningful labels associated to images for accessibility purposes is certainly better than having nothing. Today's screen-reading programs for the visually impaired use only image filenames and HTML captions when attempting to describe images on the Web — the majority of images on the Web, however, have neither captions nor extremely descriptive filenames [39]. We propose that all the labels be available for use with screen readers and that users determine themselves how many labels they want to hear for every image. Again, extensive tests are required to determine the optimal strategy.

Chapter Discussion

The ESP game is a novel interactive system that allows people to label images while enjoying themselves. We have presented evidence that people will play our game and that the labels it produces are meaningful. Our data also suggest that 5,000 people playing the game for 24 hours a day would enable us to label all images indexed by Google in a matter of weeks. This is striking because 5,000 is not a large number: most popular games on the Web have more than 5,000 players at any one time. Having proper labels associated to each image on the Web could allow for more accurate image retrieval, could improve the accessibility of sites, and could help users block inappropriate images.

Although the main application of the ESP game is to label images (which in itself has applications in areas such as accessibility), our main contribution stems from the way in which we attack the labeling problem. Rather than developing a complicated algorithm, we have shown that it's conceivable that a large-scale problem can be solved with a method that uses people playing on the Web. We've turned tedious work into something people want to do.

We will show in the next few chapters that many other problems can be attacked in the same way.

Chapter 4: Peekaboom*

As we have seen, humans understand and analyze everyday images with little effort: what objects are in the image, where they are located, what is the background, what is the foreground, etc. Computers, on the other hand, still have trouble with such basic visual tasks as reading distorted text or finding where in the image a simple object is located. Although researchers have proposed and tested many impressive algorithms for computer vision, none have been made to work reliably *and* generally.

Most of the best approaches for computer vision (e.g. [9,10,53]) rely on machine learning: train an algorithm to perform a visual task by showing it example images in which the task has already been performed. For example, training an algorithm for testing whether an image contains a dog would involve presenting it with multiple images of dogs, each annotated with the precise location of the dog in the image. After processing enough images, the algorithm learns to find dogs in arbitrary images. A major problem with this approach, however, is the lack of training data, which are prepared by hand. Databases for training computer vision algorithms currently have hundreds or at best a few thousand images [59] — orders of magnitude less than what is desired.

In this Chapter we address the problem of constructing a massively large database for training computer vision algorithms. The target database will contain millions of images, all fully annotated with information about what objects are in the image, where each object is located, and how much of the image is necessary to recognize it.

To construct such a database, we follow the approach taken by the ESP Game and introduce a new game called Peekaboom. Peekaboom is an extremely enjoyable networked game in which, simply by playing, people help construct a database for training computer vision algorithms. We guarantee the database's correctness even if the people playing the game don't intend it. As we will show, our game is also very enjoyable, with some people having played over 40 hours a week. We will further show that this game can be used to improve image-search results and to calculate object bounding-boxes similar to those in Flickr [25].

The ESP Game collects random images from the Web and outputs word labels describing the contents of the images. Given an image, the ESP Game can be used to determine what objects are in the image, but cannot be used to determine where in the image each object is located. Such location information is necessary for training and testing computer vision algorithms, so the data collected by the ESP Game is not

* Adapted from [6].

sufficient for our purposes. The game introduced in this Chapter, Peekaboom, improves on the data collected by the ESP Game and, for each object in the image, outputs precise location information, as well as other information useful for training computer vision algorithms. By playing a game, people help us collect data not because they want to be helpful, but because they have fun. Indeed Peekaboom (or the ESP Game or any game built on this premise) can be treated as a human algorithm: on input an image, it outputs (with arbitrarily high probability) a correct annotation of the image. Instead of using a silicon processor, this "algorithm" runs on a processor consisting of regular humans interacting throughout the Web.

Basic Game Play

Peekaboom, as the name may suggest, is a game with two main components: "Peek" and "Boom." Two random players from the Web participate by taking different roles in the game — when one player is Peek, the other is Boom. Peek starts out with a blank screen, while Boom starts with an image and a word related to it (see Figure 10).

The goal of the game is for Boom to reveal parts of the image to Peek, so that Peek can guess the associated word. Boom reveals circular areas of the image by clicking. A click reveals an area with a 20-pixel radius. Peek, on the other hand, can enter guesses of what Boom's word is. Boom can see Peek's guesses and can indicate whether they are hot or cold.

When Peek correctly guesses the word, the players get points and switch roles; play then proceeds on a new image-word pair. Each session of the game lasts four minutes, and players must go through as many image-word pairs as possible. If the image-word pair is too difficult, the two players can "pass," or opt out, of the current image. Passing creates the same effect as a correct guess from Peek, except that the players get no points.

To maximize points, Boom has an incentive to reveal only the areas of the image necessary for Peek to guess the correct word. For example, if the image contains a car and a dog and the word associated to the image is "dog," then Boom will reveal only those parts of the image that contain the dog. Thus, intuitively, given an image-word pair, data collected from the game yields the area of the image pertaining to the word.

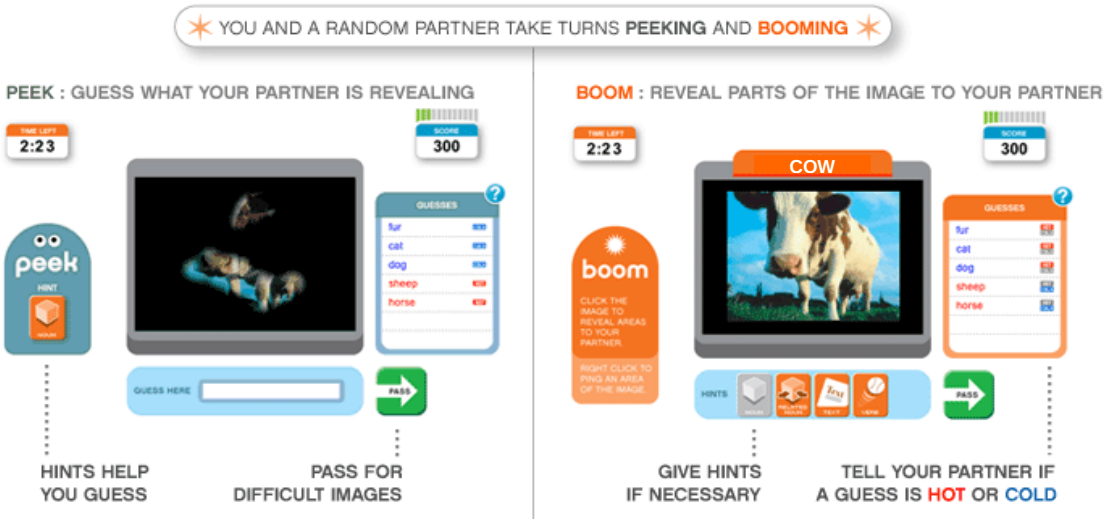

Figure 10. Peek and Boom.

Pings

Another component of the game is "pings" — ripples that appear on Peek's screen when Boom right-clicks on the image (see Figure 11). If two players were playing with the image on Figure 2, then many correct words are possible from Peek's point of view: elephant, trunk, tusk, ear. Suppose the correct word is "trunk." To get Peek to guess correctly, Boom can "ping" the trunk of the elephant by right-clicking on it. In doing so, Boom helps to disambiguate the trunk from the rest of the elephant.

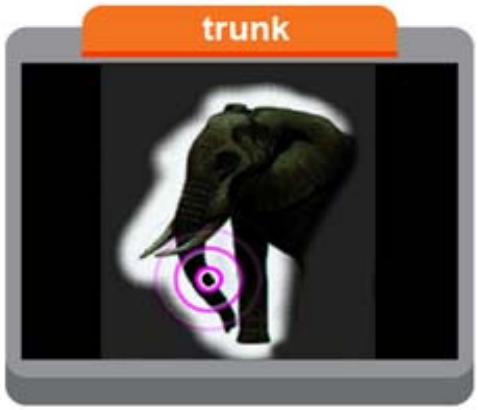

Figure 11. Pings. To help Peek, Boom can "ping" parts of the image by right-clicking on them.

Figure 12. Hints. Boom can help Peek by giving hints about how the word relates to the image.

Hints

Another feature of the game are buttons that allow Boom to give hints to Peek about how the word relates to the image (See Figures 10 and 12). Upon Boom's pressing of one of the hint buttons, a corresponding flashing placard appears on Peek's screen. The reason for having hints is that often the words can relate to the image in multiple ways: as nouns, verbs, text, or related nouns (something not in the image, but related to it).

Where Do The Images and Words Come From?

All words presented to the players are related to their corresponding image. On input an image-word pair, Peekaboom outputs a region of the image that is related to the word. As previously stated, our goal is to collect location information about millions of objects in images. An important question remains: Where do we obtain millions of images along with words that correctly describe the contents of the images? The answer lies in the ESP Game (Chapter 3). We use the labels collected from the ESP Game as the words we present to the players in Peekaboom.

Game Points and the Bonus Round

Although the exact number of points given to the players for different actions is not important, we mention it to show the relative proportions. Furthermore, we mention the different point strategies used by Peekaboom to keep players engaged.

Points are given to both Peek and Boom equally whenever Peek guesses the correct word. In the current implementation, both obtain 50 points. Points are not subtracted for passing. Points are also given to both Peek and Boom for using the hint buttons. Although this might appear counterintuitive since using hints deducts points in many other games, we actually *want* the players to use the hint buttons. As mentioned above, hints give us additional information about the relationship between the word and the image, and therefore we encourage players to use them. Twenty-five extra points are given to both Peek and Boom

whenever Peek guesses the correct word and Boom had used a hint. Points are not given for usage of the hot/cold buttons.

Every time the players correctly complete four images, they are sent to a "bonus" round. The bonus round is different in nature from the rest of the game and allows players to obtain up to 150 points. In the bonus round (see Figure 13), players simply click on an object in the image. The closer they are to each other's clicks, the more points they get. For example, both players could obtain an image of a car and be told "click on the car."

Figure 13. The Peekaboom Bonus Round. Players must click on the specified object within the image; they obtain points proportional to how close their clicks are to each other's clicks.

Players obtain between 0 and 10 points for every click in the bonus round, depending on how far the click is from their partner's corresponding click. The bonus round is timed: players have to click on the same place as their partner as many times as they can in 5 seconds. If the object is not in the image, players can pass. Because some images do not contain the object related to the word, passing in the bonus round generates 25 points for both players (so we can learn whether the object is there). Players cannot pass after they have clicked on the image.

There are two reasons for the Peekaboom bonus round. First, by giving players "bite-size" milestones (getting four images correctly), we reinforce their incremental success in the game and thus encourage them to continue playing. Second, the bonus round is an alternative approach to collecting training data for computer vision. In it, players click inside specific objects within an image. Such clicks give additional information for training computer vision algorithms. In this thesis we do not concern ourselves with such information, but remark that it is also useful.

Collecting Image Metadata

Our goal is to construct a database for training computer vision algorithms. Here we discuss exactly what information is collected by Peekaboom and how it is collected.

On input an image-word pair (coming directly from the ESP Game), Peekaboom collects the following information:

- **How the word relates to the image.** Is it an object, person, or animal in the image, is it text in the image, is it a verb describing an action in the image, is it an object, person, or animal not in the image but related to it? The ESP Game associates words to images, but does not say how the word is related to the image. Figure 12, for instance, shows multiple ways in which a word can be related to an image. Hint buttons in Peekaboom allow us to determine the relation of the word to the image. This is useful in multiple ways, but for the purposes of constructing training sets for computer vision, it allows us to weed out "related nouns" and to treat "text" separately.
- **Pixels necessary to guess the word.** When Peek enters the correct word, the area that Boom has revealed is precisely enough to guess the word. That is, we can learn exactly what context is necessary to determine what the word refers to. This context information is absolutely necessary when attempting to determine what type of object a set of pixels constitutes (see Figure 14).

Figure 14. The image on the left contains a car driving through the street, while the one on the right has a person crossing the same street. Both the car and the person are exactly the same set of pixels up to a rotation by 90 degrees. (Example taken from [60].)

- **The pixels inside the object, animal, or person.** If the word is a noun directly referring to something in the image, "pings" give us pixels that are inside the object, person, or animal.
- **The most salient aspects of the objects in the image.** By inspecting the sequence of Boom's clicks, we gain information about what parts of the image are salient with respect to the word. Boom typically reveals the most salient parts of the image first (e.g., face of a dog instead of the legs, etc.).

- **Elimination of poor image-word pairs.** If many independent pairs of players agree to pass on an image without taking action on it, then likely they found it impossibly hard because of poor picture quality or a dubious relation between the image and its label. By implementing an eviction policy for images that we discover are "bad," we can improve the quality of the data collected (as well as the fun level of the game).

When multiple players have gone through the same image, these pieces of information can be combined intelligently to give extremely accurate and useful annotations for computer vision. Later, for example, we show how a simple algorithm can use the data produced by Peekaboom to calculate accurate object bounding-boxes (see Figure 16).

The Single Player Game

Peekaboom is a two-player game. Oftentimes, however, there will be an odd number of people attempting to play the game, so the remaining person cannot be paired. To reduce their frustration, we also have a single-player version of the game in which the player is matched with a server-side "bot."

Our bot acts intelligently to simulate a human player by being based on pre-recorded games. In other words, we take data collected from pairs of humans and use it as the basis for the computer player's logic. Emulating a Boom player is fairly simple: the bot can regurgitate the sequence of recorded clicks to the human. Emulating Peek is much more complicated; the bot needs to have some concept of closeness of the human's clicks to the set of recorded clicks. For instance, if the human does not reveal the dog in the picture, the bot should not guess "dog." Our bot only reveals a certain pre-recorded guess if enough area has been revealed. Towards this end, it employs a spatial data structure whose members are circles, each of which corresponds to a click. Elements of the data structure are removed as they are clicked on by the human player. When the data structure becomes empty, the bot gives the correct answer. Moreover, it has the ability to make incorrect guesses along the way, based on the relative emptiness of the spatial data structure.

Cheating

Peekaboom is a collaborative game: partners work together to maximize their score. When both partners do not communicate outside the game environment, we can obtain correct information. However, if the two partners collude to cheat on the game, the data could be poisoned. For instance, if Boom and Peek know each other and have an outside means of communication, then Boom can simply tell Peek what words to type.

Peekaboom contains multiple anti-cheating mechanisms. Through a combination of online in-game enforcement and offline analysis, we are able to detect and deal with cheating. Before detailing Peekaboom's anti-cheating measures, we mention that cheating attempts are uncommon. Although a minority of players might obtain satisfaction from "gaming the system," the majority of them just want to play the game honestly. Indeed, as anecdotal evidence, when Peekaboom was tested in a room with children of ages 9-13, they would cover the word with their hand to prevent others in the room from seeing the answers. Nevertheless, Peekaboom does have a full set of measures to prevent collusion.

- **The player queue.** When players log on to the game server, they are not immediately paired off. Instead, the server makes them wait *n* seconds, where *n* is the number of seconds until the next matching interval. Currently, matching intervals happen every 10 seconds, and when they do, the server matches everyone in the queue with a partner (any odd person out will be paired with a bot). With a large number of players in the system, we can ensure that a player's partner is random and reduce the likelihood of colluders getting matched just because they clicked "start playing" at the same time.

- **IP address checks.** We also check player's IP addresses to ensure that they are not paired with themselves or with others that have a similar address (similarity in IP addresses can imply geographical proximity).

- **Seed images.** Because our system is a web-based game, one point of concern is that bots (i.e. automated players) might play the game and pollute the pool of collected data. To detect them, we introduce seed images into the system; in other words, those for which we have hand-verified metadata. On being presented seed images, if a player consistently fails to click on the relevant parts when playing Boom or to guess the correct words when playing Peek, they will be added to a blacklist. We discard all current and future game play data associated with anyone on the blacklist. Notice that, almost by definition, a computer program cannot successfully play Peekaboom — if it were able to do so, then it would be able to recognize the objects in the images. Therefore, this strategy prevents bots (as well as otherwise malicious players) from poisoning our data.

- **Limited freedom to enter guesses.** Since Boom can see all of Peek's guesses, the game allows a limited form of communication between the players. Indeed, many of the Peekaboom players use the guess field as a way to communicate with their partner. It is not uncommon for the first guess in a game to be "hi" or for the first guess after passing on an image to be the correct word associated with the previous image. It is also not uncommon for players to type "sorry" after

taking too long on an image. A possible cheating strategy is to exchange IM screen names through the guess field and then, using IM, communicate the correct words. Although we have never observed attempts to execute such a strategy, we can mitigate it by not allowing Peek to enter any non-alphabetical characters (such as numbers). Similarly, we can prevent Boom from seeing any guesses that are not words in the dictionary (currently we do allow Boom to see such guesses because we have not seen players attempt to cheat in this way).

- **Aggregating data from multiple players.** In addition to the above strategies, we aggregate data from multiple players for a given image-word pair. By doing this, we can simply eliminate any outliers.

Implementation

We implemented the architecture of the game under the client-server model. The client application is delivered as a Java applet, while the server is written purely in Java. Applets connect to a server, http://www.peekaboom.org, which then matches the players with games of Peekaboom. Upon two players' completion of a match, the server writes their game play data and scores to disk. We then compile the collected data into desired formats.

Our implementation of the game contains many features to improve game-play:

- **Spelling check.** Incorrectly spelled words are displayed in a different color to notify players. This is important because the Peek player usually types multiple guesses in a short time, often making spelling mistakes.

- **Inappropriate word replacement.** Since Boom can see Peek's guesses, we do not allow Peek to enter inappropriate words. Whenever one of Peek's guesses is among a list of possible inappropriate words, we substitute it with another word chosen from a list of innocent words such as "love," "caring," "ILuvPeekaboom," etc. The same is done for misspelled variants of inappropriate words.

- **Top scores list and ranks.** The Peekaboom website prominently displays the cumulative top scores of the day as well as the top scores of all time. Furthermore, players are given a rank based on the total number of points they have accumulated throughout time (see Figure 15). The different ranks are: Fresh Meat (0-15,000 points), Newbie (15,000-75,000 points), Player (75,000-250,000 points), Gangster (250,000-1,000,000 points) and Godfather (1,000,000 or more points).

  We remark that ranks have proven an important component of Peekaboom's incentive strategy. Of the 15,000+ players that have obtained an account, 47% of them have scores that are less than

5,000 points above of the rank cutoffs. Given that these intervals cover less than 2% of the space of possible cumulative scores, this strongly suggests that many players simply play to reach a new rank.

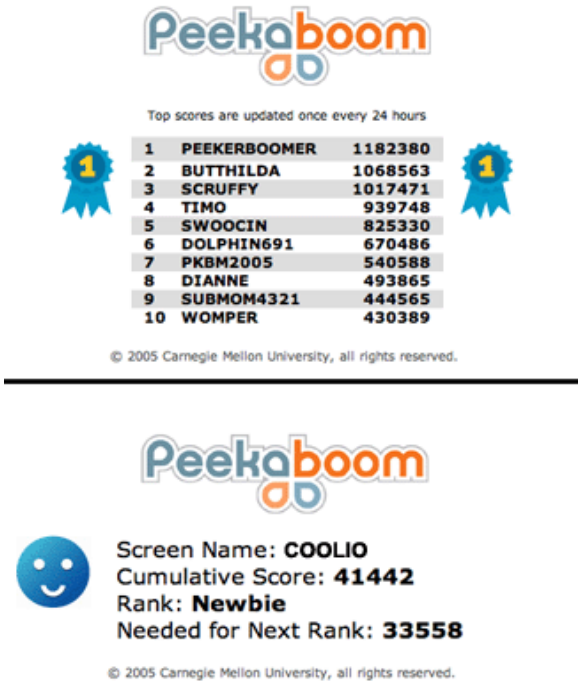

Figure 15. Top scores and player ranks. Players are shown their current rank and the number of points remaining for the next rank.

Additional Applications

Before going to the evaluation section, we mention two additional applications for the data collected by Peekaboom. A benefit of these applications is that they are "direct" in that they do not require the training of machine learning algorithms.

Improving Image-Search Results

Peekaboom gives an accurate estimate of the fraction of the image related to the word in question. This estimate can be calculated from the area revealed by Boom. The fraction of the image related to a word can be used to order image-search results: images in which the word refers to a higher fraction of the total pixels should be ranked higher. Much like the goal of the ESP Game is to label all images on the Web, we can imagine Peekaboom doing the same and thus further improving image-search.

Object Bounding-Boxes

In the same vein, Peekaboom can be used to directly calculate object bounding-boxes similar to those used in Flickr [25] (see Figure 16). Flickr is a photo sharing service that allows users to "tag" images with keywords and to associate keywords with rectangular areas in the image (the areas and tags, however, are not guaranteed to be correct since a user can enter anything they wish for their own images). To exhibit the power of the data collected by Peekaboom, we show how to use it calculate such rectangles. We emphasize, however, that the data collected by Peekaboom is significantly richer and that to calculate the rectangles, we discard vast amounts of the information collected by our game.

Since Peekaboom annotates arbitrary images on the Web, its data allows for an image search engine in which the results are highlighted (similar to the highlighted words in Google's text search results). Using the data obtained in the first two weeks of game-play, we have implemented a prototype of such a search engine (see Figure 16). The search engine can be accessed from the Peekaboom website: http://www.peekaboom.org.

The bounding boxes were calculated as follows. For a single play of an image-word pair, we create a matrix of 0s and 1s. The dimensions of the matrix are the same as the dimensions of the image (in pixels). At first, every entry in the matrix is a 0. We add a 1 in every pixel clicked by Boom, as well as in the circle of radius 20 pixels around the click. We thus obtain a matrix of 0s and 1s corresponding to the exact area that was revealed in a single game-play. Next, we combine different plays of the same image-word pair by adding their corresponding matrices. This gives a matrix whose entries are integers corresponding to the number of different players that revealed each pixel of the image. On this combined matrix, we apply a threshold of 2, meaning that we substitute every value less than 2 with 0 and every value greater than or equal to 2 with 2. This gives a matrix corresponding to all the pixels that have been revealed by at least 2 players. Next, we cluster these pixels and calculate the bounding boxes by taking, for each cluster, the leftmost, rightmost, topmost and bottommost points. This algorithm may produce multiple bounding-boxes for a single image-word pair. For instance, in Figure 16, we can see that many of the results for "eyes" have two-bounding boxes, one corresponding to each eye.

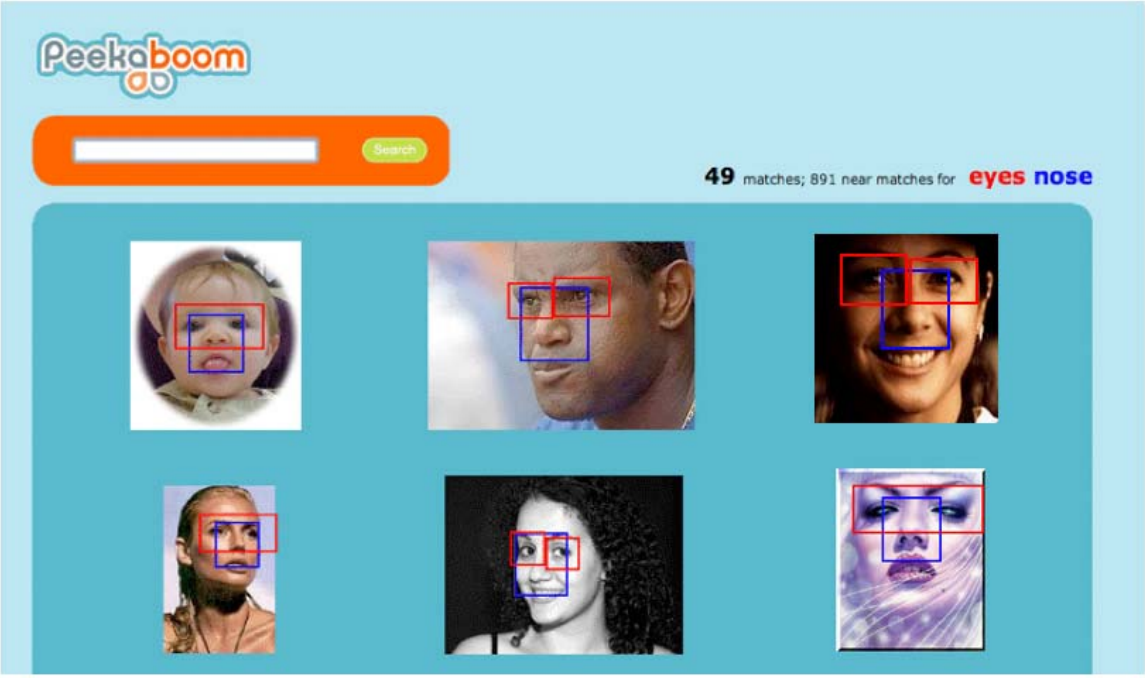

Figure 16. Object Bounding-Boxes calculated form Peekaboom data.

As we will see, the results produced by this simplistic algorithm are extremely accurate. Such results could be improved by making intelligent use of the additional data given by Peekaboom (such as pings, the precise order of the areas revealed, etc.), but for the purposes of this thesis, we use the simplistic algorithm.

Alternative: Using Ping Data for Pointing

Instead of showing bounding-boxes calculated from revealed areas, we could show arrows or lines pointing to the objects (see Figure 17). Such pointers can be easily calculated from the ping data. The simplest algorithm for doing so is to select a ping at random and assume it is a good pointer for the object. We will show that this simplistic algorithm gives very accurate results. (Figure 17 shows an image in which the different objects have been located using ping data.) More elaborate algorithms could give even better results. We remark, however, that simply "averaging" the pings over multiple players to obtain a single pointer does not give accurate results. For instance, if the object was "eyes," averaging the pings gives a pointer to a region that is not an eye. Averaging would also work poorly on concave objects.

Figure 17. Calculation of object pointers using pings.

Evaluation: User Statistics

The evaluation of our claims consists of two parts. First, we must show that the game is indeed enjoyable. Second, we must show that the data produced by the game is useful.

It is difficult to evaluate how enjoyable a game really is. One approach is to ask participants a series of questions regarding how much they enjoyed playing the game. Our data for such an approach were extremely positive, but we follow a different approach: we present usage statistics from arbitrary people playing our game online (this is the same approach that was used by the ESP Game).

Peekaboom was released to a general audience on August 1 of 2005. We present the usage statistics from the period starting August 1, 2005 and ending September 1, 2005. A total of 14,153 different people played the game during this time, generating 1,122,998 pieces of data. By "different people" we mean different user IDs. By a "piece of data," we mean a successful round of Peekaboom in which Peek correctly guessed the word given Boom's revealed region. It is important to mention that a single image-word pair can have multiple pieces of data associated to it if it occurs in multiple games. Moreover, we *require* an image-word pair to be displayed in multiple games before we consider it analyzed.

If 14,153 people gave us 1,122,998 pieces of data, then (since each data point was viewed by two people) on average each person played on 158.68 images. Since each session of the game lasts 4 minutes and on average players go through approximately 8.7 images during a session, in this one month period each person played on average 72.96 minutes (without counting the time spent waiting for a partner, etc.).

Over 90% of the people played on more than one occasion (by this we mean that more than 90% of the people played on different dates). Furthermore, every player in the top scores list played over 800 games

(that's over 53 hours without including the time they spent waiting for a partner!). These numbers undoubtedly attest to how enjoyable the game is.

User Comments

To give a further sense for how much the players enjoyed the game, we include below some quotes taken from comments submitted by players using a link on the website:

> "The game itself is extremely addictive, as there is an element of pressure involved in beating the clock, a drive to score more points, the feeling that you could always do better next time, and a curiosity about what is going to come up next. I would say that it gives the same gut feeling as combining gambling with charades while riding on a roller coaster. The good points are that you increase and stimulate your intelligence, you don't lose all your money and you don't fall off the ride. The bad point is that you look at your watch and eight hours have just disappeared!"

> "One unfortunate side effect of playing so much in such a short time was a mild case of carpal tunnel syndrome in my right hand and forearm, but that dissipated quickly."

> "This game is like crack. I've been Peekaboom-free for 32 hours now."

> "Unlike other games, Peekaboom is cooperative (rather than competitive). I love this!"

> "This game is fast-paced. My friends and I are totally addicted."

Evaluation: Accuracy of Collected Data

The usefulness of Peekaboom as a data-collection method rests in the quality of the data we collect. Although the design of the game inherently ensures correctness of the data, we wanted to test whether it is as good as what would be collected directly from volunteers in a non-game setting. To do so we conducted two experiments to test first the accuracy of the bounding boxes we defined, and second the utility of the pointing behavior in the game.

Experiment 1: Accuracy of Bounding Boxes

In the first experiment, we tested whether the bounding boxes for objects within an image that are calculated from Peekaboom are as good as bounding boxes people would make around an object in a non-game setting. We selected at random 50 image-word pairs from the data pool that had been successfully played on by at least two independent pairs of people. The images selected all had nouns as their word (as opposed to text in the image, or an adjective, etc.). All the images chosen had the word refer to a single object in the image. For each image, Peekaboom data was used to calculate object bounding boxes using the method explained in previous sections.

We then had four volunteers make bounding boxes around the objects for each image, providing us with 200 bounding boxes drawn by volunteers. The volunteers were asked, for each image, to draw a bounding box around the object that the word referred to. We then selected at random one of the four volunteer's bounding boxes for each image, so as to end up with one volunteer-generated bounding box for every one of the 50 images.

Finally, we tested the amount of overlap between the bounding boxes generated by Peekaboom and those generated by our volunteers. The amount of overlap was determined using the formula:

$$\text{OVERLAP}(A,B) = \text{AREA}(A \cap B) / \text{AREA}(A \cup B),$$

where A and B are the bounding boxes. Notice that if A=B then OVERLAP(A,B)=1 and if A is disjoint from B then OVERLAP(A,B)=0.

We calculated the average overlap across the 50 images, as well as the standard deviation.

Results

On average, the overlap between the Peekaboom bounding boxes and the volunteer-generated ones was 0.754, with standard deviation 0.109. This means that the Peekaboom bounding boxes were very close to those generated by the volunteers. To illustrate, we show in Figure 18 the bounding box that obtained the lowest overlap score (0.552).

Given that Peekaboom was not directly built to calculate bounding boxes, this shows the wide applicability of the data collected.

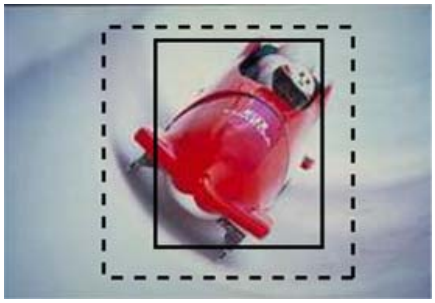

Figure 18. Experiment image with lowest overlap between a volunteer generated bounding-box (solid lines) and one generated by Peekaboom (dashed lines).

Experiment 2: Accuracy of Pings

In the second experiment, we tested whether the object pointers that are calculated from Peekaboom are indeed inside the objects. As in the previous experiment, we selected at random 50 image-label pairs from the data pool that have been successfully played on by at least two independent pairs of people. The images

selected all had the word as a “noun” (as opposed to as text in the image, or an adjective, etc.). All the images chosen had the word refer to a single object in the image. For each image, Peekaboom data was used to calculate object pointers using the method explained in previous sections.

We then asked three volunteer raters to determine, for each pointer, whether it was inside the object or not. The raters were shown examples of pointers inside and outside the object and were told that “near an object” does not count as inside the object.

Results

According to all the raters, 100% of the pointers were inside the object referred to by the word. This gives extremely positive evidence that ping data is accurate, especially since it was calculated using such a simplistic algorithm.

Related Work

We have presented a method for annotating arbitrary images on the Web and we have presented evidence that it produces high-quality data. We now survey the related work.

The ESP Game

As mentioned before, the ESP Game is two-player game that collects word labels for arbitrary images. Peekaboom is similar to the ESP Game and in fact was inspired by it. We consider Peekaboom an extension of the ESP Game. Whereas ESP gives data to determine which objects are in the image, Peekaboom can augment this data with information about *where* in the image such objects are located.

In terms of game mechanics, Peekaboom is different from the ESP Game in several ways. First, Peekaboom is *asymmetric*: whereas both players in the ESP Game are performing the same role, players of Peekaboom alternate in performing different roles. Second, Peekaboom allows a significantly higher level of interaction among the players. Whereas in the ESP Game, players cannot communicate at all, in Peekaboom one of the players can freely communicate with the other. Third, the usage of hint button has proven very successful in Peekaboom, and such buttons could as well be incorporated into ESP.

Such differences in game mechanics reflect the difference in purpose of Peekaboom and ESP.

LabelMe

LabelMe [51] is a web-based tool for image annotation. Anybody can annotate data using this tool and thus contribute to constructing a large database of annotated objects. The incentive to annotate data is the data itself. You can only have access to the database once you have annotated a certain number of images. The main difference between Peekaboom and LabelMe is the game aspect. Whereas LabelMe simply asks users to annotate an image, Peekaboom transforms the process into an enjoyable game. LabelMe relies on

people's desire to help and thus assumes that the entered data is correct. On the other hand, Peekaboom has multiple mechanisms to prevent players from polluting the data.

Chapter Discussion and Future Work

Peekaboom is a novel, complete game architecture for collecting image metadata. Segmenting objects in images is a unique challenge, and we have tailored a game specifically to this end. In the very near future, we would like to make our 1,000,000+ pieces of data available to the rest of the world by formatting it as an image segmentation library.

Like the ESP Game, Peekaboom encompasses much more than just a Java applet delivered from a website. Rather, the ideas behind the design and implementation of the game generalize to a way of harnessing and directing the power of the most intricate computing device in the world — the human mind.

Some day computers will be able to segment objects in images unassisted, but that day is not today. Today we have engines like Peekaboom that use the wisdom of humans to help naïve computers get to that point. The actual process of making computers smarter given segmentation metadata is beyond the scope of this thesis, since it would require a far more sophisticated interpretation of the data than the simple bounding box derivation we have presented.

Chapter 5: Phetch*

The Web is not built for the blind. Only a small fraction of major corporate websites are fully accessible to the disabled, let alone those of smaller organizations or individuals [64]. However, millions of blind people surf the Web every day, and Internet use by those with disabilities grows at twice the rate of the non-disabled [45].

One of the major accessibility problems is the lack of descriptive captions for images. Visually impaired individuals commonly surf the Web using "screen readers," programs that convert the text of a webpage into synthesized speech. Although screen readers are helpful, they cannot determine the contents of images on the Web that do not have descriptive captions. Unfortunately the vast majority of images are not accompanied by proper captions and therefore are inaccessible to the blind (as we show below, less than 25% of the images on the Web have an HTML ALT caption). Today, it is the responsibility of Web designers to caption images. We want to take this responsibility off their hands.

We set our goal to assign proper descriptions to arbitrary images. A "proper" description is *correct* if it makes sense with respect to the image, and *complete* if it gives sufficient information about its contents. Rather than designing a computer vision algorithm that generates natural language descriptions for arbitrary images (a feat still far from attainable), we opt for harnessing humans. It is common knowledge that humans have little difficulty in describing the contents of images, although typically they do not find this task particularly engaging. On the other hand, many people would spend a considerable amount of time involved in an activity they consider "fun." Thus, like the ESP Game and Peekaboom, we achieve our goal by working around the problem, and creating a fun game that produces the data we aim to collect.

We therefore introduce Phetch, a game which, as a side effect, generates explanatory sentences for randomly chosen images. As with the ESP Game, we show that if our game is played as much as other popular online games, we can assign captions to all images on the Web in a matter of months. Using the output of the game, we mention how to build a system to improve the accessibility of the Web.

Design of a Useful Game

A traditional algorithm is a series of steps that may be taken to solve a problem. We therefore consider Phetch as a kind of algorithm. Analogous to one, Phetch has well-defined input and output: an arbitrary image from the Web and its proper description, respectively.

* Adapted from [7].

Because it is designed as a game, Phetch needs to be proven *enjoyable*. We do so by showing usage statistics of a one-week trial period. Because it is designed to collect a specific kind of data, Phetch's output needs to be proven both *correct* and *complete*. We prove, through an experiment, that the output of Phetch is both correct and complete.

Related Work: The ESP Game

One possible approach to assigning descriptions to images in order to improve accessibility is the ESP Game. If all images on the Web were labeled by the ESP Game, screen readers could use these keywords to point out the main features of an image. This, however, can be vastly improved. The ESP Game produces keyword labels, not descriptive paragraphs. While keyword labels are perfect for certain applications such as image search, they may be insufficient for describing the contents of a picture. This is demonstrated in Figure 19. We therefore design a different game whose purpose is to collect sentences instead of keyword descriptions. We further show in the evaluation section that the descriptions collected using our game are significantly more expressive than the keywords collected using the ESP Game.

Figure 19. Two inherently different images that share the same labels: "man" and "woman."

Game Mechanics

Phetch is designed as an online game played by 3 to 5 players. Initially, one of the players is chosen at random as the "Describer" while the others are the "Seekers." The Describer is given an image and helps the Seekers find it by giving a textual description of it. Only the describer can see the image, and communication is one-sided: the Describer can broadcast a description to the Seekers, but they cannot communicate back. Given the Describer's paragraph, the Seekers must find the image using an image search engine. The first Seeker to find the image obtains points and becomes the Describer for the next round. The Describer also wins points if the image is found. (See a screenshot of the Seeker's interface in Figure 20.)

Intuitively, by observing the Describer's text, we can collect natural language descriptions of arbitrary images.

Each session of the game lasts five minutes, and the players go through as many images as they can in that amount of time. The Describer can pass, or opt out, on an image if they believe it is too difficult for the game. When deciding to pass, the Describer gets a brand new image and is penalized by losing a small amount of points.

As mentioned, the Seekers have access to an image search engine, which, given a text query (either a word or a set of words), returns all images related to the query. Once a Seeker believes she has found the right image, she clicks on it. The server then tells them whether their guess was correct. The first Seeker to click on the correct image wins and becomes the next describer. To prevent Seekers from clicking on too many images, each wrong guess carries a strong penalty in points. This penalty also ensures that the text given by the Describer is a complete description of the image, since Seekers tend to not guess until they are certain.

The ESP Game-Based Search Engine

The image search engine given to the Seekers is a crucial component of the game. In particular, it must be accurate enough to find a single image given a sufficiently complete description of it. Unfortunately, the majority of popular image search engines (Google Images [26], Yahoo Image Search [68], etc.) are not nearly accurate enough. Although the results they return appear convincing (e.g., many images returned on the query “car” are really cars), attempting to find a specific image (even one that is guaranteed to be indexed) is nearly impossible for two reasons. First, the search space is very large: Google Images, for instance, searches over a billion images [26]. Second, and more importantly, each image indexed by standard image search engines on the Web has a small number of accurate keywords associated with it. Current search engines on the Web index images by using their filenames, their captions and the text around them on the website. Unfortunately, this method attaches very few accurate keywords to most images.

For these reasons, we choose to use a significantly more accurate search engine based on keywords collected by the ESP Game. To remedy the problem of a large search space, 100,000 labeled images were chosen from the ESP dataset as a basis for our search engine. Each of these images has an average of nine keyword labels associated to it. As we show below, using images from ESP has other important advantages.

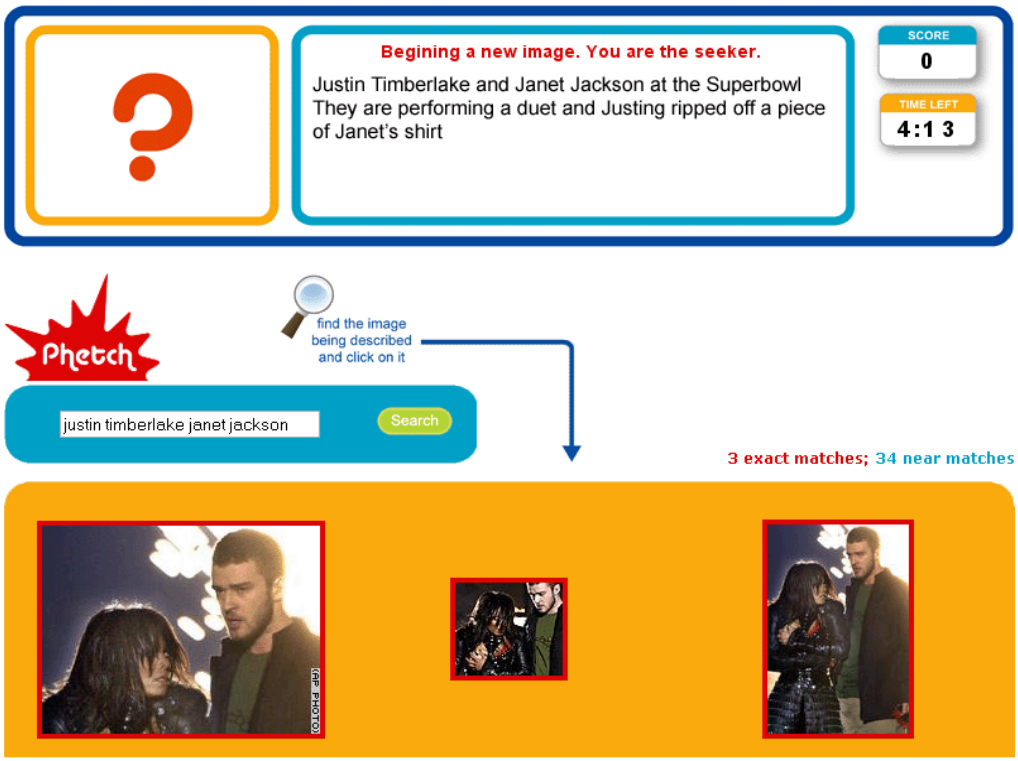

Figure 20. A screenshot of the Seeker's interface.

Emulating Players for Single-Player Games

As stated, Phetch requires three to five players: One Describer and 2-4 Seekers. Given that the total number of players on the website may not always be split perfectly into games of 3-5 players (for instance, when there is only one player), we can also pair up players with "computerized players," or "bots."

Essentially, a "bot" emulates a person by playing pre-recorded game actions. Whenever a real 3-5 person game takes place, we record every piece of text the Describer enters, how long it takes the Seekers to find the image, etc.

Emulating a Describer is easy: simply display text that was previously recorded on a successful session of the game. Emulating Seekers in a convincing manner is more difficult. If a real player enters useless descriptions of the image, we do not want the emulated Seekers to find it. Although this is not a significant problem since most Describers enter accurate descriptions of images, we nevertheless address it to protect the illusion that a real game is being played. The solution to the problem relies on the fact that we are using images from the ESP Game database, so the keywords associated with the images in the search engine are

already known. Therefore, we emulate Seekers by not guessing the right image until a certain number of the ESP keywords have been entered by the Describer.

In addition to allowing every player to quickly start a session of the game, player emulation further acts as a mechanism that helps ensure the accuracy of the descriptions obtained (as we show below).

Ensuring Description Accuracy

The following design strategies help ensure that the descriptions are both *correct* and *complete*:

- **Success of the Seekers.** We use the amount of time taken by the Seekers to find the proper image as an indicator of the quality of the description. If the Seekers do not find the image, we discard the Describer's text.
- **Random pairing of the players.** Players could collude to poison our data. For example, two officemates could play at the same time; one as a Describer entering bogus descriptions and the other as a Seeker that magically finds the images. However, this is unlikely. Phetch is meant to be played online by many players at once. By randomly assigning players to sessions, we ensure that those who wish to collude have a low probability of playing together (and even if they do play together, the mechanism described below guarantees they are unable to poison our data).
- **Description testing.** Most importantly, to determine whether a description is accurate, we use the single-player version of the game. We play back previously entered descriptions to a single-player Seeker. If they can still find the correct image, we get a significant indicator that the description is of high quality: two different people chosen at random were able to single out the image given just this description. Indeed, we can use this strategy more than once: "*N* people chosen at random were able to find the image."

Note that since Phetch game play is time-constrained, the collected descriptions are not guaranteed to be in proper English, and may well be in a chat-like language. This, however, is acceptable for our purposes since we are mainly concerned with improving the current situation.

Verifying Description Accuracy

We now show that captions collected by Phetch are indeed *correct* and *complete* with respect to the images. Moreover, we will compare the natural language descriptions collected to word labels produced by the ESP Game. To this extent, we conducted a study in which participants were assigned to one of two conditions: ESP or PHETCH. Participants in each condition were asked to single out one image among other similar

ones, based on either a natural language description of the image, or a set of word labels from the ESP Game.

Eight participants, between 25 and 38 years old who had never played the game were asked to perform our task. Fifty images were chosen at random from the images already processed by Phetch (these images were annotated by players in a one-week trial period of the game described below). For each of these images, we compiled a list of 29 ESP images that were most closely related to the original one (two images are closely related if they share as many ESP labels as possible).

Participants in the PHETCH condition were given a description of an image taken from Phetch. Participants in the ESP condition were given a list of all the ESP labels related to the image. All the participants were shown a randomly ordered 5x6 grid of 30 images containing the target image plus the set of 29 similar images. They were then asked to choose one image based only on the description or the list of keywords, depending on which condition they were in. This process was repeated for the 50 images in the test set. For each image, we tested the participant's ability to correctly single it out.

For the PHETCH condition, participants were able to single out the correct image 98.5% of the time (with standard error of 1.25%), whereas for the ESP condition, participants were able to select the correct image only 73.5% of the time (standard error = 1.25%). The difference is statistically significant ($t = -14.048$, $p = 0.0002$). This result is important for two reasons. First, it shows that the captions produced by Phetch are indeed correct and complete — had they not been, participants in the PHETCH condition would not have been able to pick the correct images such a large percentage of the time. Second, it shows that captions obtained by Phetch are significantly more descriptive than the keywords obtained from ESP.

Enjoyability

Our game may theoretically output data that is *correct* and *complete*, but if it is not engaging, people would not play, and output would not be produced. Hence it is of major importance to test our claim that Phetch is entertaining.

Although Phetch has not been formally released to the public, we present the results of having test players interact with the game. These people were partly obtained by offering select players from another gaming site (http://www.peekaboom.org) the opportunity to play.

A total of 129 people played the game in this one-week trial, generating 1,436 captions for images on the Web. Each session of the game lasted five minutes and, on average, produced captions for 6.8 images. Therefore, on average, each player spent 24.55 minutes playing the game in this one-week period, and some people played for over two hours straight. These numbers show how enjoyable the game is.

Improving Accessibility

Although it is common knowledge that image captions are a major accessibility problem [45], we were unable to find a reference for the percentage of images throughout the Web that lack a caption. We therefore conducted a study to determine the percentage of images that have an ALT tag (ALT is the caption mechanism in HTML). 2,700 Websites were collected at random (using a large crawl of the Web) in which only 28,029 (24.9%) of the 112,343 total images had an ALT tag at all. (Note that, although this number is indicative of the percentage of captions throughout the Web, other factors such as whether the captions were actually related to the image were not taken into account. A rigorous study of the current state of accessibility of the Web is outside the scope of this thesis.)

We showed that Phetch produces accurate descriptions for images on the Web. But how can such captions be used to improve accessibility? One way is as follows. All captions could be stored in a centralized location. Whenever a visually impaired individual using a screen reader visits a Website, the screen reader could contact the centralized location to ask for all captions associated to the images in the site. The screen reader would then read the caption out loud based on user preferences. Logistically, this mechanism is similar to browser toolbars (e.g. The Google Toolbar [26]) that contact a centralized server to obtain information about the websites that are being visited.

Turning Image Descriptions into Queries

Thus far we have described Phetch as a tool to obtain captions for arbitrary images. Although this is the main application of our game, other pieces of useful data can be obtained by observing the actions of the Seekers. The process of finding the correct image involves turning a plain English description into a sequence of appropriate search queries. By recording the search queries the Seekers enter versus the original description, we can obtain training data for an algorithm that converts natural language descriptions into successful keyword-based queries. Such an algorithm would have important applications in information retrieval (e.g., [12]). In this thesis we do not concern ourselves with this application, but remark that the problem of using such data to train a Natural Language Processing system has been studied before [12].

Chapter Discussion

We have introduced a novel game designed to attach descriptive paragraphs to arbitrary images on the Web. Our game uses the idea of harnessing human computation power in order to solve a problem that computers cannot yet solve (in the same way that the ESP Game or the Open Mind Initiative have proposed). People engage in our game not because they want to do a good deed but because they enjoy it.

We have shown that the descriptions obtained using Phetch are correct and complete. Although Phetch has not been formally released to the public, we hope that in the near future we can collect millions of annotations for arbitrary images on the Web and thus build a system that can improve accessibility. In today's advanced society, driven forward by the immense information exchange throughout the Internet, it can no longer be acceptable that a large portion of the population cannot make full use of this resource with ease.

Chapter 6: Verbosity*

Over the past two decades, there have been several efforts devoted to collecting a large database of "common-sense" knowledge [32,40,47]. This knowledge consists of basic facts that a majority of humans accept as truth, such as "water quenches thirst." The motivation for collecting a large database of true statements is the belief that such knowledge is necessary to create truly intelligent systems. There are also more immediate applications that make such a database useful. For example, a search engine was prototyped that converts the query "my cat is sick" to "veterinarians, Boston, MA" by following a simple chain of reasoning based on an underlying network of simple common-sense facts [33].

Efforts for collecting common-sense facts have demonstrated the promise of this approach. However, they have been unable to collect a large enough fraction of common human knowledge. After 20 years, less than five million facts have been collected — far from the estimated hundreds of millions that are required [47].

This chapter addresses the problem of constructing a truly large database of common-sense facts. Motivated by the ESP Game and Peekaboom, we introduce Verbosity — a fun game with the property that common-sense facts are collected as a side effect of game play. The design of Verbosity guarantees that facts obtained through the game are correct. As with the ESP Game, if our game is played as much as other popular online games, we can collect millions of facts in just a few weeks.

Whereas previous approaches have relied on paid "experts" or unpaid "volunteers" [32,47], we put much stronger emphasis on creating a system that is appealing to a large audience of people, regardless of whether or not they are interested in contributing to Artificial Intelligence. We have *transformed* the activity of entering facts into an enjoyable interactive process taking the form of a game. Although some of the previous approaches have called their systems games to entice people to play, they have not transformed the mode of interaction into that of a real game.

Instead of asking users to "enter a true or false statement," or to rate such statements (both far from enjoyable activities), we start with the realization that a popular party game called Taboo™ already requires the players to state common-sense facts as part of game play. In Taboo™, one of the players gives clues about a certain word to be guessed without saying the word or any of the related words in a list of "taboos." For instance, they might have to describe the word "apple" without saying "apple" and without saying "red," "pie," "fruit," "Macintosh," or "Newton." This player has to give a good enough description of the word to get his or her teammates to guess it (they might say "you can make juice out of this," or

* Adapted from [8].

"they are sometimes green"). The key observation leading to our system is that Taboo™ requires players to say a list of common-sense facts about each word in order to get their teammates to guess it. Verbosity is based on this realization (but differs from Taboo™ in multiple ways).

By playing Verbosity, people help us collect data not because they feel helpful, but because they have fun. Verbosity can be considered a "human algorithm": given a word as input, it outputs a set of common-sense facts related to the word. Instead of using a computer processor, though, this "algorithm" uses ordinary humans interacting with computers throughout the Web. Our system therefore significantly contributes to science in two ways: it collects common-sense data that can be used to improve intelligent applications, and it addresses a typical AI problem with novel HCI tools.

Why Not Use Google?

Computers now have the ability to search vast amounts of data in little time. This means that perhaps we could use a search engine to collect the information we need. Although such an approach yields some useful data [66], it is not good enough for our purposes for three reasons. First, most of the knowledge that we are trying to collect is so obvious that no one has bothered to record it. Second, there exists incorrect knowledge on the Web (for example, at the time of writing this thesis, searching for "cars can swim" returns more results on Google than "cars can't swim"). Third, the text on the Web is unstructured and turning it into a directly useful format is a non-trivial task. Our system does not suffer from these shortcomings.

Basic Game Mechanics

Verbosity is meant to be played online by two players selected at random. One of the players is chosen as the "Narrator" while the other is the "Guesser." The Narrator gets a secret word and must get the Guesser to type that word by sending hints to the Guesser. The hints take the form of sentence templates with blanks to be filled in. The Narrator can fill the blanks in with any word they wish except the secret word (or any prefix of or phrase containing the secret word). See Figure 21. For example, if the word is LAPTOP, the Narrator might say: "it has a KEYBOARD."

Figure 21. Part of the Narrator's screen.

The sentence templates that the narrator can use come in the form of cards. The Narrator starts with 7 cards in their hand, and each card contains one sentence template. To give hints about the secret word, the Narrator chooses a card, fills in the blanks in the card and sends it to the Guesser. Given the hints, the Guesser can type what they think the word is. The Narrator can see all of these guesses, and can tell the Guesser whether each is "hot" or "cold."

By observing the Narrator's hints, we can collect common-sense facts about each word. For instance, when the narrator says "This contains a keyboard" about the word "laptop," we learn that a laptop contains a keyboard.

Players take turns in narrating and guessing. Each session of the game lasts six minutes, and the players go through as many words as they can in that amount of time. The players can agree to pass on a word if they believe it is too difficult.

Sentence Templates

As mentioned before, Verbosity is inspired by the popular game Taboo™. One major difference between Taboo™ and Verbosity, however, is the use of sentence templates. In Taboo™ players can describe the secret word using arbitrary language, while in Verbosity we restrict the Narrator to using only the types of sentences available to them at the time. There are multiple reasons for using sentence templates instead of natural language:

- **Disambiguation.** In English, the same sentence may have multiple meanings. By carefully choosing the templates, we can avoid any problems with multiple meanings.
- **Categorization.** We can use the templates to categorize the types of information we get (e.g. Purpose, Spatial Concepts, Causality). In addition, by providing specific lexical templates, we can control the type of information we receive. This is important in order to make sure that we get a variety of information about a specific word.
- **Parsing.** By using sentence templates, we don't have to worry about parsing natural language sentences, some of which might have poor grammar.
- **Fun.** Requiring the Narrator to use sentence templates to describe the word adds an element of challenge and fun to the game. Instead of being constrained by "taboos," players are constrained by their hint cards.

Our implementation currently uses the following templates:

- ___ *is a kind of* ___. Allows for hierarchical categorization.
- ___ *is used for* ___. Provides information about the purpose of a word.
- ___ *is typically near/in/on* ___ (three templates). Provide spatial data.
- ___ *is the opposite of* ___ / ___ *is related to* ___ (two templates). Provides data about basic relations between words.
- ___. In the game, this is a "wildcard" that collects related words; for example "dance dance" was a clue for the word "revolution."

However, we remark that fresh templates can be easily added to our system. Indeed, adding fresh templates can improve the longevity of the game.

Scoring

The exact number of points given to the players for different actions is not important. However, we note that we neither charge nor give points for using templates. While giving points for each template used would encourage people to provide more facts per word, in user testing we discovered that players will sometimes fill a number of templates randomly to increase their score. Charging people for each template used would guarantee a higher quality of data, since people would try to describe the word using the fewest number of facts possible. However, it is counterintuitive to encourage people to use fewer facts, considering the purpose of the game. Thus, we do not assign any charge or bonus to using facts.

The scoring system is cooperative — points are given to both the Guesser and the Narrator whenever the Guesser enters the correct word. In the current implementation, both players obtain 100 points for a correct guess. Points are not subtracted for passing, nor are incorrect guesses penalized.

The Single-Player Game

As described, Verbosity requires two players. However, it is entirely possible to use the data we have collected to create an automated player, or "bot." This allows for a single-player game in which the player is paired with the bot. There are multiple reasons for doing this. First, and most importantly, we can use the single player version to ensure that the facts we collect are *independent* and *useful*. By *independent*, we mean that the fact does not rely on previous facts mentioned during a particular game session; by *useful*, we simply mean that a Guesser can determine the word based on the facts given. Second, we can pair up players with automated bots when the total number of players is not even. Third, the bot can substitute for a player who leaves the game so that their partner is not interrupted.

Narrator

To emulate a narrator in an automated session, we simply display a subset of facts previously collected about a word. These facts usually come from different game sessions and are displayed in a random order. If the guesser is able to guess the word, we can verify the independence and usefulness of these facts.

More specifically, we assign a score of zero to each new fact collected. The automated player selects a subset of facts that have scores within ±1 of each other — this ensures that we don't artificially increase the score of bad facts by using facts that are known to be of good quality. We replay the facts, and if the word is guessed correctly, we increase the score of each fact by 1. If a fact is used in a certain number of different subsets where the guesser is unable to get the word, we discard it (or, perhaps less harshly, we can flag it for manual review). If a fact attains a high enough score, then we have confirmation of its validity.

Guesser

Emulating a Guesser in a convincing manner is more difficult. If a real player enters useless descriptions of the word, we do not want the emulated Guesser to guess correctly. Although this is not a significant problem because (a) most Narrators enter accurate descriptions of words and (b) we later verify that each fact is correct and useful, we must nevertheless deal with this problem to protect the illusion that a real game is taking place. Once we have collected enough facts, we can use them to aid us in guessing the word. For now, though, we have to rely on approximations. What we do is we compile a list of related words — for example, the word dog could have "bone," "canine," "cat," "terrier," etc. as related words. If enough related words are entered, we guess the correct word.

Designing For Accuracy

As mentioned before, we use the text entered by the Narrator as common-sense knowledge about the word in question. This means we do not want to include false statements. Thus, we employ a set of design strategies to ensure the accuracy of facts entered.

- **Random pairing of the players.** Verbosity is meant to be played online by a large number of people at once. By randomly assigning the players to different sessions of the game, we force players who might want to poison the data to have a low probability of playing together.
- **Success of the Guesser.** We use the time taken by the Guesser to enter the proper word as an indicator of the quality of the Narrator's statements. If the Guesser does not get the word, for instance, we discard the Narrator's text.
- **Description testing.** Most importantly, we use the single player game mentioned above to check the quality of statements entered. We replay a permutation of collected facts for the word, and if the single-player Guesser can still guess the correct word, we have a significant indicator that the facts are useful: multiple different people chosen at random were able to guess the word given these facts.

Implementation

The current version of the game is implemented as a Java applet that connects to a centralized game server. The game can be played at http://www.peekaboom.org/verbosity.

Evaluation

We collected evidence showing that Verbosity is fun to play and that people provide correct common-sense facts while playing. Since the game has not been formally released to the public, we present results from allowing players of another game to get a sneak peak of Verbosity. A total of 267 people played the game for a period of 1 week, generating 7,871 facts. This means that, on average, each player contributed 29.47 facts. In terms of time, each person played for an average of 23.58 minutes, and some played for over 3 hours. We believe these numbers show how enjoyable the game is.

Quality of the Data

We present an analysis of the data collected in the period of time mentioned above. The analysis consisted of selecting 200 facts collected using Verbosity at random and asking the following question about them to six different raters: Is this sentence true? Since Verbosity has not been formally released to the public, the sentences were chosen without having been verified using the automaton. Overall, 85% of the sentences collected were rated as true by all six raters. Something to note is that many of the sentences not rated as

true by all were debatable — for example, “Buddha is a kind of god.” Thus, even without our mechanisms for validating facts, the collected data was extremely accurate.

Related Work

We have shown that our method for providing a fun game is a viable technique for collecting common-sense knowledge. There have been a variety of other methods for collecting common-sense knowledge on the Web, which differ in nature from ours.

Cyc

Cyc (e.g., [32]) was the first effort at building a common-sense database. Cyc started by creating a seed database of common-sense knowledge using paid experts to enter facts in CycL — a proprietary, very precise language developed in order to avoid any ambiguity problems. Using this seed database as a starting point, more data can be collected (e.g., by combining facts to infer new ones).

One problem with this approach is that the number of common-sense facts required is orders of magnitude higher than that which could be entered by an isolated set of experts — over the course of a decade, Cyc has been able to populate its main database with around a million pieces of information [47]. As evidenced by the ESP Game, our approach has the potential to collect millions of pieces of knowledge within the space of several weeks.

Open Mind and Mindpixel

More recently, the Open Mind project has relied on “netizens” (Internet users) to enter common-sense facts. Open Mind has dozens of activities, each designed to gather specific types of knowledge: spatial, hierarchical, implication, etc. So far, they have gathered several hundred thousand pieces of knowledge. Mindpixel [40], like Open Mind, relies on ordinary Internet users in order to build up their database. They have a collaborative system in which many participants create and classify a statement as true or false, thus building up a large database of true/false facts. Validation is a majority-based system, and it rewards those who consistently validate a fact inline with the other Mindpixel users.

The major improvement that we make over Open Mind and Mindpixel is that our system *transforms* the process of entering facts into a true gaming environment. Whereas activities in both Open Mind and Mindpixel require users to simply enter or verify facts, we do not need to explicitly ask users to enter them.

Chapter Discussion

We have presented Verbosity, a game to collect common-sense facts. We have shown data indicating Verbosity is enjoyable and that it produces correct data. Although the game has not been formally released

to the public, we were able to collect a large number of facts from just a few test players over the course of a week. The major contribution we present is the transformation of the tedious process of entering facts in a database into an enjoyable game. By providing an incentive for players, we gain a large quantity of computing power that can be harnessed for multiple applications. Constructing a complete common-sense database would be a monumental achievement, and we believe Verbosity can be tremendously effective in doing so.

Chapter 7: General Method

We have seen many examples of games that solve computational problems. In this Chapter we present a general framework for such "human algorithm games." In addition, we present the following.

- **A method for developing human algorithm games, and a set of design principles.** We propose a general method for designing games to solve open problems. We isolate multiple design principles used in the ESP Game, Peekaboom, Phetch and Verbosity, and show that, in essence, each game achieves its goal by seamlessly integrating player success in the game with success in the computational problem it addresses. That is, the same mechanism that makes the games enjoyable also guarantees that their output is correct.
- **Metrics for evaluating human algorithm games.** In the same way that there are many possible algorithms for every problem, there are also many human algorithm games for every problem. We therefore propose metrics to comparatively evaluate human algorithm games.
- **A list of candidate problems that could be solved using human algorithm games.** The killer application for the paradigm put forth in this thesis is Artificial Intelligence. Although it is possible to design games in which people solve arithmetic problems, doing so would be of little use since computers can do such tasks much faster than humans. Therefore, to maximize utility, we concentrate on problems that humans have no trouble with, but that computers cannot yet solve. We provide a list of problems that are prime for this paradigm.

Related Work

Although previous work has recognized the utility of human cycles and the motivational power of game-like interfaces, none has successfully combined these two concepts into a general method for harnessing human cycles using computer games.

The Open Mind Initiative

As mentioned before, one research effort that has attempted to capitalize on unique human skills is the Open Mind Initiative (e.g., [54]), a worldwide effort to develop "intelligent" software. Open Mind collects information from regular Internet users (referred to as "netizens") and feeds it to machine learning algorithms. Volunteers participate by answering questions that computers cannot answer and teaching concepts to computer programs.

There are two drawbacks to the Open Mind approach: (1) thousands of volunteers that are willing to help must be recruited and (2) people are allowed to enter incorrect facts into their system. We address these limitations by *requiring* our methods to be enjoyable and giving strong probabilistic guarantees that the data collected is free from errors. Open Mind has mostly concentrated on directly asking people to donate their free time in order to provide data (e.g. "please tell us what's in this image"). We propose a *transformative process* in which a problem is turned into a game that is fun and guarantees a correct output.

Making Work Fun

Over the last 30 years, HCI researchers have recognized and written about the importance of enjoyment and fun in a user interface [11,51,62]. Systems such as the StyleCam attempt to use game-like interaction to increase enjoyment and engagement with the software [58]. Many researchers have suggested that the incorporation of game-like aspects into user interfaces could increase user engagement and playfulness of work activities [51, 62].

Some projects have taken this notion even further, turning the user interface into a game. PSDoom provides a first-person-shooter style interface for system administrator tasks [14]. Exercise and fitness is another domain where game-like interaction has been successfully applied to increase user engagement with the task [38].

The idea of turning work tasks into games is often applied in educational domains for children's learning activities [37]. HCI researchers have noted (as we do in this thesis) that it is important not to simply slap a game-like interface onto work activities, but to integrate the activities required for learning into the game itself [14, 30]. There must be a tight interplay between the game interaction and the work to be accomplished.

The ESP Game was the first system where work activity was seamlessly integrated with game-play to solve Artificial Intelligence problems. In the ESP Game, the interaction of trying to guess what your partner is typing for a particular image has the unique side effect of accomplishing useful work by generating word labels for an image. Players are not conscious of the fact that they are doing work while playing, but the labels they generate are accurate and useful.

Although the ESP Game was the first successful integration of game-play and work, we are left with the questions: *can this approach be applied to other tasks? Is it generally useful?* None of the previous literature has concretely defined a process for taking useful work that can only be accomplished by humans and turning it into a computer game. We address this deficiency by formally defining the concept of human algorithm games and presenting a general process for constructing these algorithms.

Games as Algorithms

For clarity, we start with some definitions from [20]:

> "Informally, an ***algorithm*** is any well-defined computational procedure that takes some value, or set of values, as ***input*** and produces some value, or set of values, as ***output***. An algorithm is thus a sequence of computational steps that transform the input into the output.
>
> We can also view an algorithm as a tool for solving a well-specified ***computational problem***. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes a specific computational procedure for achieving that input/output relationship."

From Wikipedia [66], we define a ***game*** as "a recreational activity involving one or more players. This can be defined by (1) a goal that the players try to reach, and (2) some set of rules that determines what the players can or can not do. Games are played for entertainment or enjoyment."

Basic Notation, Definitions, and Examples

We define a ***human algorithm game*** as an algorithm in which the computational steps are carried out by humans playing a ***game***. A human algorithm game is thus comprised of a computational problem (referred to here as P) requiring certain input (referred to here as x) and producing an output (referred to here as y). The computation (referred to here as G; where $G(x) = y$) is performed by humans in a game activity.

The set of computational problems we are most interested in meet the following requirements:

- **Easy for humans.** Since humans will be performing the computation, it is beneficial if the problem (or small parts of the problem) can be easily solved by a human.
- **Hard for computers.** Human cycles are typically much slower and more expensive than computer cycles. Human algorithm games are therefore most usefully applied to computational problems that cannot be solved by a computer.

As stated above, the ESP Game provides an example of a human algorithm game. In this case, the computational problem, P, is that of labeling images with words. So, on input an arbitrary image x, we wish to output a set of keywords y that properly describes the image. This problem meets our general requirements for a human algorithm game to be interesting in that people can perform the computation of labeling images with keywords and computers cannot.

For the ESP Game, the basic sequence of computational steps that takes the input image and outputs a set of keywords is as follows:

> Two partners are randomly assigned from among all the people playing the game. These partners are not allowed to communicate, and the only thing they have in common is an image that they can both see. From the player's perspective, the goal of the game is to guess what their partner is typing. Once both players have typed the same string, they receive points and move on to another image (both players don't have to type the string at the same time, but each must type the same string at some point while the image is on the screen).

Since the players cannot communicate with and don't know anything about each other, the easiest way for both to type the same string is by typing something related to the common image. Notice, however, that the game does not directly tell the players to type a label for the image; all they know is that they have to "think like each other" and type the same string.

An invalid human algorithm game for this problem would be an interface in which the humans are told: "please enter keywords related to the image." Although humans are still doing the computation in this case, such an interface is not enjoyable and therefore is not considered a human algorithm game.

Another example of a human algorithm is Verbosity (Chapter 6), which has not yet been formally released to the public. In this game, the input x is a word and the output y is a set of common-sense facts related to the word. For instance, for the word "milk," the game outputs facts such as "it is white," "people usually eat cereal with it," etc. The human algorithm game, or computational procedure, to address this problem is the following:

> In a two player game, one player attempts to make the other say a target word ("milk" in our example above) without using the word. They do so by saying many facts without using the word itself in their statements (e.g. "it is a white liquid.").

Although the design of Verbosity involves other game elements not stated here, the basic idea is that players need not be asked directly for facts about "milk." The game is designed such that facts are collected as a side effect.

If users were simply asked "please enter true facts about milk" they may or may not do so. However, we do not consider the method of directly asking for contributions to be a valid human algorithm game since the process is not enjoyable for users.

Correctness

As with standard algorithms, the relationship between input and output should be "provably" correct. This means that the output of a human algorithm game should be guaranteed correct within a certain probability even if the players wish otherwise. In the case of the above invalid methods that simply ask people for the

result, the players could easily enter the wrong information if they wished to. True human algorithm games such as the ESP Game, Peekaboom, Verbosity and Phetch, on the other hand, have several mechanisms to provide correctness of the output. These mechanisms ensure that the games will provide a strong probabilistic guarantee of a correct output.

Developing Human Algorithm Games

The value of the human algorithm game concept is that it is generally useful and can be applied to many different computational problems beyond those we have stated. Using previously developed human algorithm games as an example, we have distilled a general process for developing new human algorithm games for computational problems.

First, the problem to be addressed must be clearly defined. Then the human interaction that performs the computation must be identified. Next, output verification must be incorporated into the interaction and design of the system. Finally, principles of game design must be applied to increase the enjoyability of the interaction.

Problem identification

There are many things that computers cannot yet do, but turning one of those challenges into a human algorithm game is non-trivial. The human algorithm games method works best for certain types of problems, so the first step in developing a new human algorithm game is identifying whether the problem in question is a good candidate.

As stated before, the problem should meet two general requirements. First, the computation (on input x, produce output y) must be doable by human beings with normally functioning perceptual capabilities. Second, computers must not be able to perform the $x \rightarrow y$ computation.

Another constraint on the problem is the hard bound on human processing power. A problem that requires trillions of human hours to solve, for instance, is simply infeasible to address using this technique. At the present time, there are six billion humans on earth, roughly one billion of them have Internet access [16] and 430 million of them play computer or video games [70]. Any problem requiring more human-hours than can be contributed by all the people on the Internet, for instance, is probably out of the question.

Finally, since human attention span is limited, it should also be possible to split the problem at hand into "bite-size" pieces. Indeed, to maximize the possibility that people play the game, the problem should be dissectible into sub-problems that can be solved in only a few minutes of human time.

Basic Human Interaction

Once a specific problem has been chosen, the next step is understanding and defining the human interaction for use in the game. Given that on input x we want to produce output y, we can identify the natural interactions by which humans perform this computation.

Take our example of Verbosity. The goal in this game is that given a word (x) we will produce a set of facts associated with that word (y). What is a natural interaction by which people produce facts associated with a word? There exist many different options, and there is no perfect or most correct interaction for any problem. In our example, one possibility is that facts associated with a word are produced when one person is trying to help someone else understand it. For example, someone explaining what milk is might say "milk is a dairy product."

Once identified, this basic type of interaction is then used as a starting point to create a game.

Output Verification

The basic human interaction presents us with output y that should have some relationship with x. It is necessary to verify that relationship.

We propose that the process of verifying the output be tied directly to the game interaction. Indeed, we propose that player success in the game be defined by correctness of the output they provide. Two examples of human algorithm games illustrate this point.

In the ESP Game, players are successful on a given image only when both type the exact same word for that image. Requiring players to type the same word kills two birds with one stone. First, if two players type the same word, we have a way to verify that the word is correct (since it comes from two independent sources). Second, trying to guess the same word as your partner is enjoyable. In this case, the verifiability of the output is directly tied with the enjoyability of the game.

In Verbosity, players are successful on a word only when one player provides enough true facts about the word for the other to guess it. If the facts are incorrect or incomplete, the other player will simply be unable to guess the word. Furthermore, having one player guess the word while the other describes it is precisely what makes the game fun.

The examples above represent the two general ways we propose that verifiability and enjoyment be simultaneously achieved (illustrated in Figure 21):

(1) *Asymmetric verification (e.g., Verbosity, Peekaboom, Phetch)*: In this general game framework, the output y produced by one player (the "Describer") on input x is given to one or more other

players (the "Guessers") for them to produce the input *x*. Success in the game is defined by the ability of the Describer to produce *y* and the ability of the Guesses to produce *x* from *y*.

(2) *Symmetric verification (e.g., the ESP Game):* In this general game framework, the same input *x* is simultaneously given to two or more individuals. Success in the game is defined by the ability of both players to produce the same output *y*.

The beauty of these mechanisms is that they comprise the foundation underlying the game, while simultaneously guaranteeing correctness of the output.

Previous research on game design has identified social interaction as an important criterion for player enjoyment in games [23, 56]. The output verification methods that we present have an inherent component of social interaction because two or more players must be involved in both cases. By making that fact visible to both players, the feeling of connection between them is increased. This social connection is one important factor that contributes to the enjoyability of successful human algorithm games.

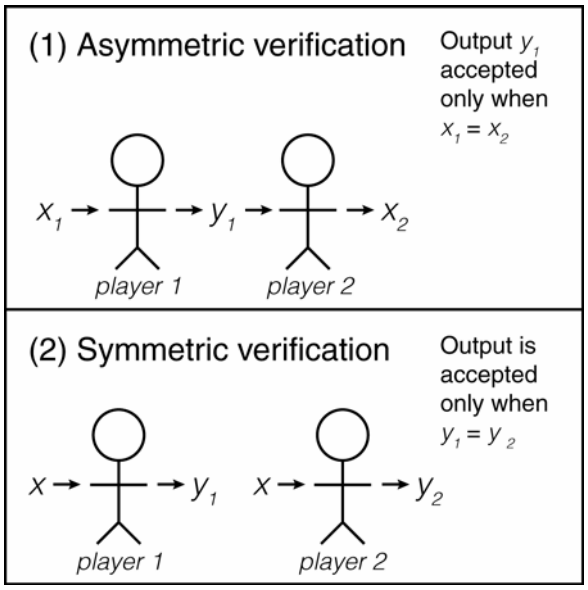

Figure 21. Asymmetric and symmetric verification methods.

There may be additional verification methods useful for creating human algorithm games. We have presented two that have been previously shown to be useful in systems such as the ESP Game, Verbosity, Phetch and Peekaboom, and leave as an open research problem the development of others.

It is important to note that these mechanisms comprise the basic foundation of each human algorithm game, but not the whole. Additional mechanisms must be added to make the game more enjoyable and more strongly guarantee correctness of the output produced.

For example, in the ESP Game (a symmetric verification game) players could circumvent the verification mechanism by agreeing prior to the game that for every image (or input x) they would always type the letter "a" (as their output y). In that case they would always match each other, and incorrect data would be entered into the system.

We present the following generally applicable mechanisms to provide correctness of output and reduce cheating that could lead to output errors:

- **Random Matching.** Human algorithm games that require two or more players, such as the ESP Game, are meant to be played by hundreds, if not thousands, of people at once, most of whom are in distributed locations. If players are randomly paired or grouped, they have no information about who their partner is, and they have no way to previously agree on any cheating strategy. The probability of two or more cheaters using the same strategy being paired together should be low.

- **Player Testing.** At random, games may present players some inputs (x) for which all the possible correct outputs (y) are already known. For these inputs, if the output y' produced by a particular player p' does not match the known correct outputs y, player p' is then considered a cheater and all their results for all inputs are thrown away. Depending on the number of "test" inputs that are presented to the players, this strategy can guarantee that the output is correct with reasonably high probability. To illustrate, assume half of the inputs given to the players are test inputs. Then the probability that a new output by a player is correct given that they have been correct on all the test inputs is at least 50%. (This probability can be amplified by repetition as we will see in the next point.)

- **Repetition.** A game should not consider an $x \rightarrow y$ mapping to be correct until a certain number of players have entered it. This strategy enables any game to guarantee correct output with arbitrarily high probability. If, for a given input x, the game only accepts an output y after n pairs have entered it, and we know that each of those n pairs entered a correct output with at least 50% probability (because of test inputs), then the output is correct with probability at least $(1-\frac{1}{2}^n)$.

General Design Principles

Perhaps the most important aspect of the human algorithm game concept is that the output is produced in an enjoyable manner. As noted with respect to the ESP Game, players in the game are not directly instructed to produce y given x, i.e., "enter keywords for this image" or "describe this image." Instead, they are told to type what their partner is typing. The fact that people enjoy the game makes them want to continue playing, in turn producing more useful output.

There are several general principles of game design that can be applied in creating human algorithm games. The principle we have chosen to focus on is *challenge*. Almost all of the previous work describing game design principles cites "challenge" as a key aspect of any successful game [36, 37, 47, 56]. This can translate into many different things as outlined by [36, 37], including: variable difficulty level, several different levels of goals, hidden information, and uncertain outcomes. Human algorithm game designers can take advantage of almost all of these methods for introducing challenge into a game, in turn encouraging continued play and enjoyment.

Creating a Game Session

Once the basic human interaction and output verification have been identified, they must be translated into a game format. This is accomplished by creating a "game session" consisting of a series of input/output interactions. First, the designer must decide approximately how long a session should last in minutes and seconds. Typical session length for previously developed human algorithm games such as the ESP Game, Verbosity, Phetch and Peekaboom are between three and six minutes. Intuitively, having short game sessions encourages people who want to take a quick break to play the game. Short sessions also mean a low startup cost for initial game play.

Next, the designer must determine how many problem instances can be addressed by the players in this timeframe. This requires preliminary user study evaluation to measure on average how long it takes people to perform one input/output interaction.

Timed Response

Creating time limits for game sessions also introduces challenge into the game in the form of *timed response* [36, 37]. This is accomplished by setting a time limit for each game session. One goal then becomes for players to complete the designated number of problem instances in the time limit. If they accomplish this goal, they are given extra points for their performance. *Timed response* is extremely effective for introducing challenge because it sets forth explicit and obvious goals for the player [36, 37]. We know from literature on motivation that goals that are both well-specified and difficult lead to better task performance than goals which are easy or vague [31]. Thus, it is essential that the number of puzzles for players to complete within a given timeframe be appropriately calibrated to introduce challenge.

Score-Keeping

In the case of human algorithm games, perhaps the most direct method of providing challenge for players is the assignment of points for each instance of successful output produced in the game. For the ESP Game, pairs are given points for each image upon which they successfully agree on a word (which is then turned into a label for that image). This score-keeping provides one level of challenge in the game [36, 37]. A

score summary following each game also provides players with performance feedback [31], allowing progress assessment on score-related goals, such as beating a previous game score, or completing all puzzle instances in the time limit.

Player Skill Levels

Related to score-keeping, an additional method of setting goals is the use of player skill levels (or "ranks"). For example, the ESP Game has five different player skill levels that each player is assigned to based on the amount of points they have accumulated. Every newcomer to the game is assigned the level of "newbie," and then has to earn a certain number of points to advance to the next level.

After each game session, players are presented with information about their current player skill level and the amount of points needed to reach the next level. This information provides them with a specific and challenging goal of reaching the next level, and clear performance feedback on how close they are to achieving that goal [31]. Data from the ESP Game indicates that presentation of this information is very influential on player behavior. Of the 50,000+ players that have obtained an account on the ESP Game, 42% of them have scores that are less than 5,000 points above the rank cutoffs. Given that these intervals cover less than 2% of the space of possible cumulative scores, this suggests that many players continue playing simply to reach a new rank.

High-Score Lists

Another method of providing challenge in human algorithm games is the presence of high-score lists. These lists present the login name and score of players with the top five (or more) highest scores. Presence of an hourly high-score list with the highest points earned in the past hour gives players a specific point level to aim for, and relatively rapid feedback (within the hour) about the achievement of their goal. A daily high-scores list and all-time high-scores list provide score level goals of increasing difficulty. These multi-level goals provide an important motivation for extended game play.

Randomness

By design, human algorithm games incorporate an aspect of randomness. Inputs for a particular game session are selected at random from the set of all possible inputs. Players are also randomly paired to prevent cheating. This randomness serves a very important role in the challenge of human algorithm games.

Because inputs are randomly selected, they are of widely varying difficulty. This variation of difficulty keeps the game interesting and also keeps both expert and novice players engaged [36, 37]. It also means that for every game session there is uncertainty over whether all inputs will be completed within the time limit, adding to the level of challenge experienced [36, 37].

Random assignment of pairs ensures that every game session will be unique. This provides varying levels of difficulty for each game session, and further uncertainty that a session will be completed within the time limit. Anecdotal evidence from the ESP Game suggests that players have a very real sense of the relative skill level of their partner and this affects their joint performance. At the same time, this random pairing means that each game session provides a unique structured social interaction for the players. The feeling of connection players can get from these games is one of the factors that keep them coming back [23, 56]. Players of the ESP Game that know they are matched with a simulated player enjoy the game less.

Pre-Recorded Games

Thus far we have presented general techniques for designing human algorithm games that rely on *multiple players*. Using multiple players makes the games more enjoyable and, at the same time, allows players to check each other's computation. However, multi-player games present some logistical problems. For instance, if the game requires three players, there may be times when some of the players cannot play because the total number of people in the game is not divisible by three. Even more seriously, when a game is just gaining popularity, it is difficult to guarantee that multiple people will be trying to play at the same time. To deal with these issues, we propose a technique introduced by the ESP Game: using pre-recorded game-play. The idea is that any of these games can be transformed into single player games by pairing a single player with a pre-recorded set of actions. (Care is taken to avoid pairing a player with a recorded game by the same player.)

In the case of a symmetric verification game such as the ESP Game, automated players are relatively easy to implement. Whenever two people are playing, simply record every action they make and when they made it. Whenever a single player wishes to play, pair them up with a pre-recorded set of moves.

In the case of asymmetric games, the problem is more complex. Verbosity, Peekaboom, and Phetch (all asymmetric verification games) have each implemented a single-player version using various techniques that are customized to the particular game.

Evaluation

Once a human algorithm game is created for a problem, how can it be evaluated? Given two different human algorithm games that solve the same problem, which one should be chosen? We introduce a set of metrics for evaluating a human algorithm game.

Efficiency and Expected Contribution

If we treat games as algorithms, a natural metric of evaluation is efficiency. There are many possible algorithms for every problem, some more efficient than others. Similarly, there are many possible human

algorithm games for a given problem. In order to choose the best solution to a problem we need a way to compare the alternatives.

Efficiency of standard algorithms is measured by counting atomic steps. For instance, QuickSort is said to run in *O(n log n)* time on average, meaning that it sorts a list of *n* elements in roughly *n log n* computational steps. In the case of human algorithm games, the notion of a computational step is not as clear. Therefore, we define efficiency through other means.

First, we define the ***throughput*** of a human algorithm game as the average number of problem instances solved, or input/output mappings performed, per human-hour. For example, the throughput of the ESP Game with no repetitions is roughly 233 labels per human-hour. This is calculated by examining how many individual inputs, or images, are matched with outputs, or labels, over a certain period of time for the game.

The notion of throughput as we define it is roughly analogous to the notion of "asymptotic efficiency" in standard algorithms, which measures how fast the number of computational steps grows as the size of the input grows.

Learning curves present a slight complication to the calculation of throughput. Most games involve a certain form of learning, and as a byproduct players become more experienced and more skilled at the game. For games where speed is important, this learning can result in faster game play. In the ESP Game, for instance, the throughput can be up to two times higher depending on the expertise of the players. To deal with such a complication, we define throughput as the *average* number of problem instances solved per human-hour. This average is taken over all game sessions throughout a reasonably lengthy period of time.

Games with higher throughput should be preferred over those with lower throughput. But throughput is not the end of the story. Because a human algorithm game is a *game*, fun must also be taken into account. It does not matter how high the potential throughput is for a game if nobody wants to play it. The real measure of utility for a human algorithm game is therefore a combination of throughput and "enjoyability."

Enjoyability is difficult to quantify and depends heavily on the precise implementation and design of the game. As we previously described, even trivial modifications to the user interface or the scoring system affect how enjoyable a game is. Our approach to quantifying this measure is to calculate the average overall amount of time the game is played by a single player. For instance, in the case of the ESP Game, on average, each player will play the game a total of 91 minutes throughout time.

Although this approach doesn't capture aspects such as "popularity" or contagion (e.g., how much players tell their friends to play), it is a fairly stable measure and indicative of the usefulness of a game. Previous work in the usability tradition on measuring how fun or enjoyable a game is has suggested the use of self-

report questionnaire measures [27, 47]. However, a behavioral measure such as the one we propose provides a more accurate assessment of how much people play the game and, in turn, how useful the game is for computational purposes.

We therefore define for a human algorithm game:

- ***Throughput*** = average number of problem instances solved per human hour.
- ***ALP (average lifetime play)*** = average (across all people who play the game) overall amount of time the game will be played by an individual player.
- ***Expected Contribution*** = Throughput * ALP.

Expected contribution is our summary measure of human algorithm game quality. It indicates the average number of problem instances that a single human will solve throughout time by playing a particular game. We propose this measure as a generally useful metric of evaluation for human algorithm games.

Analysis of the Output

Once a human algorithm game has been designed and tested, it is important to verify that the design is indeed correct — that is, that the output of the game is correct. One way to do so is to perform an analysis of the output using human volunteers. This can involve two techniques: (1) comparing the output produced in the game to outputs generated by volunteers instead of game players, and (2) having independent "raters" rate the quality of the output produced in the game.

Can a Computer Do This?

When evaluating a human algorithm game, it is important to ask the question: "can a computer play this game successfully?" If the answer is yes, then the game is of little or no utility because the computer could simply play the game to solve the computational problem. If the answer is no, then the game has truly captured a human ability and the time invested by the players may provide a useful benefit.

In fact, there is a strong relationship between human algorithm games and CAPTCHAs.

A good human algorithm game is in a sense a CAPTCHA: if the entity can play the game successfully, it must be human; otherwise it is a computer. The ESP Game, for instance, could be used to differentiate humans from computers simply by assuming that any entity that obtains a low score or no score in the game is a computer.

On the flip side, human algorithm games can also be used to break CAPTCHAs, since they provide an incentive for humans to perform a computation that computers cannot yet carry out. For example, the ESP

Game can be trivially used to break distorted-text CAPTCHAs: simply pass the images of distorted-text through the game and people will label them with the correct letters contained in the image.

Conclusion and Open Problems

We have presented a general paradigm for harnessing human computation power to solve problems that computers cannot yet solve. Some open problems that could be solved using this technique include:

- **Language Translation.** Imagine a game in which two players that do not speak the same language work together to translate text from one language to the other.
- **Monitoring of Security Cameras.** With cameras becoming less expensive over time, it is now feasible to have security cameras everywhere. Imagine a game in which people watch the security cameras and alert authorities of illegal activity.
- **Improving Web Search.** Different people have different levels of skill at searching for information on the Web. Imagine a game in which the players perform searches for other people.
- **Text Summarization.** Imagine a game in which people summarize important documents for the rest of the world. (The biggest challenge in solving this problem is that it would require an intelligent way to break it up into small “bite-size” chunks.)

We believe there is an immense amount of work to be done in the continued development of this paradigm. Indeed, we hope researchers will use and improve upon the method and metrics presented for development and evaluation of human algorithm games. We believe the techniques in this thesis present an opportunity for researchers and game designers to contribute to the progress of Artificial Intelligence.

References

1. AltaVista Company. Altavista Search Home. http://www.altavista.com/
2. von Ahn, L., and Dabbish, L. Labeling Images with a Computer Game. *In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI),* 2004, pp. 319-326.
3. von Ahn, L., Blum, M., Langford, J. Telling Humans and Computers Apart: How Lazy Cryptographers do AI. *Communications of the ACM,* February 2004, pp. 56-60.
4. von Ahn, L., Blum, M., Hopper, N., and Langford, J. CAPTCHA: Telling Humans and Computers Apart. In *Advances in Cryptology, Eurocrypt '03*, volume 2656 of *Lecture Notes in Computer Science*, pages 294-311, 2003.
5. von Ahn, L., Blum, M., Hopper, N., and Langford, J.. The CAPTCHA Web Page: *http://www.captcha.net.* 2000-2003.
6. von Ahn, L. Ruoran, L., and Blum, M. Peekaboom: A Game for Locating Objects in Images.
7. von Ahn, L., Ginosar, S., Kedia, M., Ruoran, L., and Blum, M. Improving Accessibility of the Web with a Computer Game.
8. von Ahn, L., Kedia, M., and Blum, M. Verbosity: A Game to Collect Common-sense Facts.
9. Barnard, K., Duygulu, P., and Forsyth, D. A. Clustering Art. *IEEE conference on Computer Vision and Pattern Recognition*, 2001, pages 434-441.
10. Barnard, K., and Forsyth, D. A. Learning the Semantics of Words and Pictures. *International Conference of Computer Vision*, 2001, pages 408-415.
11. Bongard, M. M. *Pattern Recognition.* Spartan Books, Rochelle Park NJ, 1970.
12. Brill, E., Dumais, S., and Banko, M. An Analysis of the AskMSR Question-Answering System. *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* 2002.
13. Carroll, J.M., & Thomas, J.M. Fun. *ACM SIGCHI Bulletin, 19(3)*, 1988, pp. 21-24.
14. Carson, C., and Ogle, V. E. Storage and Retrieval of Feature Data for a Very Large Online Image Collection. *IEEE Computer Society Bulletin of the Technical Committee on Data Engineering*, 1996, Vol. 19 No. 4.
15. Chao, D. Computer games as interfaces. *Interactions of the ACM, 11(5)*, 2004, pp. 71-72.

16. Coates, A. L., Baird, H. S., and Fateman, R. J. Pessimal Print: A Reverse Turing Test. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR' 01)*, pages 1154-1159. Seattle WA, 2001.
17. Columbia University Center for Telecommunications Research. *webSEEK.* www.ctr.columbia.edu/webseek/
18. Computer Industry Almanac: http://www.c-i-a.com/.
19. Corbis. *Stock Photography and Pictures*. http://www.corbis.com/
20. Cormen, T., Leiserson, C, and Rivest, R. Introduction to Algorithms (or, "the big white book"). The MIT Press, 1990.
21. Duygulu, P., Barnard, K., de Freitas, N., and Forsyth, D. A. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. *7th European Conference on Computer Vision,* 2002, pages 97-112.
22. Electronic Arts. *Pogo.com*. http://www.pogo.com
23. Federoff, M. (2002). Heuristics and usability guidelines for the creating and evaluation of fun in video games. *Unpublished thesis, Indiana University, Bloomington.* http://www.melissafederoff.com/thesis.html. Online Sept. 1, 2005.
24. Fleck, M. M., Forsyth, D. A., and Bregler, C. Finding Naked People. *ECCV*, 1996, pages 593-602.
25. Flickr Photo Sharing Service. http://www.flickr.com.
26. Google Inc. *Google Web Search*. http://www.google.com
27. Hassenzahl, M., Beu, A., & Burmeister, M. Engineering Joy. *IEEE Software, Jan/Feb*, 2001, pp. 70-76.
28. HotorNot: http://www.hotornot.com
29. Kanade, T., and Uchihashi, S. User-Powered "Content-Free" Approach to Image Retrieval. Proceedings of International Symposium on Digital Libraries and Knowledge Communities in Networked Information Society 2004 (DLKC04).
30. Laurel, B.K., Interface as mimesis. In, D.A. Norman & S.W. Draper (Eds.), *User centered system design: New perspectives on human-computer interaction*, 1986, pp. 67-85. Hillsdale, NJ: Erlbaum.
31. Lempel, R. and Soffer, A. PicASHOW: Pictorial Authority Search by Hyperlinks on the Web. *WWW10*.

32. Lenat, D. B. CYC: A Large-Scale Investment in Knowledge Infrastructure. *Commununications of the ACM 38(11)*, 1995, pages 32-28.
33. Lieberman, H., Liu, H., Singh, P., and Barry, B. Beating common sense into interactive applications. *AI Magazine*, Winter 2004, 25(4), pages 63-76.
34. Lillibridge, M. D., Abadi, M., Bharat, K., and Broder, A. Method for selectively restricting access to computer systems. US Patent 6,195,698. Applied April 1998 and Approved February 2001.
35. Locke, E. A., & Latham, G. P. *A theory of goal setting and task performance*, 1990, Englewood Cliffs, NJ: Prentice Hall.
36. Malone, T.M. Heuristics for designing enjoyable user interfaces: lessons from computer games. *In Proceedings of the 1982 Conference on Human Factors in computing systems,* 1982, pp. 63-68.
37. Malone, T.M. What makes things fun to learn? Heuristics for designing instructional computer games. *Proceedings of the 3rd ACM SIGSMALL Symposium*, 1980, pp. 162-169.
38. Microsoft Corporation. *MSN Gaming Zone.* http://zone.msn.com
39. Milliman, R. E. *Website Accessibility And The Private Sector.* www.rit.edu/~easi/itd/itdv08n2/milliman.html
40. Mindpixel Project: http://www.mindpixel.org
41. Mokka, S., Väätänen, A., Heinilä, J. , & Välkkynen, P. Fitness computer game with a bodily user interface, *Proceedings of the second international conference on Entertainment computing*, 2003, pp.1-3.
42. Mori, G. and Malik, J. Recognizing Objects in Adversarial Clutter — Breaking a Visual CAPTCHA. In *Conference on Computer Vision and Pattern Recognition*, June 2003.
43. N2H2, Inc. N2H2 Filter. http://www.n2h2.com
44. Naor, M. Verification of a human in the loop or Identification via the Turing Test. Unpublished Manuscript, 1997. Available electronically: *http://www.wisdom.weizmann.ac.il/\~naor/PAPERS/human.ps*
45. National Organization on Disability Website. http://www.nod.org/
46. O'Connor, B. and O'Connor, M. Categories, Photographs & Predicaments: Exploratory Research on Representing Pictures for Access. *Bulletin of the American Society for Information Science* 25.6, 1999, page 17-20.
47. The Open Mind Common Sense Project. http://www.kurzweilai.net/articles/art0371.html

48. Pagulayan, R., Keeker, K., Wixon, D., Romero, R., and Fuller, T. User-centered design in games. In *The Human-Computer Interaction Handbook: Fundamentals, Evolving Techniques and Emerging Applications*. J. A. Jacko and A. Sears (eds.), 2003, Lawrence Erlbaum Associates, Mahwah, NJ, 883-905.

49. Pinkas, B. and Sander, T. Securing Passwords Against Dictionary Attacks. In *Proceedings of the ACM Computer and Security Conference (CCS' 02)*, pages 161-170. ACM Press, November 2002.

50. Random Bounce Me Website. http://random.bounceme.net/

51. Russell, B.C., Torralba, A. Murphy, K.P. and Freeman, W.T. LabelMe: a database and web-based tool for image annotation. *MIT AI Lab Memo* AIM-2005-025, September, 2005.

52. Schneiderman, B. Designing for fun: how can we design user interfaces to be more fun? *Interactions*, *11(5)*, 2004, pp. 48-50.

53. Scheniderman, H. and Kanade, T. Object Detection Using the Statistics of Parts. *International Journal of Computer Vision,* 2002.

54. Stork, D. G. and Lam C. P. Open Mind Animals: Ensuring the quality of data openly contributed over the World Wide Web. *AAAI Workshop on Learning with Imbalanced Data Sets*, 2000, pages 4-9.

55. Stork, D. G. The Open Mind Initiative*. IEEE Intelligent Systems & Their Applications, 14-3*, 1999, pages 19-20.

56. Sweetser, P., & Wyeth, P. GameFlow: A model for evaluating player enjoyment in games. *ACM Computers and Entertainment*, *(3)3*, 2005, ACM Press.

57. Trask, R. L. *Language Change*. Routledge, 1994

58. Torralba, A., Murphy, K. P. and Freeman, W. T Instructions, Taboo. http://www.hasbro.com/common/instruct/Taboo(2000).PDF

59. The MIT CSAIL Database of objects and scenes. http://web.mit.edu/torralba/www/database.html

60. Torralba, A. An example of the importance of context in vision. http://web.mit.edu/torralba/www/gallery/gallery3.html

61. Tsang, M., Fitzmaurice, G., Kurtenbach, G., & Khan, A. Game-like navigation and responsiveness in non-game applications. *Communications of the ACM, (46)7*, 2003, pp. 57-61.

62. Turing, A. M. Computing Machinery and Intelligence. In: *Mind, Vol. 59, No. 236, pp. 433-460*. 1950.

63. Uchihashi, S., and Kanade, T. Content-free Image Retrieval by Combinations of Keywords and User Feedbacks. In Proceedings of International Conference on Image and Video Retrieval 2005 (CIVR), pages 650-659.

64. Watchfire Corporation Website. http://www.watchfire.com/

65. Webster, J. Making computer tasks at work more playful: Implications for systems analysts and designers. *In Proceedings of the ACM SIGCPR conference on Management of information systems personnel*, 1988, pp. 78-87.

66. Wetware: Gathering Common Sense. http://wetware.hjalli.com/000016.html.

67. Wikipedia: http://www.wikipedia.org.

68. Yahoo Image Search. http://www.yahoo.com

69. Yahoo!, Inc. *Yahoo! Games*. http://games.yahoo.com

70. Zona Incorporated. Online Console Gaming, 2004.http://www.researchandmarkets.com/reports/225173.