An Accessible CAPTCHA System for People with Visual Disability – Generation of Human/Computer Distinguish Test with Documents on the Net

Michitomo Yamaguchi[1,3], Toru Nakata[2], Takeshi Okamoto[3], and Hiroaki Kikuchi[1]

[1] Department of Mathematical Modeling, Analysis and Simulation, Graduate School of Advanced Mathematical Sciences, Meiji University, Tokyo, #164-8525 Japan
[2] Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology (AIST), Ibaraki, #305-8568 Japan
[3] Graduate School of Technology and Sciences, Tsukuba University of Technology, Ibaraki, #305-8521 Japan
`yama3san@meiji.ac.jp`, `toru-nakata@aist.go.jp`, `ken@cs.k.tsukuba-tech.ac.jp`, `kikn@meiji.ac.jp`

Abstract. We propose a new scheme of CAPTCHA that does not become a perceptual barrier for disable people. Our CAPTCHA system generates the tests in verbal style, so its use is not limited in specific perceptual channels. The tests are composed of several phrases and there are two kinds of tests: Human users try to (1) distinguish a phrase of strange meaning from others, and (2) identify the common topic among them. In our test we utilize open documents for material. Note that there is quite a large amount of documents on the net, so we can generate brand-new tests every time. One may say that adversaries can look for the phrases over the Internet and get several hints. Our system hides the sources by substituting the consonants of the phrases against such an attack. The mechanism is designed to imitate the phenomenon called "consonant gradation" of natural languages.

Keywords: universal design, aid for the visually-impaired, verbal interaction, information security.

1 Introduction

Demand for New Turing Tests for the Visually-Impaired. The purpose of this paper is to construct an accessible CAPTCHA system specially for people with visual disability.

The challenge-response tests of CAPTCHA [1] (Completely Automated Public Turing test to tell Computers and Human Apart) are widely used to differentiate humans from software agents. Typically, the systems challenge the user to read distorted letters. Artificial intelligence (AI) problems, which is easy for humans but difficult for current AI, are frequently employed for the purpose.

There exists a big social problem that most of the visually-impaired cannot pass current CAPTCHAs. Several researchers [2–4] have pointed out that state-of-the-art audio CAPTCHAs are too difficult for them. A recently study [5] shows that five Japanese subjects with visual impairments have tried 10 times to pass Google's audio CAPTCHA, but nobody have succeeded even one. For those reason, audio CAPTCHAs do not work for a substitution of visual ones.

We reconfirm requirements for CAPTCHA system as follows.

Accessibility. Tests depending on the specific perceptual ability should not be used as CAPTCHA anymore.

Ability as a Turing Test. Tests must distinguish humans from software agents.

Ability of Auto-generating New Tests. The system must generate brand-new tests without limitation on amount.

Related Work. Maintaining the quality of the tests is also a hard problem.

The test system of Holman et al. [6] shows pairs of a picture and a sound to a user and test that the user can understand the situation of the materials. This scheme needs to collect a myriad of pictures and sounds data. We deem it difficult to collect them enough for practice.

Contextual cognition and theory of mind have been widely studied in cognitive science. Sally-Anne test [7] is to distinguish one's sphere of knowledge from others' one. In Dick's novel, he showed an idea of Voigt-Kampff [8], which submitted eccentric phrases and checked fluctuations of one's feeling. These tests will work as a Turing test but it is difficult to generate them automatically.

There are already CAPTCHA-like tests in verbal style. Contextual cognition test is the most typical verbal test. Only the humans can evaluate naturalness of the phrases. We can feel difference between natural human-made phrases and machine-translated phrases (Yamamoto et al. [9]) or machine-synthesized phrases with moderate randomness (*KK12* system of Kamoshida et al. [10]).

Regarding information security, the system has to generate brand-new tests for every time, or the adversaries can pass the tests when the system set old ones again. Unfortunately, conventional schemes leverage strings of private documents in order to prevent adversaries from finding out their sources. It is a serious problem for us to use such schemes as a CAPTCHA since the amount of private documents is finite. The system cannot limitlessly generate brand-new tests.

Our Approach. We summarize our approach as follows.

- We utilize contextual cognition tests as a part of the accessible CAPTCHA system.
- Our system collects phrases from a large amount of documents on the Internet to generate unbounded amount of tests.
- Using public documents is not safe, because adversaries may find out sources of phrases by using search engines and utilize them to break the tests. As a countermeasure, our particular system employs substitutes the consonants of the phrases.

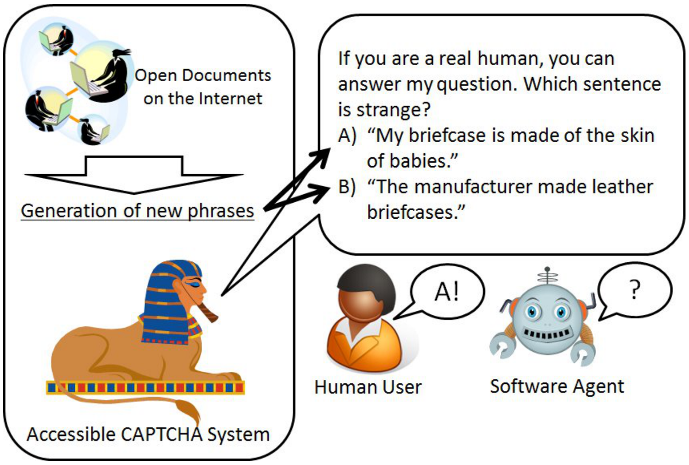
The substitutes makes harder for adversaries to look for the sources of original phrases from such erroneous ones. The mechanism is similar to the phenomenon called

Test of detecting machine-synthesized semi-random phrases:
Q. Which phrase is generated by computer? *A) The world will little note, nor long remember what we say here.* *B) Now we are dedicated in a larger people.*

Test of detecting common topic:
Q. What is the topic common to the following sentences? *– Twitter is an social networking service.* *– Vector processors are used for parallel computing.* *A) Computer, B) Sports, C) Culture.*

Example of Consonant Substitution:
Original: London, Paris, and Rome. *After distorted: Wonton, Pawi, and Home.*

(a) Simple Example of Our Verbal Turing Tests.


If you are a real human, you can answer my question. Which sentence is strange?

A) "My briefcase is made of the skin of babies."

B) "The manufacturer made leather briefcases."

Open Documents on the Internet
Generation of new phrases
Accessible CAPTCHA System
A!
?
Human User
Software Agent

(b) Aspect of Our Proposal.

Fig. 1. Sketch of Our Proposal

"consonant gradation" of natural languages. We expect that most humans can interpret the erroneous phrases to some extent due to linguistical ability.

Our Contributions. We show the brief sketch of our proposal in Fig. 1.

We propose two methods to generate contextual cognition tests. One is test of detecting machine-synthesized phrased generated by Markov chain. It is based on the scheme of *KK12*, and we try to redeem its weakness around discrete co-occurrence of terms. The other method is to generate topic detection tests. The system shows several phrases and challenges the user to answer the topic common to them.

We then implement them as CAPTCHA programs, and evaluate their performances in three points: 1) ability as a Turing test, which must be easily solved by humans, 2) generation of new tests without limitation on amount, and 3) ability of hiding the sources of the phrases appeared in the test.

2 Mechanism of Our System

2.1 Notations

Let X be an operation. We write $x \leftarrow X$ to indicate that x is assigned by X. Let i, j be integers. We denote an integral element x from i to j by $x \in [i, j]$. Let $\{x_i\}$ be a finite set of $x_0, x_1, \ldots$. We write $x \overset{D}{\leftarrow} \{x_i\}$ to indicate that x is sampled randomly from $\{x_i\}$ following a certain distribution D. If $D = \$$, it means that x is sampled uniform-randomly from $\{x_i\}$. We use the same notation of a set for an array.

We will use a multi-associative memory C and write $key \in C$ to indicate that key is a registered key of C. Suppose that, e.g., C is the form $C = \{(key, val_0), (key, val_0), (key, val_0), (key, val_1)\}$ and we pick a value c as $c \overset{D}{\leftarrow} C(key)$. In this case, c will be substituted by val_0 with 75% and by val_1 with 25%.

Let k be a string, which is an array of characters. We denote the length of string k by $|k|$. We write $\mathcal{K} \leftarrow \mathbf{thesaurus}(k)$ to indicate that a thesaurus outputs $\mathcal{K}$, which is the set of synonyms of k.

Let $\mathcal{M}$ be a stock of documents, which is utilized as the source of k. We write $\{m_i\} \leftarrow \mathbf{search}(+k, -\mathcal{K}, \mathcal{M})$ to indicate that a set of phrases $\{m_i\}$ is retrieved from $\mathcal{M}$ by searching with keyword k, without the words in $\mathcal{K}$.

We need morphological analyses for our experiments. We write $A \leftarrow \mathbf{ma}(k)$ to indicate that a morphological analyzer outputs A, which is the array of morphemes of k and the last element of A is a terminal symbol. We denote a concatenation string of each element in A by $\mathbf{concat}(A)$. We write $\{(seg_{M,i}, seg_{R,i})\} \leftarrow \mathbf{da}(m)$ to indicate that a dependence analyzer outputs two segments of documents, which are retrieved from document m. For instance, **da**("I hit the ball with the bat.") outputs $\{(seg_{M,i}, seg_{R,i})\} =$ {("hit", "the ball"), ("hit", "with the bat"), ...}.

2.2 Overview of the System

We begin by showing a framework of our system. Let $\mathcal{V}$ be a *verifier*, i.e. CAPTCHA system and $\mathcal{P}$ be a *prover*, who tries to pass the test. $\mathcal{V}$ determines whether $\mathcal{P}$ is human or not as follows.

1) $\mathcal{V}$ runs a program $\mathcal{G}$ which generates AI problems using the contextual cognition if $\mathcal{V}$ needs to differentiate human from software agent.
2) $\mathcal{G}$ collects phrases from source documents $\mathcal{M}$, analyzes morphology of them, and generate/update corpora. We use open documents in the Internet $\mathcal{M}$.
3) $\mathcal{G}$ generates a pair of (z, a) with the corpora, where z is a problem and a is its answer. The phrases in z have a certain property such as contextual naturalness.
4) $\mathcal{V}$ repeats the steps 2)–3) N times and outputs $\{(z_i, a_i)\}$ to $\mathcal{P}$ as a test. $\mathcal{P}$ can recognize the test by his/her favorite perception since it consists of textual information.
5) $\mathcal{V}$ receives the answers $\{a'_i\}$ from $\mathcal{P}$. $\mathcal{V}$ checks whether $a_i = a'_i$ or not for all i. If the number of correct answers is greater than t, $\mathcal{V}$ judges $\mathcal{P}$ as a human.

Note that we may skip the step (2) in the case of updating if $\mathcal{G}$ can generate brand-new problems in high ratio.

2.3 Basic Components of the Process

We introduce several functions which are the building blocks of our AI problems.

Generation of Corpora. We generate two kinds of corpora: a n-th-order Markov chain modeled corpus (C_0) and a pre-post-state modeled corpus (C_1). They are multi-associative memories. The key of C_0 is a morpheme n-gram and its values are co-occurrence of it. The value of C_1 is a morpheme and key is a couple of morphemes before and after it.

We write $C_0 \leftarrow \mathbf{cpsgen}_0(\{m_j\}, n)$, e.g.: {("I hit", "the"), ("hit the", "ball"), ...} $\leftarrow$ $\mathbf{cpsgen}_0$({"I hit the ball with the bat."}, 2), to indicate the following procedure for all m_j.

1) Compute $mor_j \leftarrow \mathbf{ma}(m_j)$. Assign $C \leftarrow \emptyset$ and $i \leftarrow 0$. C is a local variable, whose data structure is the same as C_0.
2) If $i+n < |mor_j|$, go to the step (3). Otherwise, output C as C_0 and finish this process.
3) Assign $key \leftarrow (mor_j[i], \ldots, mor_j[i+n-1])$, $val \leftarrow mor_j[i+n]$, and $C \leftarrow \{C, (key, val)\}$.
4) If $mor_j[i]$ is an independent morpheme, tags it.
5) Go to the step 2) after computing $i \leftarrow i+1$.

We write $C_1 \leftarrow \mathbf{cpsgen}_1(\{m_j\})$, e.g.: {("I the", "hit"), ("hit the", "the"), ...} $\leftarrow$ $\mathbf{cpsgen}_1$({"I hit the ball with the bat."}), to indicate the following procedure for all m_j.

1) Compute $mor_j \leftarrow \mathbf{ma}(m_j)$. Assign $C \leftarrow \emptyset$ and $i \leftarrow 1$. C is a local variable, whose data structure is the same as C_1.
2) If $i+1 < |mor_j|$, go to the step 3). Otherwise, output C as C_1 and finish this process.
3) Assign $key \leftarrow (mor_j[i-1], mor_j[i+1])$, $val \leftarrow mor_j[i]$, and $C \leftarrow \{C, (key, val)\}$.
4) Go to the step 2) after computing $i \leftarrow i+1$.

Generation of Synthesized Phrases. We generate synthesized phrases by Markov chain of order n. Let ℓ_L be the minimum length of them and ℓ_H be the maximum one. We write $s \leftarrow \mathbf{mcpgen}(n, \ell_L, \ell_H, C_0)$ to indicate the following procedure.

1) Pick $\ell \overset{\$}{\leftarrow} [\ell_L, \ell_H]$.
2) We generate the beginning of a synthesized phrase. Pick *key* uniformly and randomly from tagged keys in C_0, $val_0 \overset{D}{\leftarrow} C_0(key_0)$, and assign $ary \leftarrow (key_0, val_0)$.
3) If val_0 is a terminal symbol or $\ell < |\mathbf{concat}(ary)|$, output $\mathbf{concat}(ary)$ as a synthesized phrase and finish this process. Otherwise, go to the step 4).
4) Assign $key_0 \leftarrow (ary[|ary| - n], \ldots, ary[|ary| - 1])$. Pick $val_0 \overset{D}{\leftarrow} C_0(key_0)$ and assign $ary \leftarrow (key_0, val_0)$. Go to the step 3).

2.4 Countermeasures against Breaking the Test by Software

Extraction of Discrete Co-occurrence Features. In a linguistic sense, co-occurrence is interpreted as an indicator of semantic proximity or an idiomatic expression. We call *discrete co-occurrence* to indicate a fixed pattern of terms that often appear together with keeping several distance each other. For example, phrase of "not only . . . , but also . . ." is a discrete co-occurrence pattern.

We generate unnatural phrases by Markov chain of small order n. However, in that case, it is difficult for us to generate phrases which have a feature of discrete co-occurrence. Because each term of the phrases only depends on the last n terms. We are afraid of adversaries to use this shortcoming.

Therefore, we extract the feature as $\hat{C}_0$ and append it to C_0. We write $\hat{C}_0 \leftarrow \mathbf{dcogen}(\{m_j\}, n, C_1)$ to indicate that, for all m_j, the following procedure.

1) Compute $\{(seg_{M,i}, seg_{R,i})\} \leftarrow \mathbf{da}(m_j)$. Assign $C \leftarrow \emptyset$. C is a local variable, whose data structure is the same as C_0 and $\hat{C}_0$.
2) For $(seg_{M,i}, seg_{R,i}) \in \{(seg_{M,i}, seg_{R,i})\}$, we do as follows.
  2-1) Compute $mor_{M,i} \leftarrow \mathbf{ma}(seg_{M,i})$ and $mor_{R,i} \leftarrow \mathbf{ma}(seg_{R,i})$.
  2-2) Assign $key_1 \leftarrow (mor_{M,i}[|mor_{M,i}| - 1], mor_{R,i}[0])$. If $key_1 \in C_1$, pick $val_1 \overset{D}{\leftarrow} C_1(key_1)$. Otherwise, we assign an empty string to val_1.
  2-3) Assign $ary \leftarrow (mor_{M,i}, val_1, mor_{R,i})$, $key_1 \leftarrow (ary[0], \ldots, ary[n-1])$, $val_1 \leftarrow (ary[n], \ldots, ary[|ary| - 1])$, and $C \leftarrow \{C, (key_1, val_1)\}$. If $key_1[0]$ is an independent morpheme, tags it.
3) We output C as $\hat{C}_0$ and finish this process.

The reason why we insert val_1 between $seg_{M,i}$ and $seg_{R,i}$ is that because of randomness. The phrase which consists of $seg_{M,i}$ and $seg_{R,i}$ only cannot be used as a part of an unnatural phrase.

Consonant Substitution to Hide the Document Source. To protect the tests from attacks using search engines, our system hides the sources by substituting the consonants of the phrases in them. Let s_p be a string without consonant substitution and s_a be a string with one. Let r_L be a the minimum number of the substitution and r_H be a maximum one. We write $s_a \leftarrow \mathbf{cogd}(s_p, r_L, r_H)$ to indicate the following procedure.

1) This is an initialize step.
  – Let $\mathcal{L}_L$ be a set which includes all kinds of a group of Japanese consonant, that is, $\mathcal{L}_L = \{\text{"}group-a\text{"}, \text{"}group-ka\text{"}, \ldots\}$.

- Convert kanji (Chinese characters) of s_p into hiragana (Japanese syllabary characters) and output the result as s_m.
- Pick $r \overset{\$}{\leftarrow} [r_L, r_H]$.

2) We check the consonant of s_m.

- For each letter of s_m, check which the group of consonant is and output the result as an array L_{s_m}, whose element is a kind of the group of Japanese consonant. For example, we get (“*group−a*”, “*group−sa*”, “*group−ga*”, “*group−a*”) from the term “a-sa-ga-o”.
- If $|L_{s_m}| < r$, compute $r \leftarrow |L_{s_m}| - 1$.
- If $r > 0$, go to the step (3). Otherwise, output s_m as s_a and finish this process.

3) We substitute the consonants of s_m.

- Pick $u \overset{\$}{\leftarrow} L_{s_m}$ and $v \overset{\$}{\leftarrow} \mathcal{L}_L \backslash u$.
- Check an index i of the element u in the array L_{s_m}.
- Replace the i-th letter in s_{s_m} by a letter of the same group of vowel in v
- Go to the step (2) after computing $r \leftarrow r - 1$.

The output of **cogd** is similar to the phrase which includes several “mistakes” such as misprint and mishearing. We expect that human can correct them by interpreting contexts and his/her experience [11–13].

2.5 Detail of Generation of Our CAPTCHA

We show two kinds of AI problems concerning contextual cognition.

Markov-Chain Phrase Problem. Markov-chain phrase problem is that a prover selects the most unnatural phrase among several synthesized phrases.

We use *word salad* as a synthesized phrase by Markov chain. Word salad usually keeps grammatical correctness to some extent and stands for certain meanings. The naturalness of their meanings differs in respect to the order n of the generation algorithm i.e. the synthesized phrases become more unnatural as n is small, vice versa.

We give an account of an algorithm $\mathcal{G}_0$, which generates a Markov-chain phrase problem. Let p be the number of choices and $\mathcal{M}$ be phrases of source documents. Let n_{NP} be the order of Markov chain to generate natural phrases and n_{WS} be one to generate word salad. Let ℓ_L and ℓ_H be the minimum/maximum length of synthesized phrases, respectively. Let r_L and r_H be the minimum/maximum number of substituting consonants, respectively. We require $p > 1$, $n_{NP} > n_{WS} > 0$, $\ell_H > \ell_L > 0$ and $r_H > r_L > 0$. $\mathcal{G}_0$ is inputted p, n_{NP}, n_{WS}, ℓ_L, ℓ_H, r_L, r_L and $\mathcal{M}$, then outputs (z, a) as follows.

1) The system collects phrases $\{m_j\}$.

- Pick $\{m_j\} \overset{\$}{\leftarrow} \mathcal{M}$.

2) The system generates corpora for Markov chain.

- Compute $C_{0,NP} \leftarrow \mathbf{cpsgen}_0(\{m_j\}, n_{NP})$ to generate natural phrases.
- Compute $C_{0,WS} \leftarrow \mathbf{cpsgen}_0(\{m_j\}, n_{WS})$ to generate word salad.

The system extracts discrete co-occurrence and append them to $C_{0,WS}$.

- Compute $C_1 \leftarrow \mathbf{cpsgen}_1(\{m_j\})$.
- Compute $\hat{C}_{0,WS} \leftarrow \mathbf{dcogen}(\{m_j\}, n_{WS}, C_1)$.

  - Assign $C_{0,WS} \leftarrow \{C_{0,WS}, \hat{C}_{0,WS}\}$.

3) The system generates a word salad and natural phrases by Markov chain.
  - Pick $x \overset{\$}{\leftarrow} [0, p-1]$.
  - Compute $s_x \leftarrow \textbf{mcpgen}(n_{WS}, \ell_L, \ell_H, C_{0,WS})$.
  - For $i \in [0, p-1]\backslash x$, compute $s_i \leftarrow \textbf{mcpgen}(n_{NP}, \ell_L, \ell_H, C_{0,NP})$.

4) The system substitutes consonants of the synthesized phrases.
  - For $i \in [0, p-1]$, compute $z_i \leftarrow \textbf{cogd}(s_i, r_L, r_H)$.

5) The system outputs a problem.
  - Assign $z \leftarrow (z_0, \ldots, z_{p-1})$ and $a \leftarrow x$.

Topic Detection Problem. Topic detection problem is that a prover selects the most related keyword with submitted phrases from choices.

We try to generate the phrases which do not include strings of the choices by using a thesaurus. Moreover, we append a phrase which is not related with the keyword to ones which are related with it.

We give an account of an algorithm $\mathcal{G}_1$, which generates a topic detection problem. Let q be the number of submitted phrases and $\mathcal{K}$ be a set of keyword. We require $p > 1$, $q > 2$, $n_{NP} > 0$, $\ell_H > \ell_L > 0$ and $r_H > r_L > 0$. $\mathcal{G}_1$ is inputted p, q, n_{NP}, ℓ_L, ℓ_H, r_L, r_L, $\mathcal{K}$, and $\mathcal{M}$, then outputs (z, a) as follows.

1) The system chooses a keyword key_t used as a correct answer and generate a set $\hat{\mathcal{K}}_t$, whose element is a synonym of key_t.
  - Pick $key_t \overset{\$}{\leftarrow} \mathcal{K}$ and compute $\mathcal{K} \leftarrow \mathcal{K}\backslash key_t$.
  - Compute $\hat{\mathcal{K}}_t \leftarrow \textbf{thesaurus}(key_t)$.

2) Let $\mathcal{K}_d$ be a set of a dummy keyword and $\hat{\mathcal{K}}_d$ be a set, whose element is a synonym of the corresponding element of $\mathcal{K}_d$. Assign $\mathcal{K}_d \leftarrow \emptyset$ and $\hat{\mathcal{K}}_d \leftarrow \emptyset$. To generate $\hat{\mathcal{K}}_d$, repeat the following steps if $|\mathcal{K}_d| < p-1$.
  - Pick $key_d \overset{\$}{\leftarrow} \mathcal{K}\backslash\mathcal{K}_d$.
  - Assign $\mathcal{K}_d \leftarrow \{\mathcal{K}_d, key_d\}$.
  - Compute $\hat{\mathcal{K}}'_d \leftarrow \textbf{thesaurus}(key_d)$ and assign $\hat{\mathcal{K}}_d \leftarrow \{\hat{\mathcal{K}}_d, \hat{\mathcal{K}}'_d\}$.

3) The system extracts phrases, which are related with key_t but do not include strings of the choices.
  - For $\hat{key}_t \in \hat{\mathcal{K}}_t$, $\{m_j\} \leftarrow \textbf{search}(+\hat{key}_t, -\{key_t, \mathcal{K}_d\}, \mathcal{M})$. For simplicity, we assume that $\{m_j\}$ in this step includes the results for all $\hat{key}_t$.
  - Compute $C_{0,NP} \leftarrow \textbf{cpsgen}_0(\{m_j\}, n_{NP})$.
  - Pick $x \overset{\$}{\leftarrow} [0, q-1]$. For $i \in [0, q-1]\backslash x$, compute $s_i \leftarrow \textbf{mcpgen}(n_{NP}, \ell_L, \ell_H, C_{0,NP})$.

4) The system extracts phrases, which are related with a dummy keyword $\hat{key}_d$ but do not include strings of the choices.
  - Pick $\hat{key}_d \overset{\$}{\leftarrow} \hat{\mathcal{K}}_d$.
  - Compute $\{m_j\} \leftarrow \textbf{search}(+\hat{key}_d, -\{key_t, \mathcal{K}_d\}, \mathcal{M})$ and $C_{0,NP} \leftarrow \textbf{cpsgen}_0(\{m_j\}, n_{NP})$.
  - Compute $s_x \leftarrow \textbf{mcpgen}(n_{NP}, \ell_L, \ell_H, C_{0,NP})$.

5) The system substitutes consonants of the extracted phrases.
  - For $i \in [0, q-1]$, compute $z_i \leftarrow \textbf{cogd}(s_i, r_L, r_H)$.

6) The system outputs a problem. The choices consist of key_t and the elements of $\mathcal{K}_d$.
  - Assign $z \leftarrow (z_0, \ldots, z_{q-1})$ and $a \leftarrow key_t$.

3 Experiment

3.1 Tools and Parameters

We implemented an evaluation program (hereinafter referred to as the **EvaPro**) to evaluate our proposals. We used MeCab [14] to analyze morphology, Cabocha [15] to analyze dependence, and Weblio thesaurus [16] to get synonyms of queried terms.

We show several parameters of our AI problems in **EvaPro**.

Markov-Chain Phrase Problem ($\mathcal{G}_0$). We employed "Aozora Bunko [17]" as $\mathcal{M}$. We then set $p = 4$, $n_{NP} = 7$, $n_{WS} = 1$, $(\ell_L, \ell_H) = (40, 80)$, and $(r_L, r_H) = (2, 5)$.
Topic Detection Problem ($\mathcal{G}_1$). We employed documents which is collectable by GAPI [18] as $\mathcal{M}$. We then set $p = 4$, $q = 5$, $n_{NP} = 7$, $(\ell_L, \ell_H) = (30, 60)$, $(r_L, r_H) = (2, 5)$, and $\mathcal{K} = \{$"*sport*", "*weather*", "*economy*", "*meal*"$\}$. In fact, we set $\mathcal{K}$ in Japanese.

Note that we determine the range of substitution degree (r_L, r_H) and the minimum phrase length ℓ_L the following reason. Let s be a phrase and $\hat{s}$ be a phrase to which the consonant substitution is applied. We assume that adversaries try to restore $\hat{s}$ to its original string s and output s'. If $s' = s$, adversaries can get several hints of our problems to query on s'. Therefore, the number of a prospective for s', that is, $\sum_{i=r_L}^{r_H} \binom{\ell_L}{i}$ should be large. In fact, the number of the prospective for s' is about 1.5 billion under the condition of $\ell_L = 30$ and $(r_L, r_H) = (2, 5)$. It takes about 11 days to find s by using a computer which can operate at 1.8 million per second. (In fact, it is difficult for adversaries without the knowledge of s to confirm whether $s' = s$ or not because several prospective for s' may natural but $s' \neq s$.)

Consequently, it is difficult for adversaries to find s by brute-force attacks.

Meanwhile, we determine ℓ_H owning to an availability. We suppress the amount of phrases by ℓ_H.

Table 1. Results of our Subjective Experiment

(a) Results of Markov-Chain Phrase Test.

Sight	w/o Consonant Substitution		w/ Consonant Substitution	
	Average Accuracy Rate [%]	Capability [%] as a Turing Test	Average Accuracy Rate [%]	Capability [%] as a Turing Test
Totally Blind	89	86	73	**57**
Low Vision	78	82	49	**41**
All	85	83	60	**46**

(b) Results of Markov-Chain Phrase Test.

Sight	w/o Consonant Substitution		w/ Consonant Substitution	
	Average Accuracy Rate [%]	Capability [%] as a Turing Test	Average Accuracy Rate [%]	Capability [%] as a Turing Test
Totally Blind	71	86	81	**100**
Low Vision	67	71	60	**71**
All	69	75	70	**79**

3.2 Experiment on Test's Performance for Real Humans

We evaluate four kinds of tests: Markov-chain phrase test with/without consonant substitution and Topic detection test with/without one. We show our process of the subjective experiment as follows.

1) **Preprocessing.** For each AI problem, **EvaPro** collects phrases from $\mathcal{M}$, analyzes morphology of them, and generates a corpus.
2) **Generate tests.** **EvaPro** generates four kinds of tests. Each one consists of ten AI problems and includes the alternative answer choices, which consists of four items. Finally, **EvaPro** outputs them as a text. Note that this process is completely automated, that is, we do not modify anything in the text without appending several explanations.
3) **Distribute tests.** 24 subjects participate the experiment. Seven subjects of them are the totally blind, recognize the tests by their auditory sense with a screen reader. Seventeen subjects are the weak-sighted, recognize them by sight (and hearing, partly).
4) **Check results.** The subjects answer the tests without any training. We check them as follows.

    Average accuracy rate. This is an average of the rate how subjects correctly answer every a problem.

    Capability as a Turing test. The rate is that subjects correctly answer at least t times among N problems. In this case, we set that $(N, t) = (10, 7)$.

Note that we determine a threshold t to the following effect. We assume that a verifier judges a prover as a human by at least t correct answers among N problems. If there exists an adversary $\mathcal{P}^*$ who solves an AI problem by the probability of P, the success probability of brute-force attacks to the test by $\mathcal{P}^*$ is $\sum_{i=t}^{N} \binom{N}{i} P^i (1-P)^{N-i}$. Bursztein et al. [19] claim that a CAPTCHA system is unuseful if there exists an adversary who solves it by the probability of 1% more than. Hence, we set $t = 7$ corresponding to $N = 10$ and $P = 1/p = 0.25$.

Results and Discussion. The results appear in Tab. 1. (Data from out previous work [20].) The rate of Markov-chain phrase test is lower than one of topic detection test. In my understanding, the reason why differences of naturalness between natural phrases and word salad becomes small is that phrases with consonant substitution are more unnatural than without ones. In fact, an influence of the consonant substitution in Markov-chain phrase test is significant by our statistical test but is not significant in topic detection test.

To compare our proposal with a conventional system, we also show our results [20] of experiment for Google's audio CAPTCHA of the version at November 2013. The capability as a Turing test was only 7%. Moreover, there were only five among 24 Japanese who can answer correctly at least one in ten challenges.

Consequently, our system is better suited for the visually-impaired.

3.3 Experiment on Performance against Cracking

We consider adversaries with search engines and generation rate of new phrases.

Attacks Using Search Engines. Straightforward attack is that adversaries convert documents on the Internet into ones written in hiragana and they seek source documents of our problems. However, this approach needs vast amounts of memories. Thus, we employ a typically and easily-implemented attack as follows: suppose that there are phrases written in hiragana and their consonants are swapped, adversaries repair several "mistakes" in the phrases with conventional tools, and convert them into the ones which includes several kanji-characters.

We show the following experiment as such an attack.

1) Extract 100 natural phrases from $\mathcal{M}$ and swap their consonants in the same manner as Sec. 2.5.
2) Convert them into sound files with Microsoft Speech Platform 11. They are 48KHz-16bit mp3 files in stereo.
3) Convert them into text files with Dragon Speech 11, which is software of speech recognition. Conversion to kanji-characters and restoration are done in this process.
4) For each one, we check whether it includes a source of the corresponding substituted phrase to less than top ten affairs. If it includes, we consider that adversaries find the source of the phrase. By checking all the results of the samples, we calculate a detection rate against attacks by search engines.

Generation Rate of New Phrases. Let $\mathcal{Z}$ be a multiple set, whose element is a generated phrase in each test. We call $z \in \mathcal{Z}$ new phrase if $z' \notin \mathcal{Z} \backslash z$, where $z' = z$. For each test, we generate it 10000 times under the same condition of the subjective experiment, assign generated phrases to $\mathcal{Z}$, and check they are new phrases or not.

Results and Discussion. Attacks Using Search Engines. The detection rate by Yahoo! is 9% under the condition of $(r_L, r_H) = (2, 2)$ and 1% under the condition of $(r_L, r_H) = (2, 5)$. In our subjective experiment, the number of choices by a problem is four, thus the success probability of brute-force attacks is 25%. The four sums of detection rate, that is $1 \times 4\%$, is lower than this probability. Therefore, an advantage of adversaries does not increase by such attacks.

Generation Rate of New Phrases. We collected phrases of 1717 lines, 87260 letters, and 4361 kinds of morphemes in Markov-chain phrase test. Moreover, we collected phrases of 1209 lines, 109042 letters, and 9726 kinds of morphemes in topic detection test. The rate of Markov-chain phrase test and topic detection test is 99.94% and 99.65%, respectively.

Each test generated new phrases with a small amount of documents at a high rate. Therefore, we have only to collect materials for test at any time in case of update. This means that our system can dispose old phrases and generate new ones.

4 Conclusion

We constructed an accessible CAPTCHA system, which generates two type of AI problems using the contextual cognition. We showed that our system could differentiate humans from software agents with high probability compared with conventional Google's audio CAPTCHA, generate new phrases as problems without limit by using phrases on the Internet, and have a tolerant against adversaries by existing search engines.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
2. Bigham, J.P., Cavender, A.C.: Evaluating existing audio captchas and an interface optimized for non-visual use. In: CHI 2009, pp. 1829–1838. ACM (2009)
3. Bursztein, E., Bethard, S., Fabry, C., Mitchell, J.C., Jurafsky, D.: How good are humans at solving captchas? a large scale evaluation. In: SP 2010, pp. 399–413. IEEE Computer Society (2010)
4. Shirali-Shahreza, S., Shirali-Shahreza, M.H.: Accessibility of captcha methods. In: AISec 2011, pp. 109–110. ACM (2011)
5. Yamaguchi, M.: A generating method of accessible verbal questions with online documents for substitution of captcha-like tests. Verbal Interface & Interaction 15(4), 337–352 (2013) (Japanese)
6. Holman, J., Lazar, J., Feng, J.H., D'Arcy, J.: Developing usable captchas for blind users. In: Assets 2007, pp. 245–246. ACM (2007)
7. Wimmer, H., Perner, J.: Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. Cognition 13(1), 103 (1983)
8. Dick, P.K.: Do androids dream of electric sheep? (1968)
9. Yamamoto, T., Tygar, J., Nishigaki, M.: Captcha using strangeness in machine translation. AINA 2010, 430–437 (2010)
10. Kamoshida, Y., Kikuchi, H.: Word salad captcha - application and evaluation of synthesized sentences. NBIS 12, 799–804 (2012)
11. Miller, G.A., Licklider, J.C.R.: The intelligibility of interrupted speech (1950)
12. Rawlinson, G.: The Significance of Letter Position in Word Recognition. University of Nottingham (1976)
13. Saberi, K., Perrott, D.: Cognitive restoration of reversed speech. Nature 398(6730), 760 (1999)
14. Kudo, T.: Mecab: Yet another part-of-speech and morphological analyzer, `http://mecab.sourceforge.net/`
15. Kudo, T., Matsumoto, Y.: Japanese dependency analysis using cascaded chunking. In: CoNLL 2002, pp. 63–69 (2002)
16. Weblio-Thesaurus, `http://thesaurus.weblio.jp/`
17. Aozora-Bunko, `http://www.aozora.gr.jp/`
18. GAPI. Net-0.5.0.1, `http://gapidotnet.codeplex.com/`
19. Bursztein, E., Martin, M., Mitchell, J.: Text-based captcha strengths and weaknesses. In: CCS 2011, pp. 125–138. ACM (2011)
20. Yamaguchi, M., Okamoto, T.: An evaluation of a captcha for the visually impaired using questions of context interpretation. National University Corporation Tsukuba University of Technology Techno Report 21 (2014) (Japanese)