Text-based CAPTCHA Algorithms*

Philip Brighten Godfrey

15th December 2001

Abstract

A CAPTCHA is a test that a computer can automatically administer to determine if a user is a human or a computer. Several aural and visual CAPTCHAs have already been developed, but a completely text-based CAPTCHA is still elusive. In this paper I describe the problems we've encountered in building such a system and current areas of exploration.

1 Introduction

A CAPTCHA is a Completely Automatic Public Turing Test to tell Computers and Humans Apart. It is a test that can be generated and graded by a computer. Most humans can easily pass the test, but computers cannot.

A CAPTCHA can be used to protect a system which is intended to be used only by humans. For example, such a test might be used as part of an online survey to prevent automated programs from "stuffing the ballot box" with thousands of submissions.

A system based on visual recognition of distorted text was developed by the HIP Project at Carnegie Mellon and is being used by Yahoo! to prevent automated programs ("bots") from signing up for hundreds or thousands of free email accounts, which spammers use to send advertisements.

A text-based system would be interesting and useful for several reasons. First, text is a new medium; all current tests are either visual or aural. Second, blind people cannot pass a visual CAPTCHA test. Third, a text-based interface would be more convenient in certain situations, such as a text-only console on a Unix system.

In the following sections I will describe a flawed text-based CAPTCHA and the fundamental problem which afflicts all of our text-based CAPTCHAs thus far.

2 An example broken text-based CAPTCHA

The following is a preliminary system we developed that exploits humans' ability to analyze the meaning in sentences.

The test is generated from a short passage of English text. The computer administering the test selects one word (a noun, verb, or adjective) from the source text. It replaces the chosen word with another "bogus" word selected randomly – either from a set of words of the same part of speech as the chosen word, or, more generally, according to some probability distribution. To pass the test, the user must identify the bogus word. The hope is that a human will easily be able to identify the word which does not make sense in the context of the sentence, while that same task would be very hard for a computer.

The following is an example of a test sentence in which one word has been replaced by a bogus word of the same part of speech as the original word:

*This research is sponsored in part by NSF Grant Nos. CCR-0122581 and CCR-0085982, and in part by Carnegie Mellon's Undergraduate Research Initiative. These results represent the views of the author and not those of Carnegie Mellon University.

> A chair had been placed close to the central window, with its back turned towards it. In this I was asked to sit, and then Mr. Rucastle, walking up and down on the other side of the room, began to tell me a series of the funniest laurels that I have ever listened to.

The user would be prompted to select the word which doesn't make sense in context.[1] You may try an interactive demonstration of the test online at `http://ragtime.bigw.org/~godfreyb/CAPTCHA/`.

This system has two major problems.

First, it is possible for a computer to pass the test when the bogus word is chosen based on part-of-speech. I developed a program based on a trigram model[2] which can pass the CAPTCHA approximately 39% of the time. This is significant since a random guess would be correct only about 8% of the time. Indeed, there is promise that an improved version would do even better. But we can fight fire with fire. The trigram model can be defeated by selecting the bogus word randomly according to the probability distribution generated by the trigram model, rather than by part-of-speech. Using this method, the bogus word would appear perfectly normal to the trigram model. Those changes, in turn, would likely affect the difficulty of the exam for a human, which raises new questions and trade-offs.

The second problem is much more difficult. We must find a secure way to obtain a coherent sentence in which to put the bogus word. I'll refer to this as the "Source Text Problem".

3 The Source Text Problem

All text-based CAPTCHA systems we've considered

The "Find the Bogus Word" CAPTCHA relies on a source of coherent text. Further, the source text must not be attainable by an attacker; if the attacker can generate the source text given a challenge, then it can pass the test. Other systems we've considered have this same requirement:

- **Distinguishing between coherent and incoherent text.** The user is presented with a coherent sentence and an incoherent (e.g. trigram-generated) sentence and asked to identify which sentence makes more sense. Here
- **Removing inserted words.** The computer giving the exam inserts many words into a coherent sentence and presents the modified sentence to the user. The user must identify the words of the original sentence.

It is tempting to extract the text from a large private corpus. But that doesn't satisfy the "Public" requirement of a CAPTCHA – the entire system, except for a stream of random bits used to generate each challenge, must be public. Additionally, a corpus of private text has practical problems; even a very large corpus is finite, and we'll eventually have to reuse material from tests we've already presented.

There are several ways we could get the source text.

- **Generate coherent text.** A computer can generate random English text based on systems such as context free grammers. But this can be parsed just as well as it can be generated, so in some sense computers "understand" these sentences just as well as a human can. This area needs more exploration; perhaps we can effectively generate a sentence that can't be parsed.
- **Obfuscate public text.** We could extract text from a large public corpus, such as the World Wide Web, and obfuscate it before using it in the CAPTCHA. That is, we must transform a sentence s into another sentence $s' = f(s)$ such that

  1. $f^{-1}(s')$ is hard to compute, and

[1] In this case, the correct response is "laurels", which replaced "stories" in the original text. This example was taken from the Adventures of Sherlock Holmes, by Arthur Conan Doyle.

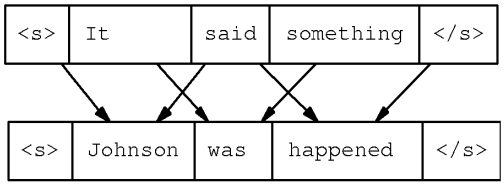
[2] Trigram models use statistical methods to predict the likelihood of a given word occurring, given the the previous two words as context.

2. if s is coherent, then s' is coherent.

The last method initially appeared to be the most promising. The trick is to design a transformation f that satisfies the two properties above. We explored several strategies unsuccessfully.

- Replace words with synonyms or more general counterparts (hypernyms). For example, "elm" might be replaced with "tree", "plant", or "organism". Unfortunately, an attacker with sufficient computing power can still obtain the source text through a search of the corpus.
- Insert many bogus words into the sentence. There is a tradeoff here. If we insert a few bogus words, some meaning is preserved in the sentence, but a search of the corpus is not defeated. If we insert enough words that searching is defeated, no significant meaning is discernible in the transformed sentence.

Currently we are exploring a "Total Trigram Transformation". This transformation replaces every word in the sentence with a new word. Suppose the words in the original sentence are w_0, ..., w_n and the words in the transformed sentence are w'_0, ..., w'_n. Then each w'_i is chosen randomly from a trigram model distribution based on w_{i-1} and w_{i+1}. Below is a diagram of the transformation.



A CAPTCHA based on this transformation works as follows. We have access to a large public corpus of well-written sentences. Extract a random sentence from the corpus; this is the Good Sentence. Generate a second sentence with a trigram model trained on the corpus; this is the Bad Sentence. Transform these two sentences. Present the Transformed Good Sentence and the Transformed Bad Sentence to the user, and have the user identify which one of the two makes more sense. The hope is that enough of the structure of the sentence will be preserved that a human will be able to decide which sentence "flows" better, and that the sentence is obfuscated enough that the CAPTCHA can't be broken by searching the corpus for the original sentence. Preliminary results indicate that humans may be able to do somewhat better than guessing.

4 Conclusion

Many working or promising visual and aural CAPTCHAs have been designed, even following similar patterns of obfuscation that have been hard in the text domain. Why are these other domains easier? In general, it seems much easier to distort audio or images while preserving meaning than it is to do the same to text. One reason may be that humans are excellent at processing images and audio; we're constantly looking at distorted images or listening to speech over background noise. We process distorted text relatively less frequently.