2Z-3

HMM-based CAPTCHA Breaker for Overlapped Symbols

Shotaro Sano† **Takuma Otsuka**† **Hiroshi G. Okuno**†
†Graduate School of Informatics, Kyoto University

1. Introduction

CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) are programs that distinguish humans from automated programs by presenting tasks that humans can easily solve but computers cannot. Many Websites use CAPTCHAs to prevent their services from various abuses such as flooding from spam accounts. In most cases, the user is required to answer the characters of a distorted image, also known as a visual CAPTCHA.

While they have been widely used in recent Web services, even visual CAPTCHAs in popular Web services (such as Google, Microsoft, Yahoo!) are sometimes easily solved by simple machine learning algorithms [1]. This fact immediately threatens the quality of many Web services with unauthorized accesses by malicious programs. Thus, there has been a huge demand to organize guidelines for the design of secure visual CAPTCHAs by examining breaking techniques prior to malicious users.

Hindle et al. [2] summarized the attacking process of automated programs as three steps: preprocessing, segmentation, and classification. The preprocessing stage reduces redundant information on the question such as background clutter and noise. The segmentation stage divides the preprocessed information into regions of single character. Finally, the classification stage labels each character with a certain supervised method.

Based on this attacking process, numerous security assessments have been undertaken. They concluded that the strength of a visual CAPTCHA depends on the difficulty of its segmentation, since given a perfect segmentation, a machine often attains superior accuracy of classification to humans [3].

To prevent automated segmentation, recent visual CAPTCHAs present a question with continuous characters (Figure 1), which so far is considered to be the most secure defensive technique [1]. It makes use of Sayre's paradox: a word cannot be segmented before being recognized and cannot be recognized before being segmented.

In this paper, we discuss a framework that automatically solves visual CAPTCHAs with continuous characters. The framework formulates the decoding process with a well-known sequence recognition method based on the hidden Markov models (HMMs) that is successfully used in automatic speech recognition and cursive handwriting recognition systems.

We tested the efficiency of our framework with actual CAPTCHA data collected from the visual versions of Google's reCAPTCHA‡. The solvers cracked the reCAPTCHA (as of July 2013) with 31.75% accuracy, which means continuous visual CAPTCHAs are no longer safe.

2. ReCAPTCHA schema

An example of the reCAPTCHA questions is shown in upper-left of Figure 1. A question consists of six to eight alphabetic characters including both upper and lower cases. When all characters of the question are correctly estimated, the CAPTCHA is solved. To prevent segmentation from automated programs, the reCAPTCHA removes the space between characters of a question.

In addition, the reCAPTCHA distorts the image of the question. The distortion is applied in two steps: the linear transformation, which transforms the image with an affine filter, and the wavy transformation, which waves the image.

‡ReCAPTCHA is an application programming interface (API) provided by Google to embed the CAPTCHA system, which has been used by various Web services including Google, Twitter, and Facebook.

Figure 1: Visual CAPTCHAs of Google (upper left), Yahoo! (upper right), Microsoft (lower left), and Amazon (lower right). Characters are connected except for Microsoft CAPTCHA.

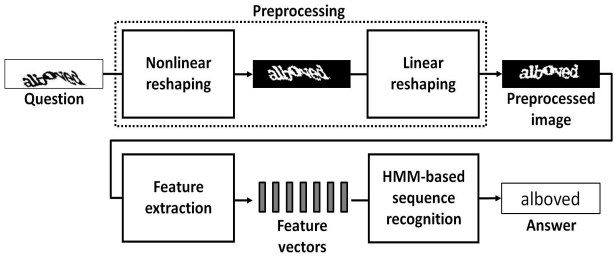

Figure 2: Decoding process of reCAPTCHA solver. Question is decoded in three steps of preprocessing, feature extraction, and HMM-based sequence recognition.

To increase the usability, the reCAPTCHA regards a response as correct even when one of the characters in a question is misestimated in terms of Levenshtein distance. For example, a question whose original answer is "abcdefg" may be labeled as "bbcdefg", "bcdefg", or "aabcdefg."

In summary, the reCAPTCHA adopts the following defensive techniques: using continuous characters, randomized text length, linear transformation, and wavy transformation. The reCAPTCHA also adopts an additional idea to ensure usability by allowing off-by-one error to label a question.

3. ReCAPTCHA solver

Figure 2 depicts the pipeline of the reCAPTCHA solver. The input to the solver is an image of question, and the solver outputs the answer label sequence of the question. The solver decodes an input in three steps: (1) the input image is nonlinearly reshaped, and then linearly reshaped (preprocessing); (2) the preprocessed image is converted into a sequence of feature vectors (feature extraction); and (3) the feature sequence is labeled with the HMM-based sequence recognition method (HMM-based sequence recognition).

3.1 Preprocessing

The linear transformation and the wavy transformation should be reshaped to reduce the horizontal overlap between each character before the question is recognized by the HMMs. This is because the HMM-based recognition method implicitly performs vertical segmentation of the image.

3.1.1 Nonlinear reshaping

Suppose that each pixel value of the thresholded image in the question is represented as $I(x,y)$ where x and y is the horizontal and vertical position of the pixel respectively:

$$I(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ belongs to the text area,} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

First, the center line of the distorted word $c(x)$ is detected:

$$c(x) = \frac{\sum_{w \geq |x-x'|} \{y \cdot I(x',y)\}}{\sum_{w \geq |x-x'|} I(x',y)}, \quad (2)$$

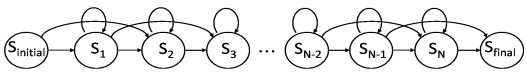

Figure 3: HMM topology for reCAPTCHA solver. State transitions are limited up to two skips.

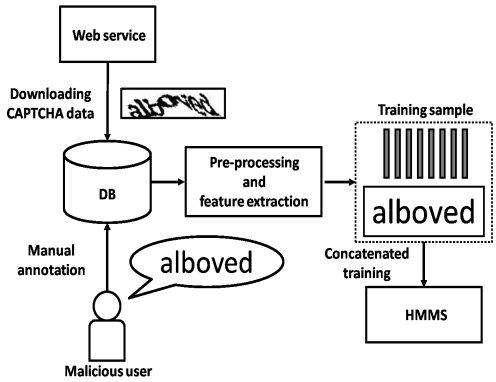

Figure 4: Training framework of reCAPTCHA solver. User manually annotates dataset downloaded from target CAPTCHA and stored in DB. HMMs are trained with annotated data.

where w is the window size. In this study, we set $w = 20$. The reshaped image $J(x,y)$ is obtained by vertically aligning each column of $I(x,y)$ to straighten the detected center line:

$$J(x,y) = I(x,y-c(x)+\frac{h}{2}), \tag{3}$$

where h is the height of the image.

3.1.2 Linear reshaping

The linear reshaping can be represented as a transformation matrix A that has single parameter θ as the following equation:

$$A = \begin{pmatrix} 1.0 & \tan\theta \\ 0.0 & 1.0 \end{pmatrix}. \tag{4}$$

The solver finds the optimum reshaping parameter θ that minimizes the horizontal projection of the text area.

3.2 Feature extraction

A sliding window along the horizontal axis is used to extract feature vectors. The window size is set to five columns in this study, and the height of each image of the reCAPTCHA is 57 pixels. Thus, a 285-dimensional vector is extracted for each window. The feature vectors are decomposed into 25 dimensions with sparse principal component analysis.

3.3 HMM-based sequence recognition

Given a sequence of feature vectors O, this module finds the optimal answer sequence of labels $\hat{W}$, out of all possible answers L. This problem is formulated as the following equations:

$$\hat{W} = \arg\max_{W \in L} P(W)P(O|W), \tag{5}$$

$$= \arg\max_{W \in L} P(O|W), \tag{6}$$

where Equation (6) is obtained because $P(W)$ is considered to be equal for all possible answers.

The optimal answer for Equation (6) is found out with the HMM-based sequence recognition method [4]. In this method, an HMM is trained to compute the likelihood of a character, and several HMMs compose a concatenated HMM to compute the likelihood of a question $P(O|W)$. For an arbitrary possible answer W, its corresponding concatenated HMM can be made up. Therefore, the optimal answer $\hat{W}$ is obtained by searching it from the set of all possible answer L to maximize $P(O|W)$.

As shown in Figure 3, a character is modeled as a left-to-right HMM. Each HMM has 20 states including the non-emission states. Observation likelihood function is 25-dimensional Gaussian distribution for each state.

3.4 Training framework

The HMMs are trained with actual data of the reCAPTCHA. As outlined in Figure 4, the training set of questions is downloaded and stored in the database (DB). The data are manually annotated for each question by the solver's user. Then, the HMMs are trained with the feature sequences extracted from the data in the DB and corresponding transcriptions using the concatenated training [5].

Table 1: Average performance of reCAPTCHA solver

	Closed test	Open test
Strict accuracy	18.38%	10.50%
Off-by-one accuracy	44.94%	**31.75%**

4. Solver evaluation

We evaluated the solver's performance as well as assessed the security of the reCAPTCHA. The experiments were performed with 2000 questions downloaded from actual reCAPTCHA as of July 2013.

The solver's performance is evaluated in terms of two accuracy metrics of "strict accuracy" and "off-by-one accuracy". For strict accuracy, an output is regarded as correct only when it is strictly the same as the original answer. For off-bey-one accuracy, on the other hand, an output is regarded as correct even when the Levenshtein distance between the output and the original answer is less than two. Note that the actual vulnerability of the reCAPTCHA is represented by off-by-one accuracy.

Table 1 lists the average performance evaluated in five-fold cross validation. The solver cracked the reCAPTCHA with 31.75% accuracy.

5. Conclusion

Our solver cracked the visual version of the reCAPTCHA with 31.75% accuracy. We conclude that our solvers disclosed the vulnerability of the reCAPTCHA in accordance with the criteria for breaking CAPTCHAs claimed in previous works, e.g., "a CAPTCHA schema is broken when the attacker is able to reach a precision of at least 1%" by Bursztein et al. [1]. CAPTCHAs even with continuous characters are no longer safe.

Future work contains finding out the weakness of our HMM-based attack and organizing the guidelines to design more secure CAPTCHAs.

References

[1] E Bursztein, M Martin, and J. Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 125–138. ACM, 2011.

[2] Hindle A, M. W Godfrey, and R. C Holt. Reverse engineering captchas. In *15th Working Conference on Reverse Engineering*, pages 59–68. IEEE, 2008.

[3] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *Proceedings of the Second Conference on Email and Anti-Spam*, pages 21–22, 2005.

[4] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[5] K. F. Lee and H. W. Hon. Large-vocabulary speaker-independent continuous speech recognition using hmm. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 123–126. IEEE, 1988.