July, 2014

Darwin: A Ground Truth Agnostic CAPTCHA Generator Using Evolutionary Algorithm

Eric Y Chen, *Carnegie Mellon University*
Lin-Shung Huang, *Carnegie Mellon University*
Ole J Mengshoel, *Carnegie Mellon University*
Jason D Lohn, *Carnegie Mellon University*

Darwin: A Ground Truth Agnostic CAPTCHA Generator Using Evolutionary Algorithm

Eric Y. Chen, Lin-Shung Huang, Ole J. Mengshoel, and Jason D. Lohn
Carnegie Mellon University
{eric.chen, linshung.huang, ole.mengshoel, jason.lohn}@sv.cmu.edu

ABSTRACT

We designed and implemented Darwin, the first CAPTCHA generator using evolutionary algorithm. We evaluated the effectiveness of our proposed CAPTCHAs with MTurk users (non-attackers) and Antigate workers (attackers). Due to our ground-truth agnostic fitness function, we are able to discover a new category of CAPTCHAs in which attackers answer correctly but non-attackers answer incorrectly.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

Keywords

CAPTCHA, Evolutionary Algorithm, Mechanical Turk

1. INTRODUCTION

CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) [4] are tests designed to differentiate human beings from computers. CAPTCHAs are commonly used to protect websites against abusive *bots* that run automated tasks over the Internet, such as posting spam on message boards, creating fraudulent accounts or sending votes to online polls. While the state-of-the-art, reCAPTCHA [5], can be effective against OCR-based bots [1], existing CAPTCHAs were not designed to prevent attackers that actually employ cheap *human* labor overseas to solve the tests (e.g., Antigate.com). In this work, we generate CAPTCHAs that are effective in distinguishing Antigate workers (attackers) from MTurk users (non-attackers). Our hypothesis is that Antigate workers (attackers) are more experienced in solving CAPTCHAs than average MTurk users (non-attackers), and may solve CAPTCHAs differently (e.g. characteristically higher accuracy). We built a system called Darwin based on evolutionary algorithm to find CAPTCHAs that effectively differentiate attackers (employed by CAPTCHA solving services) from non-attackers (recruited over MTurk [3]). Interestingly, our Darwin system not only generated CAPTCHAs that MTurk users would solve correctly and Antigate workers would solve incorrectly, but also discovered a new category of *ground-truth agnostic* CAPTCHAs. Ground-truth agnostic CAPTCHAs are interesting because attackers would answer them correctly (and fail the test) but non-attackers would answer incorrectly (and pass the test, counterintuitively).

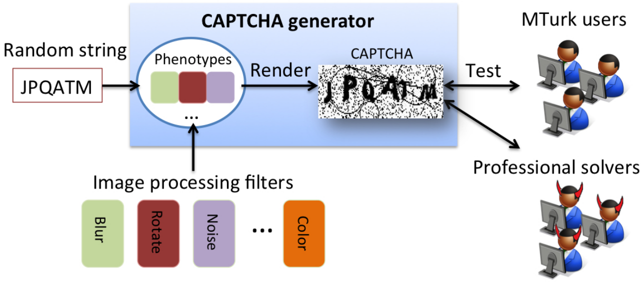


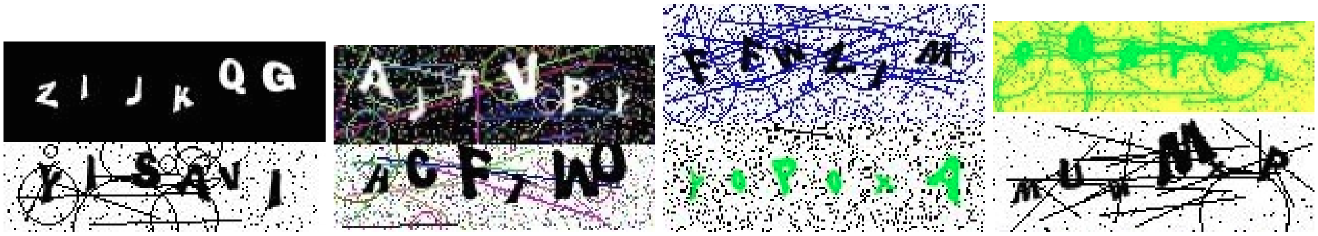
Figure 1: The Darwin system generates CAPTCHA images from random text strings by applying a set of image processing filters. Each CAPTCHA image is evaluated by MTurk users (non-attackers) and professional solvers (attackers) via Antigate.com to find effective filters and parameters.

2. DARWIN SYSTEM ARCHITECTURE

Our system Darwin generates CAPTCHAs by passing randomly generated text strings (as the solution) through a series of image processing filters to render a distorted image of the random string (as the test). The architecture of Darwin is illustrated in Figure 1. Darwin learns how to generate effective CAPTCHAs by using an evolutionary algorithm; each *genotype* in the population is a binary string representation of a specific combination of image processing filters and their corresponding parameters (the *phenotype*). At each generation, Darwin generates a corpus of new CAPTCHA images based on the phenotypes in the population and evaluates the fitness of each CAPTCHA. Lastly, individuals are selected based on their fitness values to produce offsprings through *mutation* (rate=0.1) and *cross-over* (rate=0.8) operations, and then passed on to the next generation.

Phenotype and Genotype. We utilized a number of image processing filters listed below (in four main groups): (1) **font properties**: font, font size, font color, character spacing, character rotation, vertical displacement, (2) **image properties**: width, height, transparency, background

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598413.



(a) Type 1 (19%): MTurk users and Antigate users both correct.

(b) Type 2 (22%): MTurk users correct, Antigate users incorrect.

(c) Type 3 (6.6%): MTurk users incorrect, Antigate users correct.

(d) Type 4 (52%): MTurk users and Antigate users incorrect.

Figure 2: Four different types of CAPTCHAs with samples images generated by Darwin.

color, (3) **visual effects**: Gaussian blur, grayscale, and (4) **random noise**: pixels, lines, circles. Each image processing filter is configurable through certain parameters. For example, the angle parameter for the "character rotation" transformation filter is a number between 0 and 360.

Fitness Evaluation. Unlike most existing evolutionary algorithms, Darwin's evaluation function is not a computer program, but rather real users from MTurk and Antigate (an online CAPTCHA solving service). The fitness value is calculated from the result of real user tests, providing a qualitative measurement of the individual's genotype for deciding whether this individual will carry on to the next generation. The fitness function is:

$$\frac{\sum_{i=1}^{N} Lev(M_i, A_i)}{N[1 + Lev_{Avg}(M_{i..N}) + Lev_{Avg}(A_{i..N})]} \quad (1)$$

where N is the number of different CAPTCHAs generated for each individual. M_i and A_i represents the user's answer for the *ith* CAPTCHA from MTurk and Antigate, respectively. $Lev(A, B)$ represents the Levenshtein distance between string A and B. That is, the minimum number of single-character edits required to transform one string into the other; and $Lev_{Avg}(S)$ represents the average Levenshtein distance between a set of strings S. In layman's terms, our fitness function is the average difference between MTurk user's answers and Antigate worker's answers, divided by the consistency of their answers.

One important observation about our fitness function is that it is **ground-truth agnostic**. That is, we need not assume that a CAPTCHA must be solved correctly by non-attackers. In other words, an effective CAPTCHA could possibly be intentionally difficult for non-attackers (MTurk users) to solve correctly, as long as we can distinguish their results from attackers (Antigate workers).

Diversity Maintenance. To introduce diversity into our populations, we employed the Age-Layered Population Structure (ALPS) [2] technique. The basic idea of ALPS is to divide the population into multiple sub-populations (or *layers*) based on their age. At random intervals, Darwin introduces new, randomly generated individuals into the youngest layer. As they age, individuals move from layer to layer but may be out-competed by individuals of similar age. To keep our total population size constant, we set a maximum layer number (3 in our evaluation) where individuals from the oldest layer always compete in a free-for-all manner. The advantage of using this technique is that it gives newly created individuals time to evolve and fine-tune themselves before competing with more fit individuals.

3. EVALUATION AND DISCUSSION

We ran Darwin for 50 generations and evaluated a total of 5250 CAPTCHAs. Based on evaluation results, we categorized CAPTCHAs into four different types (with actual images) shown in Figure 2. Type 1 CAPTCHAs are solved correctly by both MTurk users and Antigate workers. Type 2 CAPTCHAs are solved correctly by MTurk users but incorrectly by Antigate workers. Type 3 CAPTCHAs are solved correctly by Antigate workers but incorrectly by MTurk users. Finally, Type 4 CAPTCHAs are solved incorrectly by users from both platforms. To select CAPTCHAs that are suitable for real world usage, we recorded 300 individuals with the highest fitness values that we refer to as *elites*. Each of these individuals generated 2 CAPTCHAs. We eliminated all elites that generated a Type 1 or Type 4 CAPTCHA. In addition, we eliminated all elites that generated inconsistent CAPTCHAs (e.g., one CAPTCHA is Type 2 while the other is Type 3). Out of the four types, Type 1 and Type 4 are less valuable to us. Type 2 is considered to be "ideal" under the current CAPTCHA paradigm because non-attackers (MTurk users) solved them correctly while attackers (Antigate workers) solved them incorrectly. Type 3 is the special case of ground-truth agnostic CAPTCHAs that our attackers solved correctly while non-attackers didn't. Hence, when using Type 3 CAPTCHAs, websites must reject users who answered them correctly (which is counterintuitive). In real world deployment scenarios, we advise developers to use a mixture of Type 2 and Type 3 CAPTCHAs.

4. REFERENCES

[1] E. Bursztein, M. Martin, and J. C. Mitchell. Text-based CAPTCHA strengths and weaknesses. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2011.

[2] G. S. Hornby. ALPS: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2006.

[3] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the crowdworkers?: shifting demographics in mechanical turk. In *Extended Abstracts on Human Factors in Computing Systems*, 2010.

[4] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of EUROCRYPT*, 2003.

[5] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 2008.