Towards a Theory of AI Completeness

Dafna Shahaf and Eyal Amir
Computer Science Department
University of Illinois, Urbana-Champaign
Urbana, IL 61801, USA
{dshahaf2,eyal}@uiuc.edu

Abstract

In this paper we present a novel classification of computational problems. Motivated by a theoretical investigation of Artificial Intelligence (AI), we present (1) a complexity model for computational problems that includes a human in the process, and (2) a classification of prototypical problems treated in the AI literature. The contribution of this paper is useful for automatically distinguishing between human and computer users. Also, this work serves as a formal basis for investigation of problems that researchers treat as hard AI problems. Most importantly, this work allows progress in AI as a field to be more measurable, instead of measurable with respect to problem-specific quantities.

1 Introduction

Many problems that appear easy for humans are difficult for computers. Difficulty for computers is either in computational terms (time or space required) or in having no known solution at all. For example, we know how to solve planning, but the solution is intractable in general; other problems, such as vision and natural language understanding, we do not know how to solve with a computer.

For this reason, some recent systems harness the computational power of *people*: Interactive Evolutionary Computation (Takagi 2001) uses human judgment to evaluate solutions, thus allowing us to address problems with no formal model developed, like beauty or creativity. The Cypher-mint.com PayCash system identifies its clients by taking their picture and sending it to a human worker. Online games recently developed (von Ahn 2006) use people to label images over the web, label parts of images, and generate commonsense facts.

Other systems, like CAPTCHA (von Ahn *et al.* 2003), utilize the same idea to tell humans and computers apart. The user proves he is human by answering a question that is hard for computers and not for people. This kind of system is used successfully to block spammer bots.

Current computational models help analyzing and classifying computational problems from the perspective of time and space taken by a computer. However, many problems of interest to researchers in artificial intelligence escape such analysis, and are hard to classify or investigate formally. Furthermore, current Computer Science theory does not address problems from the perspective of difficulty for computers or difficulty for AI researchers to solve.

This paper presents the foundations of a theory of computation that can help analyze problems of interest to AI researchers. In particular, we present a new computational model, *Human-Assisted Turing Machine (HTM)*. In this model, a Turing machine (TM) can access a human oracle – a black-box machine that can decide problems that humans solve. We are inspired by the introduction of oracle machines to the TM framework, and its effect on Computational Complexity theory. By reasoning about worlds in which certain actions are free, oracles enabled a deeper understanding of the relationship between complexity classes. Similarly, we hope that reasoning about worlds in which humans can help Turing Machines can lead to new insights about AI problems.

We present several alternative formal definitions of human oracles, varying in their computational abilities, error rates, and utilities. We also define complexity measures, which are used to analyze complexity of problems in our model. Those measures examine ways in which computational workload can be split between the computer and the human oracle: our intuition is that a problem is hard if we cannot solve it without having the human perform a substantial part of the computation. Those measures can also tell us how much of the problem we already managed to delegate to a computer; using reductions, progress in one problem can translate to progress in another.

We analyze several AI problems in this framework, and show how the choice of a human oracle model and complexity measures affects the algorithm design and complexity. Our analysis of traditional AI problem shows that Optical Character Recognition (OCR) of printed text is easy, linear classification requires only poly-log help from a human (even if he makes mistakes), and the Turing Test is only quadratically harder when the human oracle is not fixed throughout the test. We also observe that commonsense planning is at least as hard as the Turing Test and Image Labeling.

The main contribution of this paper is formalizing the complexity of problems that involve parts we do not know how to solve or formalize, and for which we need humans.

We believe that a complexity theory for AI can give a deeper understanding of the structure of problems. It can provide insights about the sources of difficulty, and direct the search for efficient solutions. This investigation can also lead to a set of problems that one can use to distinguish computers from humans; this will provide a precise mathematical definition of problems that one could have proposed instead of the Turing Test.

2 Human-Assisted Turing Machines

We are interested in problems that are easy for humans and hard for computers. We can help computers solve such problems by allowing them to interact with humans. In this section we examine ways in which computation can be split between humans and computers. Our new model enables such split and contributes to a theory of (AI-)hardness of problems.

Definition 2.1 (Human-Assisted Turing Machine) *Let H be an oracle machine, representing a human (an oracle is a black box which is able to decide certain problems (Turing 1938); this is an abstract machine used in complexity theory and computability theory). A formal definition of this machine will follow in Section 3.1. A Human-Assisted Turing Machine (HTM) M^H is a Turing machine (TM) with access to the oracle H.*

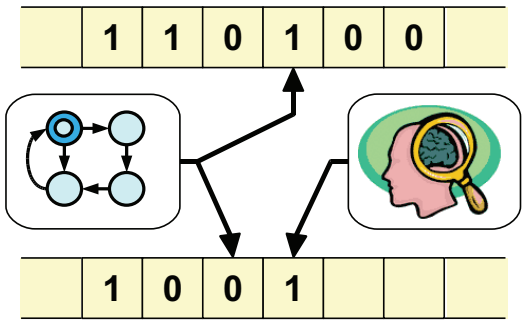

Figure 1: Human-Assisted Turing Machine

The Turing machine has two tapes, one for its computation and another for the human oracle. It can write on the second tape an input for the oracle, then tell the oracle to execute. The oracle computes a function, erases its input, and writes its output to the tape (see Figure 1). Oracles are usually assumed to perform their computation in a single step, but some of our models may assume differently.

Before we define complexity formally, let us start with an example. Consider the following problem: we are given n randomly drawn samples that we need to classify. We know the classifiers are simple threshold functions, $h_w(x) = 1$ if $x > w$, and 0 otherwise (with w depending on the input). Assume we do not know w, but a human can classify the data correctly.

Figure 2 shows two algorithms in the HTM framework. *Classify1* simply gives a person all data to classify. Assuming the person takes constant time to answer, this results in an $O(n)$-time algorithm, and the person sees all n samples.

PROCEDURE *Classify1*(H, S)
H a human, S a vector of samples
1: **for** i= 1 to $|S|$ **do**
2: lbl:= H**.GetLabel**$(S(i))$
3: *Label*$(S(i)$,lbl)
4: **return** labeled S

PROCEDURE *Classify2*(H, S)
H a human, S a vector of samples
1: *HeapSort*(S) //Sort Samples
2: left:= 1 , right:= $|S|$, mid := floor((left+right)/2)
3: lblLeft:= H**.GetLabel**$(S(0))$
4: **while** left $<$ right **do** //Binary Search
5: mid := floor((left+right)/2)
6: lbl := H**.GetLabel**$(S(mid))$
7: **if** lbl == lblLeft **then** left := mid+1 **else** right := mid-1
8: *Label*$(S$(mid),lbl)
9: *Label*$(S$(1 ... mid-1),lblLeft)
10: *Label*$(S$(mid+1 ... $|S|$),¬lblLeft)
11: **return** labeled S

Figure 2: Classification Algorithms with human intervention.

Classify2 is based on the *Active Learning* paradigm: note that if we lay these points down on the line, their hidden labels are a sequence of 0's followed by a sequence of 1's. Our goal is to discover the point w at which the transition occurs. *Classify2* sorts the samples and performs a simple binary search, which asks the human for just $\log n$ labels. This algorithm takes $O(n \log n)$ time, but relies much less on the human. Assuming people are more expensive than computers, this is an improvement.

It seems there is a trade-off between the workload of the human and that of the computer. In the following sections, we develop methods to decide which algorithm is better.

2.1 Complexity in the HTM Model

A complexity measure (based on Blum's definition (Blum 1967)) is a function ϕ mapping human-assisted Turing machines M^H and w input to non-negative integers. It has to satisfy the following two axioms:

1. $\phi(M^H, w)$ is defined if and only if M^H terminates on w
2. Given M^H, w, k, it is decidable whether $\phi(M^H, w) = k$.

Time and space are two examples of complexity measures for computations. For notational convenience, we define $\phi(M^H)$ as a function of input w

$$\phi(M^H)(w) := \phi(M^H, w)$$

A complexity model is a tuple $\langle \mathbb{M}^{\mathbb{H}}, \Phi_H, \Phi_M, \prec \rangle$. $\mathbb{M}^{\mathbb{H}}$ is a set of HTMs. Φ_H is a vector of complexity measures, representing the complexity of the human oracle (that is, it depends only on the oracle tape). Similarly, Φ_M is a vector of complexity measures, representing the complexity of the Turing machine. The vectors can be of any positive length.

$\prec$ is an ordering over $\Phi_H \times \Phi_M$, human-machine complexity pairs.

Definition 2.2 (Algorithm Complexity) *The complexity of executing an algorithm with machine M^H is a pair $\langle \Phi_H(M^H), \Phi_M(M^H) \rangle$. The first element represents the complexity of the human's part of the work, and the second – of the machine's part.*

In other words, the pair represents the way the work is split between the person and the computer. For example, the complexity measures can be the usual time-space measures; several others are presented in Section 3.2.

Definition 2.3 (Problem Complexity) *A problem L is of complexity C if there is an algorithm of complexity C that decides it. We are mostly interested in the minimal complexity (according to $\prec$)*

$$\min\{\langle \Phi_H(M^H), \Phi_M(M^H) \rangle \mid M^H \text{ decides } L\}$$

For partial $\prec$, this is the set of minimal pairs.

This set represents the trade-off between humans and computers in solving the problem. Using different ϕ functions and $\prec$ relations, we can define many interesting complexity measures.

For example, let Φ_H, Φ_M be time complexities, and $\langle \Phi_H, \Phi_M \rangle \prec \langle \Phi'_H, \Phi'_M \rangle$ iff $\Phi_H + \Phi_M < \Phi'_H + \Phi'_M$. This gives the minimum total time to decide L.

Using different $\Phi's$, $\prec$ we can define the *humanness* of an algorithm, that is how much of the computational burden it takes from the human. A possible definition would be the minimal amount of help from a human that allows a computer to solve the problem easily (poly-time).

Example 2.4 Let us take a closer look at the classifying algorithms of Figure 2. Let Φ_H, Φ_M be the time complexities of the human and the machine, respectively. We are interested in the minimal amount of human-time needed to allow poly-time for the TM (this always exists, since the TM can simply give the human all input). Formally, $\prec$ is defined as

$$\langle \Phi_H, \Phi_M \rangle \prec \langle \Phi'_H, \Phi'_M \rangle \text{ iff } \Phi_M \text{ is poly-time and } \Phi'_M \text{ is not, or both are poly-time and } \Phi_H < \Phi'_H.$$

In *Classify1*, the person sees all of the input, and takes $O(n)$ time to label it. In *Classify2*, a Turing machine can do $O(n \log n)$ of the work (sorting the samples and querying the person). The person will see only $\log n$ of the data, and will need only $O(\log n)$ time to label it.

The algorithms' complexities are $\langle O(n), O(n) \rangle$ and $\langle O(\log n), O(n \log n) \rangle$, respectively. The classifying *problem* has complexity

$$\min\{\langle O(n), O(n) \rangle, \langle O(\log n), O(n \log n) \rangle\} = \langle O(\log n), O(n \log n) \rangle$$

This can later be compared to other problems.

Note that if, for example, we allow a universal TM and intelligence is mechanizable (humans are at most polynomially faster than TMs) then the hierarchy collapses, as human work can be taken over with only a polynomial increase in the cost. For this reason, it may be interesting to restrict the kind of TMs that can be used in our models.

3 A Closer Look into the Model

In previous sections, we gave the high-level description of the HTM framework. However, no formal definition of the human oracles was given. In this section we present several alternative human models (3.1), along with some useful complexity measures (3.2); we later show how the choice of a model affects the complexity and the algorithm design.

3.1 Alternative Human Models

Oracle A human is an oracle machine that can decide a set of languages L_i in constant time:

$$H \subseteq \{L_i \mid L_i \subseteq \Sigma^*\}.$$

Oracle With Time Complexity A human is a machine that can decide a set of languages L_i. Unlike the oracle model, answering a query might take more than a constant time.

$$H \subseteq \{\langle L_i, f_i \rangle \mid L_i \subseteq \Sigma^*, f_i : \mathbb{N} \to \mathbb{N}\}.$$

f_i is the time-complexity function for language L_i: the human can decide if $x \in L_i$ in $f_i(|x|)$ time. Alternatively, we can define f_i on the input itself, not its length.

Probabilistic Model So far we have assumed an idealized human, always giving the correct answer. A more realistic model relaxes this assumption: a human is modeled as

$$H \subseteq \{\langle L_i, p_i \rangle \mid L_i \subseteq \Sigma^*, 0 \leq p_i \leq 1\}.$$

That is, a human can function as an oracle to language L_i, but each of its answers is true only with probability p. We assume the answers are independent; this is like asking a different person each question.

Utility Some human tasks require considerably more skills than others. For example, translating from French to Navajo can be done by very few, and perhaps requires a joint effort of several people. The previous models did not capture this notion.

When dealing with real humans, the question of utility arises. Humans need to benefit from participating in the algorithm, for example by being paid. One can also make the algorithm enjoyable for humans, e.g. a game (von Ahn 2006). In this model, a human is a machine that can decide a set of languages L_i. .

$$H \subseteq \{\langle L_i, u_i \rangle \mid L_i \subseteq \Sigma^*, u_i : \mathbb{N} \to \mathbb{N}\}.$$

u_i is the utility function for language L_i: the human can decide if $x \in L_i$, and he requires $u_i(|x|)$ utility. This can be used to design algorithms with a limited budget.

Persistence The previous models assumed independence between the different answers of H. In practice, however, sometimes the same person answers all the queries. This is similar to the notions of Persistent Oracles and Interactive Proof Systems.

This model has some drawbacks. For example, algorithms for the probabilistic case may become useless (e.g. the classifying algorithm in Section 5); repeating a query multiple times cannot change the answer. However, we can sometimes take advantage of it: when trying to handle a conversation (Turing Test) with the help of a single

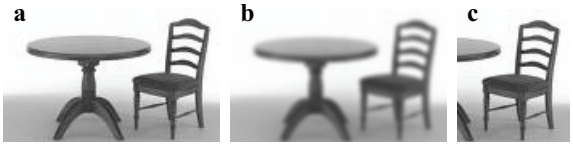

Figure 3: Finding a Chair

person, we do not need to send him the conversation history with every query.

Combinations of those models can also be used. E.g., we can have a probabilistic model with utilities.

3.2 Measures of Human Effort

Complexity measures for TMs are well-studied. We proceed to propose a few interesting measures for the human part of the computation; those will later help us measure the humanness of algorithms. Note that we usually assume random access (also for the TM).

Human Time, Space, Utility The most natural measure is probably the time complexity. In the Oracle model, this is equivalent to the number of calls; in the time-complexity model, this takes into account the human's complexity functions, f_i. Other measure might involve utilities, and perhaps even space.

Human Input Size Estimating the size of the input the person handles; the intuition behind this measure is that if a person needs to read almost all of the original input, the algorithm probably does not take the burden from him. In the oracle model, this is the total size of queries.

Information The previous measure suffers from some limitations. E.g., the TM could zip the input to the person (so he unzips and answers); this results in a smaller human-input size, but it is nevertheless not an improvement. In order to avoid those considerations, we propose an alternative information-theoretic definition, information gain: *IG(I|HI)* is the average number of bits the person needs in order to know the original input *I* when given input *HI*.

To see the difference between the last two measures, consider the vision problem of finding a chair. This is a function problem: the input is a picture, and the user needs to find a chair in it (see Figure 3a). An algorithm B reduces the resolution of the image (3b) and queries a person, while another algorithm, C, queries the user with a small region where it believes a chair is (3c). In this example, images 3b and 3c are of exactly the same size. Therefore, the human-input measure would consider the algorithms equivalent. However, our intuition is that the algorithm C is much better, and saves the person a lot of work. The Information measure captures this: the person can reconstruct most of 3a from 3b, and not from 3c.

Note that when dealing with humans, our measures do not have to make complexity-theory assumptions (e.g., neglecting constant factors, concentrating on worst-case behavior).

3.3 In Practice

So far we assumed that a model of H is given to us, including the languages that humans can decide, the probability of success, and utilities. We now discuss ways to obtain this model in practice.

Empirical Evaluation A simple method is to collect statistics for any language L that humans can decide.

Canonical Actions (Brown & Hellerstein 2004) define a set of canonical actions that users can take (such as selecting a feature or entering a variable) and build a complexity model for each of them. The resulting model is used to evaluate any algorithm that interacts with humans through those actions. We can take a similar approach, defining a set of building-block actions and evaluating them empirically. The difficulty here is defining those actions.

Individual Tests an interesting question arises when the problem requires a skill level that varies considerably among people, such as translation. In this case, perhaps a general model of humans is not enough; instead, we can test every candidate. Only after we establish some estimate of his abilities can we use his answers to the real queries. Note that this requires algorithms that do not know in advance the exact human model, but rather need to be able to use the humans that are available to them.

When evaluating a person, one must also take into consideration malicious users. Those users can score high on their tests and provide good answers for a long time, only to trick the system into believing their answers and later manipulate it. (Gentry, Ramzan, & Stubblebine 2005) show that for this reason (among others), a simple majority vote can be better than Bayesian inference in the presence of adversaries.

4 Extensions and Complexity Classes

So far, we discussed mainly *Deterministic Machines*. Many extensions can be made, and complexity classes can be defined. We denote by $C_{\mathcal{H}}$ a complexity class in the HTM framework that is similar to class *C* in the TM framework. Complements are defined as usual: $coC_{\mathcal{H}}$ is the class of problems whose complement is in $C_{\mathcal{H}}$.

Non-Deterministic Machines A non-deterministic HTM accepts an input if an accepting state is reachable. For example, $NP_{\mathcal{H}}$ – decision problems which can be verified by a deterministic HTM in poly-time. We believe the commonsense-planning decision problem is in $NP_{\mathcal{H}}$ – a person might not be able to solve it in poly-time (in the input and solution length), but he can verify a solution.

Interactive Proof Systems In those systems, an all-powerful prover interacts with a verifier. Messages are sent between them until the verifier is convinced that he knows the answer to the problem; both the cases of a human prover and a human verifier are interesting. The Turing Test can be formalized as an interactive proof system (see (Bradford & Wollowski 1995; Shieber 2006), although this is another framework).

Parallel and Distributed Computation Using two or more machines and humans to accomplish a common

task is one of the most interesting directions, and is the basis to many human-in-the-loop systems nowadays, such as Wikipedia and Collaborative Spam Filtering. It is also useful in coping with human mistakes.

This direction is related to many important fields, such as decision theory, game theory and cryptography. It raises interesting questions – e.g., it might be non-trivial to break the problem down, and to combine the solutions later. Problems like natural-text generation are difficult in this sense: a text generated in pieces in not likely to be very coherent. Even sorting becomes complicated to distribute when the criteria is subjective, e.g. aesthetic.

Furthermore, since many of those systems are online services, and the users are often unknown, the issue of trust becomes crucial. See (Gentry, Ramzan, & Stubblebine 2005) for more details.

Function Problems Unlike decision problems, which have a Yes/No answer, Function Problems can have complex answers. This seems to be natural to many problems that humans can solve. $FP_{\mathcal{H}}$ – function problems that can be solved by a deterministic HTM in poly-time. Note that while function problems are equivalent to decision problems in terms of computability, they do not necessarily have the same complexity.

Many other paradigms can be extended to our framework, e.g. Randomized, Approximation, and Online Computation (the same way Genetic algorithms are extended to Interactive Genetic Algorithms). Note that many relations between complexity classes continue to hold in our framework, e.g.

$$P_{\mathcal{H}} \subseteq NP_{\mathcal{H}},\ coNP_{\mathcal{H}}$$
$$NP_{\mathcal{H}},\ BPP_{\mathcal{H}} \subseteq MA_{\mathcal{H}} \subseteq AM_{\mathcal{H}} \text{ (Arthur-Merlin protocols)}$$

4.1 Reductions

For decision problems, A_1 is R-reducible to A_2 means that there is a deterministic reduction algorithm in complexity class R, which transforms instances $a_1 \in A_1$ into instances $a_2 \in A_2$, such that the answer to a_2 is YES iff the answer to a_1 is YES. A more general notion, applying also to function problems, requires an oracle machine that computes the characteristic function of A_1 when run with oracle A_2.

For example, the problem of participating in a Turing Test (pretending to be human and convincing the judge) can be reduced to the problem of judging a Turing Test (telling if the other party is a computer). The problem of Natural Language Understanding (NLU) can be reduced to Speech Understanding using text-to-speech tools (or to Vision's Scene Understanding, via text-to-image tools). Since text-to-speech is not considered very complicated, we say that Speech Understanding is at least as hard as NLU.

Formally, if problem L can be reduced to problem L' using R of complexity $\langle \Phi_H, \Phi_M \rangle$, R's output is of size $m = f(n)$ and L' of complexity $\langle \Phi'_H, \Phi'_M \rangle$, then

L is of complexity $\langle \Phi_H(n) + \Phi'_H(m), \Phi_M(n) + \Phi'_M(m) \rangle$. The definition of reduction should prove helpful to define notions of completeness.

5 Examples: Analyzing AI Problems

In this section we analyze several problems using the HTM framework (see Figure 4). We use the complexity measures and $\prec$ defined in Example 2.4.

Classifying Refer again to the classification problem in Section 2. We know the classifiers are simple thresholding functions, but not the threshold. We are given n samples that a human can classify correctly. The algorithms in Figure 2 are for the Oracle model, and take time $\langle O(n), O(n) \rangle$ and $\langle O(\log n), O(n \log n) \rangle$, respectively (Example 2.4).

Switching to the time-complexity model, assume that the person needs $O(m^2)$ time to classify a sample of size m, and we are given n such samples (total input size nm). The algorithms then take $\langle O(nm^2), O(nm) \rangle$, $\langle O(m^2 \cdot \log(n)), O(mn \log n) \rangle$ time, respectively. The human's time complexity is just the number of queries times $O(m^2)$. The TM needs to sort the input, query the oracle with $\log n$ samples, each of size m, and then label each sample.

What about the probabilistic model? Assume that the person can classify the instances with probability p, and we require reliability level $0 < r < 1$. The algorithms for the previous models do not guarantee us this reliability in general. However, for any $p \neq$½ and a fixed r we can solve the problem with only $O(\log^2(n))$ queries to the human.

PROOF SKETCH: The idea is to ask only $\log n$ questions, but to repeat each one of them until we are sure of the answer with at least $r^{1/\log n}$ reliability. Let E_m be the number of errors in the first m answers; this is a binomial random variable. Wlg $p >$½ (otherwise negate the answers). Applying Chebyshev inequality,

$\forall \epsilon > 0.\ P(|\frac{E_m}{m} - (1-p)| < \epsilon) \geq 1 - \frac{p(1-p)}{(m\epsilon^2)}$

Let $0 < \epsilon < p -$½:

$P(\frac{E_m}{m} <$½$) \geq P(\frac{E_m}{m} - (1-p) < \epsilon) \geq$

$P(|\frac{E_m}{m} - (1-p)| < \epsilon) \geq 1 - \frac{p(1-p)}{(m\epsilon^2)} = 1 - O(\frac{1}{m})$

Therefore, we can use majority vote for m big enough. To ensure reliability $r^{1/\log n}$, we need m such that $[1 - O(\frac{1}{m})]^{\log n} > r$. Since $[1 - \frac{1}{a \log n}]^{\log n} \rightarrow e^{-1/a}$, there is a constant b such that $b \log n$ times suffice.

OCR Optical character recognition (OCR) is computer software designed to translate images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text. Several new approaches in OCR are cooperative approaches, where computers assist humans and vice-versa. Computer image processing techniques can assist humans in reading extremely difficult texts.

Assume that we receive an image of a printed text. We also assume that the page(s) are all in the same font and orientation. A human can look at the first sentence, and supply valuable information to the TM: the orientation of the page, and the letters in that sentence. The TM should proceed with the OCR process, based on this knowledge: whenever it encounters a new symbol, it can ask the person for help. Since the number of symbols is small, the person should take $O(1)$ time to answer them all. The complexity of the problem is likely to be $\langle O(1), poly(n) \rangle$. It seems like this problem is easier, from the AI perspective, than the previous one.

Problem	Complexity
OCR, Printed Text	Oracle Model: $\langle O(1), poly(n)\rangle$
Turing Test	Oracle Model: $\langle O(n), O(n^2)\rangle$, Linear Read Time: $\langle O(n^2), O(n^2)\rangle$, Persistent: $\langle O(n), O(n)\rangle$
Classifying (n samples)	Oracle: $\langle O(n), O(n)\rangle$, $\langle O(\log n), O(n \log n)\rangle$, Error $p \neq$½: $\langle O(\log^2 n), O(n \log n)\rangle$ Quadratic Oracle, sample size m: $\langle O(nm^2), O(nm)\rangle$, $\langle O(m^2 \cdot \log(n)), O(mn \log n)\rangle$
Image Labeling	$O(n, n)$. Arthur-Merlin game for labeling image parts.

Figure 4: Analyzing AI Algorithms in the HTM Framework – Sample Results

Turing Test In the Persistent Oracle model, we can simply toss each question to the human, and return his reply: an n-sentence conversation has complexity $\langle O(n), O(n)\rangle$ (n queries). If the oracle is not persistent, however, we have to send the person the whole conversation history every time, resulting in $\langle O(n), O(n^2)\rangle$. If the person takes linear time to read the query, we get $\langle O(n^2), O(n^2)\rangle$.

As noted earlier, the Turing Test can also be formalized as an interactive proof system (and as an instance of common-sense planning, we suspect). We furthermore believe that this is one of the hardest AI problems, in a sense that everything can be reduced to it: for any problem humans solve easily, we can create an $O(1)$-description of the question, and send it along with the input to an Extended-Turing-Test machine (non-textual input); the reply should be correct.

The ESP Game The ESP Game (von Ahn 2006) is an online game, using people to determine the contents of images by providing meaningful labels for them. The game is played by two randomly chosen partners, who cannot communicate. They both see an image; from the player's perspective, the goal is to guess what the partner is typing for the image (with no a-priori constraints).

This can be interpreted as a simple majority-vote classification algorithm, with humans as experts. More formally, each game is a Turing machine with two persistent Human oracles, who do not have access to each other's tapes. The Turing machine then queries them, by writing an image on their input tapes, and receives back their answers. Image labeling has more than one correct answer, so we apply majority voting to each of their possible answers: since $k = 2$ (number of experts), this means taking only answers they both agreed on. For n images (assuming a fixed maximal image size and number of answers), we get $\langle O(n), O(n)\rangle$.

Peekaboom The game of Peekaboom (von Ahn 2006) is slightly more complicated. Again, this is an online game for two players. One of the players is given an image and a label, which describes a part of the image (e.g., a hat that somebody wears). The player can then send the other player a part of the original image, hoping he would recognize the label. From the second player's point of view, this is similar to the ESP Game: he is given an image, and needs to provide a label that agrees with the label of the other player.

This can be thought of as a Arthur-Merlin game (an Interactive Proof protocol), where Arthur is a polynomial machine with access to a human. Arthur sends Merlin the query "is there a part in image I that can be labeled with l"? If Merlin answers "no", we know he is lying (assuming we believe the labeling source that provided l). Otherwise, he sends back a part of the picture as a proof. Arthur then uses a human to see if this part really is labeled with l.

6 Conclusions

In this paper we proposed a new model of computation, which includes a human in the loop. This model leads to a novel classification of computational problems, based on the way the work can be split between a human and a computer – that is, how much of the computational burden is delegated to the computer; this classification is suited for a theoretical investigation of AI.

We discussed several models of a human, differing in their time-complexity, error and utility models. We also suggested several interesting complexity measures. We then analyzed AI-problems with those tools, and showed how the chosen model affected the algorithm design and overall complexity.

The contribution of this paper is formalizing the complexity of problems that involve parts we do not know how to solve or formalize. This work will serve as a formal basis for investigation of the hardness of AI problems; it can also be useful to generate tests that automatically distinguish between humans and computers.

References

Blum, M. 1967. A machine-independent theory of the complexity of recursive functions. *J. ACM* 14(2):322–336.

Bradford, P. G., and Wollowski, M. 1995. A formalization of the turing test. *SIGART Bulletin* 6(4):3–10.

Brown, A. B., and Hellerstein, J. L. 2004. An approach to benchmarking configuration complexity. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop: beyond the PC*, 18. NY, USA.

Gentry, C.; Ramzan, Z.; and Stubblebine, S. 2005. Secure distributed human computation. In *EC '05: Proceedings of the 6th ACM conference on Electronic commerce*. NY, USA: ACM Press.

Shieber, S. M. 2006. Does the Turing Test demonstrate intelligence or not? In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

Takagi, H. 2001. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE* 89(9):1275–1296.

Turing, A. M. 1938. *Systems of logic based on ordinals: a dissertation*. Ph.D. dissertation, Cambridge University, Cambridge, UK. Published by Hodgson Son, London, UK.

von Ahn, L.; Blum, M.; Hopper, N.; and Langford, J. 2003. Captcha: Using hard AI problems for security. In *Proceedings of Eurocrypt*, 294–311.

von Ahn, L. 2006. Games with a purpose. *Computer* 39(6).