Towards Human Interactive Proofs in the Text-Domain

Using the Problem of Sense-Ambiguity for Security

Richard Bergmair[1] and Stefan Katzenbeisser[2]

[1] Austrian Partner of the University of Derby
A-4060 Leonding, rbergmair@acm.org
[2] Technische Universität München
Institut für Informatik
D-85748 Garching, skatzenbeisser@acm.org

Abstract. We outline the linguistic problem of word-sense ambiguity and demonstrate its relevance to current computer security applications in the context of Human Interactive Proofs (HIPs). Such proofs enable a machine to automatically determine whether it is interacting with another machine or a human. HIPs were recently proposed to fight abuse of web services, denial-of-service attacks and spam. We describe the construction of an HIP that relies solely on natural language and draws its security from the problem of word-sense ambiguity, i.e., the linguistic phenomenon that a word can have different meanings dependent on the context it is used in.

Key words: HIP, CAPTCHA, text, natural language, linguistic, lexical, word-sense ambiguity, learning

1 Introduction

As networking technology is making progress and society finds ever more efficient ways of sending information around the globe, our attitude towards the way we make information available is rapidly changing; this change is mainly taking us into one direction: automation. If we want to make information or services available on the net, then we actually use a computer that offers services to other computers. When we gather information from the net, then we have a computer that obtains this information for us from other computers. It is important to recognize that the act of sending information to or receiving information from the net is usually automated. This phenomenon lies at the heart of many highly security-critical situations. Although automation drives the progress of the IT industry, there are situations where too much automation is undesirable. For example, consider the following scenarios:

– *Abuse of web-services:* While free e-mail services should be available to subscribers, web bots that subscribe to thousands of accounts in order to send spam are obviously unwanted guests. In a similar scenario, online polls should only accept votes from humans, not from web bots aiming at manipulating the poll by stuffing the ballot box with thousands of invalid votes [20].

- *Fighting spam and worms:* We want to accept e-mails from our friends and colleagues, but not from automated mass-mailers distributing spam. For this purpose, it would be highly desirable to have a device that can decide whether an e-mail originated from a human or from an automated bot (the company *www.spamarrest.com* is already marketing this idea) [12, 20].
- *Privacy issues:* Personal data can be made available to humans without high risk if the recipients are trusted. However, we want to prevent sensitive data from being replicated, indexed, or otherwise analyzed at a large scale, for example by web-spiders of search-engines or web-advertisers.
- *Denial-Of-Service:* Online services have to process requests issued by legitimate human users, but, in order to prevent denial-of-service attacks, they need to refuse processing requests made by malicious scripts for the sole purpose of putting load on the system [10].
- *Dictionary Attacks:* Computer systems could try to process only login attempts by human users, but refuse to process login attempts originating from a bot running a dictionary attack [13].
- *Protecting Business Interests:* Online shops may want to offer their services only to humans, but not to bots for comparative shopping [12].

The core component that would be needed to protect against any of the aforementioned security threats is a device allowing us to distinguish between humans and computers in communication systems. This problem is investigated in the context of Human Interactive Proofs (HIPs) [19]. The computer security community only recently discovered the power of HIPs for solving practical security problems.

To our knowledge, the earliest appearance of HIPs in the literature is an unpublished manuscript by Moni Naor [12], dating back to 1997. He suggested the inclusion of a test into a communication system that can easily be passed by humans, but that cannot be passed by computers, similar in setup to Turing's test [18]. Turing's test evaluates whether or not a machine shows intelligent behavior by having an interrogator communicate to the machine and to another human. In his test-setup both the computer and the human are hidden from the interrogator, interacting with him through the same interface. If the interrogator is unable to distinguish between the computer and the human when talking to them over the interface, then we could in fact attribute intelligent behavior to the machine. On the other hand, if we agree that the machine is not "intelligent" enough (i.e., there are some questions of the interrogator that can be solved by a human but not by the machine), we can devise a test that distinguishes the human from the machine—in essence, this amounts to an HIP.

In January 2002, the *First Workshop on Human Interactive Proofs* was held at Xerox PARC [22]. One year later, von Ahn et. al. [20] proposed the notion of the "Completely Automated Public Turing Test to Tell Computers and Humans Apart" (CAPTCHA). While Turing's original version of the test requires a human interrogator to judge the answers received from the testee, the "completely automated" Turing test substitutes the human interrogator for a computer program, which is able to automatically generate and grade the tests passed on to and received from the testee. Furthermore, adhering to Kerckhoffs' principle, this computer program is assumed to be public knowledge.

Von Ahn et. al. [20] implemented, and investigated in further detail, image-based CAPTCHAs. Generally, their systems rely on the difficulty of recognizing the semantic content of images. For example, their OCR-based CAPTCHA, called *GIMPY*, produces images by typesetting words from a dictionary in the generation phase. *BONGO* draws geometric shapes, and *PIX* employs a public database of images labeled by humans with an identification of their content. In any case, the output of this phase is an image, and a symbolic representation of the semantics that would be assigned to it by a human. The image is then distorted in such a way that humans would still be able to recognize the image content, but state-of-the-art algorithms known to the AI-community would not. Although all programs and databases used by these systems are assumed public knowledge, the distortion depends on a private source of randomness. During the testing phase, the testee is asked to identify the content of the image (e.g., read a word typeset by *GIMPY*); the syntactic representation of this content is then checked against the representation of the intended semantics produced in the generation phase. Other treatments of OCR-based CAPTCHAs can be found in [3, 17].

The security of the system depends heavily on the assumption that a computer program cannot "pass" the test. This assumption is usually justified by the lack of appropriate AI algorithms for this purpose. However, this does not mean that such algorithms cannot exist in general. For example, Malik and Mori [11] have developed a program that can recognize *GIMPY* images with a success rate of 92%. This shows that a constant evaluation of the security of existing HIPs as well as the construction of various new HIPs is an important topic. In particular, it is important to have CAPTCHAs operating on many different kinds of media, for example ones that can be used by the visually impaired. CAPTCHAs concentrating on the sound-domain, for example, were investigated by [1, 2].

The medium of interest in this paper will be natural language text. We discuss the major problems that arise when constructing an HIP in the text domain in Sections 2 and 3. Constructions for HIPs are proposed in Sections 4 and 5. We will show that it is indeed possible to construct an HIP based on natural language texts. Finally, future research directions are outlined in Section 6.

2 HIPs in the Text-Domain

The idea of HIPs operating in a text-domain is near at hand, since computational linguistics is one of the most prominent research disciplines in artificial intelligence. The construction of an HIP (or CAPTCHA) in the text-domain is often cited as an important open problem [20, 21, 14, 8]. To our knowledge, the only attempt to create text-based CAPTCHAs was made by Godfrey [8].

In his first approach, he randomly selected a word from a piece of text taken from a datasource of human-written text, and substituted it by another word selected at random, in the hope that it would be easy for humans to pick that word (because it didn't fit in the context), but difficult for computers. However he could write a program that had considerable success-rates in "cheating" the test by taking into account statistic characteristics of natural language.

In a second approach, he used a statistic model to generate essentially meaningless text in order to get a "bad" sentence, and selected a "good" sentence from a repository of human-written text. The idea was that, after applying random transformations to both, a human should still be able to distinguish between the good and the bad sentence, which turned out not to be the case.

Godfrey concluded his contribution with a discussion of why text was so much more problematic a medium for the construction of CAPTCHAs than images or sounds. He attributed this to the fact that humans are more often exposed to distorted images and sounds than to distorted text, and, as a result, it is more difficult for humans to recognize distorted text. However, we believe that the problem is actually due to the types of text-manipulations studied so far.

A linguistic HIP cannot work if, on one hand, we rely on a human's ability to assign meaning to a meaningful text as the very distinction between humans and computers, and, on the other, we carry out semantically significant distortions, thereby presenting meaningless text to the testees. It is not surprising that a human has no true advantage over a computer in handling text, where semantic content has been destroyed, e.g. by randomly replacing or shuffling words or otherwise manipulating the text by statistical models that treat words as meaningless symbolic black-boxes.

In this paper we will present a kind of text-manipulation that uses a model of natural language semantics to ensure that semantic content is preserved thoughout all distortions applied to a text. In particular, we will use lexical semantics to construct an HIP that draws its security from the problem of *word-sense ambiguity*, i.e., the phenomenon that a single word can have different meanings and that different words can have the same meaning, depending on the context in which a word is used. This issue has been thoroughly studied both in computational and theoretical linguistics. Computer Science has been facing this problem, ever since the first computers were to be used for natural language translation in the early 1950s, but it has remained a major gap between humans' and computers' ability to understand natural language.

3 The Problem of Sense-Ambiguity

The problem of word-sense ambiguity is closely linked to the notion of synonymy, defined by Miller et. al. [9] in the following way:

> "According to one definition (usually attributed to Leibniz) two expressions are synonymous if the substitution of one for the other never changes the truth value of a sentence in which the substitution is made.

	move	*impress*	*strike*	*motion*	*movement*	*work*	*go*	*run*	*test*
s_1	1	1	1	0	0	0	0	0	0
s_2	1	0	0	1	1	0	0	0	0
s_3	1	0	0	0	0	0	1	1	0
s_4	0	0	0	0	0	1	1	1	0
s_5	0	0	0	0	0	0	0	1	1
...									

Fig. 1. A lexical matrix.

> By that definition, true synonyms are rare, if they exist at all. A weakened version of this definition would make synonymy relative to a context: two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value."

Roughly, a *linguistic context* can be seen as the surrounding of a word in the text in which it appears, helping to determine the word's meaning. We can use this definition of synonymy to organize words into synonymy-sets or *synsets* for short. A synset contains words that can *sometimes* (i.e., in a linguistic context) be substituted for each other. Using this approach, we can infer the meanings of words from their organization into synsets (such models for meaning are called differential theories of semantics). The association between a word and its meaning is referred to as *word-sense.*

For example, *move*, in a sense where it can be replaced by *run* or *go*, has a different meaning than *move*, in a sense where it can be replaced by *impress* or *strike.* If we wanted our dictionary to model semantics explicitly, we would have to formulate statements of the form "use *move* interchangeably with *run* or *go if you want to express that* something goes through a process" or "use *move* interchangeably with *impress* or *strike if you want to express that* something has an emotional impact on you". However, in differential approaches to semantics, as adopted in this paper, we do not represent meaning explicitly, because representing the "*if you want to express that...*"-part of the above phrases is very difficult, if not impossible. All we do is to formulate statements of the form "there exists *one* sense for *move*, in which it can be interchanged by *run* or *go*" and "there exists *another* sense for *move*, in which it can be interchanged by *impress* or *strike*". This yields two synsets {*move*, *run*, *go*} and {*move*, *impress*, *strike*}.

Miller et. al. [9] used the *lexical matrix* to demonstrate the relation between words and their senses; an example is given in Figure 1. If we wanted to look up the meaning of a word, say *run*, we would get multiple senses s_3, s_4, and s_5. This ambiguity is called *polysemy.* Inversely, if we want to look up the word for a specific meaning we have "in mind", say s_2, we would get multiple words: *move*, *motion* and *movement.* This ambiguity is called *synonymy.* Alternatively, we can represent the phenomenon by a VENN diagram in the space of word-meanings (see Figure 2), displaying words as sets of meanings they express. Sense-ambiguity is the phenomenon that these sets overlap.

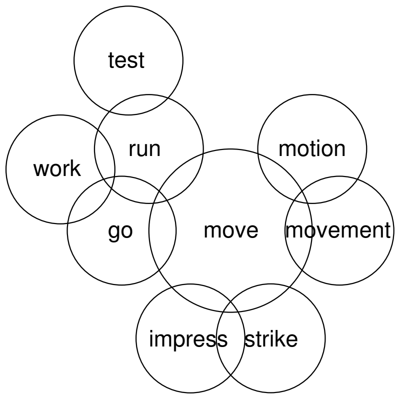
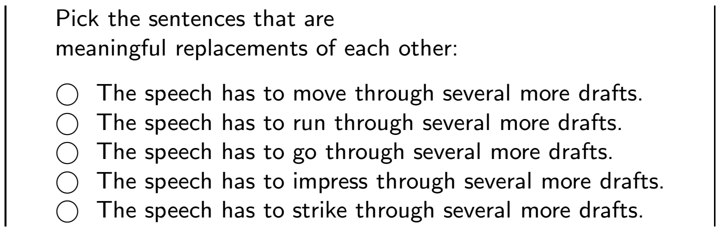


Fig. 2. Ambiguity of words in the space of meanings.

The problem of sense-ambiguity in natural language texts can be used in an HIP to tell computers and humans apart. As expressed by Miller et. al. in the paragraph quoted earlier, we have to think of word-sense as being resolved by a linguistic context, i.e., by clues inherent to the actual use of a word in a sentence or paragraph. An interesting property of linguistic context is that humans seem to have no problem in using it to resolve sense ambiguity, whereas computers have constantly struggled with the issue.

In order to clarify the role of context, we turn back to our example. In a sentence like *Today's sermon really moved me*, the word *move* can be replaced by *impress* or *strike*. When we use the same word in a sentence like *The speech has to move through several more drafts*, then we use it in a linguistic context, where it can be replaced by *go* or *run*. The sentences *Today's sermon really ran me* or *The speech has to impress through several more drafts* are clearly incorrect. However it will be very difficult for a computer—using differential models for semantics—to judge whether these sentences are correct, as the word *move* appears in the two synsets $\{move, run, go\}$ and $\{move, impress, strike\}$. Unless a computer is able to solve the problem of sense-ambiguity, it cannot decide which synset contains the correct substitutions of the word *move*. Therefore the test shown in Figure 3 will be very difficult for a computer, but trivial for a human native speaker of the language.

A human can tell from the context which of the words are correct replacements, and which of the words are not. Making a computer do so is studied in the context of Word-Sense Disambiguation (WSD). Current approaches to WSD are nowhere near completely solving the problem of lexical ambiguity. State-of-the-art WSD systems operate at a precision of up to 65% [16], whereas human annotators agree about word-sense in 90% of the cases [15, 4]. We believe that the gap of 25% is large enough to construct a reliable HIP if we repeat the test several times independently in order to reduce the error probability. In addition, the performance of 65% is achieved only for test-scenarios that capture typical


Pick the sentences that are
meaningful replacements of each other:

○ The speech has to move through several more drafts.
○ The speech has to run through several more drafts.
○ The speech has to go through several more drafts.
○ The speech has to impress through several more drafts.
○ The speech has to strike through several more drafts.

Fig. 3. A task that is difficult for a computer but trivial for a human.

text in everyday written-language. By selecting word-senses that are especially difficult to disambiguate, eventually checking whether a test can be passed by any of the well-known techniques before presenting it to the testee, we can make work much harder for WSD systems.

4 Constructing HIPs

For the construction of an HIP based on sense-ambiguity, we need a *corpus* C, which is a collection of correct natural language sentences. Furthermore we need a *lexicon* consisting of the set of words W and the organization of these words into synsets $S_1, \dots, S_n$ with $S_i \subseteq W$ for $1 \leq i \leq n$. Denote the set of all synsets by $S = \{S_1, \dots, S_n\} \subseteq 2^W$. Both the corpus and the lexicon can safely be assumed public wisdom. However we need to rely on a private database establishing a mapping $sa : C \times W \mapsto S$, which we will refer to as the *secret annotation.*

The Public Lexicon: The set W is a table containing words, represented by character-coded strings. The set S is stored as a table associating words and symbolic tokens in such a way that each group of words assigned to the same symbolic token is a semantic equivalence-class relative to some linguistic context (similar to the intuitive picture of synsets presented in the previous section).

The Public Corpus: Each sentence $c \in C$ contains at least one word w_c that is contained in at least two distinct synsets $S_a, S_b \in S$, so that $w_c \in S_a \cap S_b$. Thus, looking up the word w_c in the table representing S yields multiple symbolic tokens, indicating that w_c has different meanings.

The Secret Annotation: If two synsets S_a and S_b contain word w_c, the annotation sa resolves this conflict from the linguistic context. For example, if S_a contains the correct replacements for w_c in the sentence c, we have $sa(c, w_c) = S_a$. This mapping will have to be established by a table, or by tagging the sentences in C via control-symbols, but in any case it needs to be based on data that (initially) has been entered by humans. The security of the scheme presented herein relies on the fact that no computer will be able to compute the annotation using state-of-the-art WSD algorithms with low error probability.

A database containing such knowledge could be initialized from available word banks, such as WordNet. Princeton's WordNet [7, 9] is a lexicon containing 139 786 English words organized into 111 222 synsets. 15 892 of these words are useful for our purposes, because they are contained in multiple synsets, each of which contains multiple words[1]. WordNets for most other important languages are readily available or currently under construction. If they were not public, the sample sentences contained in the WordNet glossaries or the SemCor [5, 6] corpus could be used to initialize the corpus.

We can now characterize the set $R(c)$ which contains correct replacements of a sentence c, and the set $Q(c)$ containing sentences that cannot be distinguished from correct replacements of a sentence c without solving the problem of sense-ambiguity. Let the function $subst(c, w_c, w_s)$ denote the result of substituting the word w_c in the sentence c for another word[2] w_s. We write $w_c \in c$ if w_c occurs in c. Formally, we define $Q(c)$ and $R(c)$ for any $c \in C$ as follows:

$$\begin{aligned} Q(c) &= \{subst(c, w_c, w_s) \mid w_c \in c, S_c \in S, w_c \in S_c, w_s \in S_c\}, \\ R(c) &= \{subst(c, w_c, w_s) \mid w_c \in c, S_c = sa(c, w_c), w_c \in S_c, w_s \in S_c\}. \end{aligned}$$

In other words, given a sentence $c \in C$ and synonymy-sets S, the sentences originating from *correct* substitutions are given by $R(c)$; here, a sentence $c' \in R(c)$ originates from c by substituting a word $w_c \in c$ with one of those synonyms which are contained in the annotated synset. In $Q(c)$ the word w_c is replaced with a word w_s that appears in *any* synset that also contains w_c. Note that an automated adversary can correctly reproduce $R(c)$ only if he can solve the WSD problem.

The human interactive proof is carried out in several phases:

1. In the generation phase, the tester composes n test instances $t_1, \ldots, t_n$, where n acts as security parameter. Each instance t_i consists of all sentences in $Q(c)$ for a randomly selected sentence $c \in C$ in random order. All test instances are then presented (in a human-readable form and in a random arrangement) at once to the testee.
2. In the testing phase, the testee solves the problem, by selecting the sentences that are meaningful replacements of each other in each test-instance. The testee returns the selections to the tester.
3. In the verification phase, the hypothesis "For each test instance t_i, the testee could distinguish sentences chosen from $Q(c)$, from sentences chosen from $R(c)$" is confirmed or rejected by the tester. More precisely, the tester checks for each test instance t_i, whether the testee selected all sentences in $R(c)$ and no sentences in $Q(c)\backslash R(c)$. The tester accepts, if the testee answered all instances t_i correctly.

[1] The actual number will be slightly smaller, because in WordNet, synsets are assigned with regard to semantic considerations. A word can be assigned to multiple synsets, that have distinct identifiers, but the same elements. Of course these should be counted only once, for our purposes.

[2] Strictly speaking, there could be multiple appearances of w_c in c. In this case we consider only the first occurrence of w_c.

5 Learning HIPs

A basic problem is that any resource used by a machine to automatically generate test instances will allow an adversary to solve them. For example, using WordNet-glossaries for C is *not* a good idea from a strategic point of view, since they are public, and would therefore allow the adversary to solve the test in the same way an answer to the test is evaluated. In particular, the annotation sa must remain secret (which violates Kerckhoffs' principle).

However, we believe that—in practice—relying on such a private datasource is not a great limitation, if we see it as a dynamic component that grows as the system is being used, and only needs to be seeded with initially private hand-annotated data.

For any $c \in C$, let

$$P(c) = \{subst(c, w_c, w_p) \mid w_c \in c, w_p \in W\}$$

be the set of sentences originating from c by substituting w_p for w_c, where w_p is chosen randomly from the set of all words W.

We can use answers received from a human testee to learn new annotations. For this purpose, we present sentences from $Q(c)$ to the testee as above in order to judge whether he is a human. In addition, we present randomly chosen sentences $p \in P(c)$ together with all the sentences from $Q(p)$. This second batch of sentences will be used to train the HIP for future tests.

For each group, pick the sentences that are meaningful replacements of each other:	
○ We'll send your order tomorrow.	c
○ We'll ship your order tomorrow.	$c_1 \in R(c)$
○ We'll broadcast your order tomorrow.	$c_2 \in Q(c)$
○ We'll cough your order tomorrow.	$d \in P(c)$
...	
○ We'll take your order tomorrow.	$p \in P(c)$
○ We'll accept your order tomorrow.	$p_1 \in Q(p)$
○ We'll hire your order tomorrow.	$p_2 \in Q(p)$

Fig. 4. The HIP can learn from these answers.

Figure 4 shows an example. By random choice we have replaced *send* by *cough*, and the resulting sentence d makes no sense. However, we might just as well end up with a replacement that yields a meaningful sentence, as is the case with sentence p, where *send* has been replaced by *take*. Unfortunately, at this point we don't know which sense the word *take* is used in, since it is contained in two synsets: {*take*, *rent*, *hire*} as in *We took a sailing boat* or {*take*, *accept*} as in *We will gladly take your orders*. This is why the sentences from $Q(p)$ need to be presented as well. A human user will come to the conclusion that p_2 makes no sense; this allows the tester to conclude that the synset $S_p = \{take, accept\}$ contains the correct substitutions for the word $w_p = take$ in the linguistic context imposed by $p =$ *We'll take your order tomorrow*. Thereby the tester can learn a new annotation by remembering the sentence p together with its annotation $S_p = sa(p, w_p)$ for future use. Of course, this should be done only if the tester collected evidence in the verification phase that the testee was in fact human.

4. In the learning phase, sentence $p \in P(c)$ can be added to the corpus C, if the (human) testee considered it meaningful. An association $sa(p, w_p) = S_p$ can be added to the annotation, if each sentence originating from the substitution of w_p by any $w_x \in S_p$ in p was considered a correct replacement by the testee.

If we think of large-scale scenarios such as free e-mail providers telling humans from web-robots that register for free email accounts, we know that such a resource would grow very fast, and we could therefore rely on a highly sophisticated lexical resource originating from a very well trained language learner. A user in the scale of a business that relies on web-robots to sign up for free e-mail accounts, on the other hand, is very unlikely to succeed in outperforming such a linguistic resource.

6 Conclusions and Future Research

In this contribution we have identified word-sense ambiguity as a very promising linguistic phenomenon to build a secure text-based HIP upon. We presented the details of a construction, allowing us to distinguish computers from humans via purely textual interfaces in a fully automatic way. We also showed that it is possible to use answers provided by humans as part of the test for training the linguistic model that serves as a back-end of the HIP.

Although we cannot claim to have solved the problem of creating a CAPTCHA in the text-domain (in the sense of a facility that does not rely on any private resources but a randomness-source), we showed that the learning nature of our HIP helps us to overcome limitations arising from a private linguistic model.

Transformations that provide for a serious computational obstacle to anyone trying to reverse them, paralleling the construction of current image-based CAPTCHAs, have not yet been found for the text-domain. However, we hope to have identified a direction where to look for them, by pointing out the relevance of natural language semantics to the topic. Lexical models provide only for the tip of the iceberg of natural language semantics, so we believe it will be fruitful to investigate the application of other linguistic models as well.

References

1. Tsz-Yan Chan. `http://drive.to/research`.

2. Tsz-Yan Chan. Using a text-to-speech synthesizer to generate a reverse turing test. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, page 226. IEEE Computer Society, 2003.
3. Allison L. Coates and Richard J. Fateman. Pessimal print: A reverse turing test. In *Sixth International Conference on Document Analysis and Recognition (ICDAR '01)*, 2001.
4. Philip Edmonds. Introduction to Senseval. *ELRA Newsletter*, 2002. Available electronically: `http://www.senseval.org/publications/senseval.pdf`.
5. Miller et. al. Semcor 1.6. `ftp://ftp.cogsci.princeton.edu/pub/wordnet/semcor16.tar.gz`, 1998.
6. Miller et. al. and Rada Mihalcea. Semcor 2.0. `http://www.cs.unt.edu/~rada/downloads/semcor/semcor2.0.tar.gz`, 2004.
7. Christiane Fellbaum, editor. *WordNet, An Electronic Lexical Database.* MIT Press, 1998.
8. Philip Brighten Godfrey. Text-based CAPTCHA algorithms. In *First Workshop on Human Interactive Proofs*, 2002. Unpublished Manuscript. Available electronically: `http://www.aladdin.cs.cmu.edu/hips/events/abs/godfreyb_abstract.pdf`.
9. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An on-line lexical database. `http://www.cogsci.princeton.edu/~wn/5papers.ps`, August 1993.
10. William G. Morein, Angelos Stavrou, Debra L. Cook, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. Using graphic turing tests to counter automated ddos attacks against web servers. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pages 8–19. ACM Press, 2003.
11. Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume I, 2003.
12. Moni Naor. Verification of a human in the loop or identification via the turing test. Unpublished Manuscript. Available electronically: `http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps`, 1997.
13. Benny Pinkas and Tomas Sander. Securing passwords against dictionary attacks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 161–170. ACM Press, 2002.
14. Bartosz Przydatek. On the (im)possibility of a text-only captcha. In *First Workshop on Human Interactive Proofs*, 2002. Unpublished Abstract. Available electronically: `http://www.aladdin.cs.cmu.edu/hips/events/abs/bartosz_abstract.pdf`.
15. Philip Resnik. Selectional preference and sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*, April 1997.
16. Senseval. `http://www.sle.sharp.co.uk/senseval2/Results/all_graphs.xls`, 2001. accessed March 6, 2004.
17. Patrice Y. Simard, Richard Szeliski, Josh Benaloh, Julien Couvreur, and Iulian Calinov. Using character recognition and segmentation to tell computer from humans. In *Seventh International Conference on Document Analysis and Recognition*, volume I, 2003.
18. Alan M. Turing. Computing machinery and intelligence. *Mind*, 49:433–460, 1950.
19. Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. Hips. `http://www.aladdin.cs.cmu.edu/hips/`.

20. Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. CAPTCHA: using hard ai problems for security. In *Advances in Cryptology, Eurocrypt 2003*, volume 2656 of *Springer Lecture Notes in Computer Science*, pages 294–311, May 2003.
21. Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.
22. Xerox PARC. *First Workshop on Human Interactive Proofs*, January 2002.