INTERNATIONAL JOURNAL OF ENHANCED RESEARCH IN MANAGEMENT AND COMPUTER APPLICATIONS
VOL. 2, ISSUE 3, MARCH 2013 ISSN NO: 2319-7471

Audio Based Recaptcha

[1]Nishant Mehta, [2]Swapnil Shinde
Dept. of Computer Engineering, G.H.R.I.E.T Pune, India
[1]nishantmehta0708@gmail.com, [2]swpnlshnd@gmail.com

Abstract: The twenty-first century is filled with many new gadgets and technological innovations. The society is getting digitalized with every passing hour. Various speech to text converters are digitalizing the audio files but the main obstacles is noise which halts the progress of the converters. Another important thing is that they can't recognize accents of all the people. So the efficiency of these programs is not considerably high. They can never overcome the human ability to recognize the audio so the proposed system provides a way to utilize the human capability and also provide security against BOTs. The main aim of this paper is to enhance the concept of Recaptcha and apply it to the audio files. The current Recaptcha module is helping the world by digitalizing the text and this project will aim to apply to audio files and digitalize them.

Keywords: CAPTCHA, ReCAPTCHA, audio files, digitalization.

I. Introduction To Captcha

A CAPTCHAs are automated tests designed to tell computers and humans apart by presenting users with a problem that humans can solve but current computer programs cannot. Because CAPTCHAs can distinguish between humans and computers with high probability, they are used for many different security applications: they prevent bots from voting continuously in online polls, automatically registering for millions of spam email accounts, automatically purchasing tickets to buy out an event, etc. Once a CAPTCHA is broken (i.e., computer programs can successfully pass the test), bots can impersonate humans and gain access to services that they should not. Therefore, it is important for CAPTCHAs to be secure.

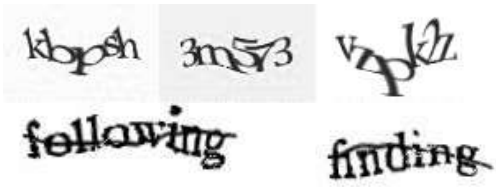

Fig. 1 Modern Captcha

II. Introduction To Recaptcha

Physical books and other texts written before the computer age are currently being digitized (e.g., by the Google Books Project and the nonprofit Internet Archive) to preserve human knowledge and to make information more accessible to the world. The pages are photographically scanned and the resulting bitmap images are transformed into text files by optical character recognition (OCR) software.

Fig. 2 Recaptcha

This transformation into text is useful because the books can then be indexed, searched, and stored in a format that can be easily analyzed and manipulated. One of the stumbling blocks in the digitization process is that OCR is far from perfect at deciphering the words in bitmap images of scanned texts.

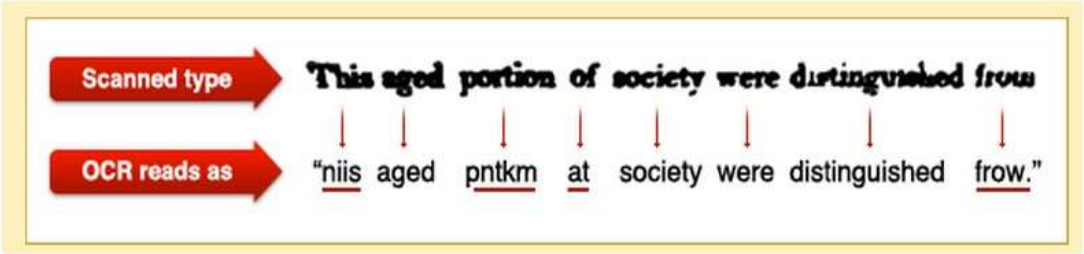

Fig.3 Characters recognized by OCR

By contrast, humans are more accurate at transcribing such print. For example, two humans using the "key and verify" technique, where each types the text independently and then any discrepancies are identified, can achieve more than 99%accuracy at the word level (4, 5). Unfortunately, human transcribers are expensive, so only documents of extreme importance are manually transcribed. The apparatus, called "reCAPTCHA," is used by more than 40,000 Web sites and demonstrates that old print material can be transcribed, word by word, by having people solve CAPTCHAs throughout the World Wide Web. Whereas standard CAPTCHAs display images of random characters rendered by a computer, reCAPTCHA displays words taken from scanned texts. The solutions entered by humans are used to improve the digitization process. To increase efficiency and security, only the words that automated OCR programs cannot recognize are sent to humans. However, to meet the goal of a CAPTCHA (differentiating between humans and computers), the system needs to be able to verify the user's answer. To do this, reCAPTCHA gives the user two words, the one for which the answer is not known and a second "control" word for which the answer is known. If users correctly type the control word, the system assumes they are human and gains confidence that they also typed the other word correctly. We start with an image of a scanned page. Two different OCR programs analyze the image; their respective outputs are then aligned with each other by standard string matching algorithms and compared to each other and to an English dictionary. Any word that is deciphered differently by both OCR programs or that is not in the English dictionary is marked as "suspicious". These are typically the words that the OCR programs failed to decipher correctly. According to our analysis, about 96% of these suspicious words are recognized incorrectly by at least one of the OCR programs; conversely, 99.74% of the words not marked as suspicious are deciphered correctly by both programs. Each suspicious word is then placed in an image along with another word for which the answer is already known, the two words are distorted further to ensure that automated programs cannot decipher them, and the resulting image is used as a CAPTCHA. Users are asked to type both words correctly before being allowed through. It refers to the word whose answer is already known as the "control word" and to the new word as the "unknown word." Each reCAPTCHA challenge, then, has an unknown word and a control word, presented in random order. To lower the probability of automated programs randomly guessing the correct answer, the control words are normalized in frequency; for example, the more common word "today" and the less common word "abridged" have the same probability of being served. The vocabulary of control words contains more than 100,000 items, so a program that randomly guesses a word would only succeed 1/100,000 of the time. Additionally, only words that both OCR programs failed to recognize are used as control words. Thus, any program that can recognize these words with no negligible probability would represent an improvement over state-of-the-art OCR programs.

III. Introduction To Audio Captcha

Current CAPTCHAs rely on superior human perception, leading to CAPTCHAs that are predominately visual and, therefore, unsolvable by people with vision impairments. Audio CAPTCHAs that rely instead on human audio perception were introduced as a non-visual alternative but are much more difficult for web users to solve. Part of the problem is that the interface has not been designed for non-visual use.

Most CAPTCHAs on the web today exhibit the following pattern: the solver is presented text that has been obfuscated in some way and is asked to type the original text into an answer box. The technique for obfuscation is chosen such that it is difficult for automated agents to recover the original text but humans should be able to do so easily. Visually this most often means that graphic text is displayed with distorted characters. In audio CAPTCHAs, this often means text is synthesized and mixed in with background noise, such as music or unidentifiable chatter. Although the two types of CAPTCHAs seem roughly analogous, the usability of the two types of CAPTCHAs is quite different because of inherent differences in the interfaces used to perceive and answer them.

Visual CAPTCHAs are perceived as a whole and can be viewed even when focus is on the answer box. Once focusing the answer box, solvers can continue to look at visual CAPTCHAs, edit the answer that they provided, and verify their answer. They can repeat this process until satisfied without pressing any keys other than those that form their answer. Errors primarily arise from CAPTCHAs that are obfuscated too much or from careless solvers. Audio playback is linear. A solver of an audio CAPTCHA first plays the CAPTCHA and then quickly focuses the answer box to provide their answer.

For sighted solvers, focusing the answer box involves a single click of the mouse, but for blind solvers, focusing the answer box requires navigating with the keyboard using audio output from a screen reader. Solving audio CAPTCHAs is difficult, especially when using a screen reader.

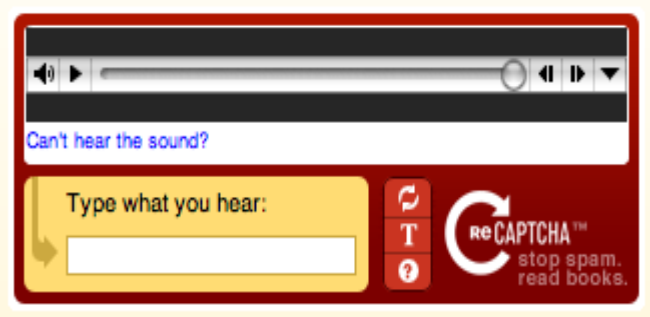

Fig. 4 Audio Captcha

Screen readers voice user interfaces that have been designed for visual display, enabling blind people to access and use standard computers. Screen readers often speak over playing CAPTCHAs as solvers navigate to the answer box, speaking the interface but also talking over the CAPTCHA. A playing CAPTCHA will not pause for solvers as they type their answer or deliberate about what they heard. Reviewing an audio CAPTCHA is cumbersome, often requiring the user to start again from the beginning, and replaying an audio CAPTCHA requires solvers to navigate away from the answer box in order to access the controls of the audio player. The interface to audio CAPTCHAs was not designed for helping blind users solve them non-visually.

IV. Introduction to Audio Based Recaptcha

The Audio Based ReCAPTCHA has mixture of essence of Captcha, ReCAPTCHA and Audio Captcha. It extends the functionality of Recaptcha and Audio Captcha. The ReCAPTCHA nowadays are used for digitalizing the books which are difficult for Optical character recognizer to recognize, since the human has more power of interpretation than that of OCR, similarly the Audio based Recaptcha will digitalize the audio. The drawbacks of current speech to text converter are that they cannot distinguish the accent of the voice , such as the word "schedule" spoken by U.S accent person varies from that of Indian accent person. So many times it happens that the text converter cannot recognize the voice or interpret it and the other problem is that the audio files which are filled with noise are not converted with a high level of accuracy. But we humans can understand the audio even with a considerable degree of noise. . So what if human's fine hearing capability is utilised to do so.
Following image (see Fig. 4) shows us a glimpse of Audio based recaptcha.

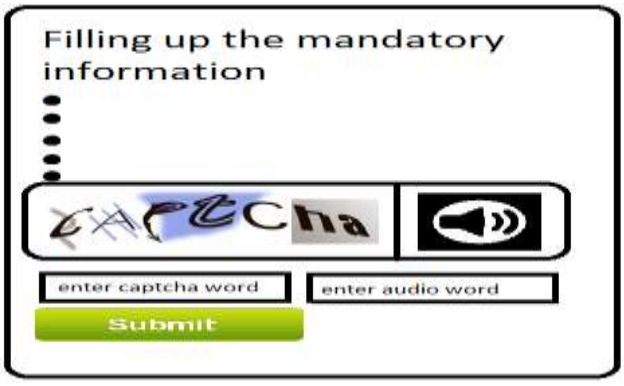

Fig. 5 Audio Based Recaptcha

When user fills out the form he will get a image CAPTCHA which will be for the security measure. In the other box he will hear an audio file and type out what he hears. The data will be stored and mapped to the audio file. Now based on the previous text associated with the audio file the user will be shown a suggestion box. The system will have a dictionary feed as well because we want to cut down on the inefficiency caused by the spelling mistakes. Take an example if the user wants to type Hello and ends up typing hell they both are different words for sure. At the backend the text processing will take place and if a particular text passes a certain amount of occurrences then it will be finalized. Markers will be used to denote the strength of the word. The conversion of audio by available converters is not totally accurate. We make use of human capability to convert the audio files to text. Humans are able to recognize the voice even if spoken in a particular accent. Along with the provided security the users will be digitalizing the audio files.

A. Ranking Function

The Ranking function is the engine of or rather the heart of Audio Based ReCAPTCHA where ranks are allotted to different keys of specific audio samples. The two main databases which are involved in here are :

1) Audiodb Table
2) Suggest Table
3) Range Table
4) Random Table

Now let's look at the structural schema of these database tables

#	Name	Type	Collation	Attributes	Null	Default
1	AID	varchar(10)	latin1_swedish_ci		No	None
2	ASAMPLE	longblob			No	None
3	FLAG	int(2)			No	0
4	KEY1	varchar(20)	latin1_swedish_ci		No	####
5	KEY2	varchar(20)	latin1_swedish_ci		No	####
6	KEY3	varchar(20)	latin1_swedish_ci		No	####
7	KEY4	varchar(20)	latin1_swedish_ci		No	####
8	KEY5	varchar(20)	latin1_swedish_ci		No	####
9	PROB1	int(11)			No	0
10	PROB2	int(11)			No	0
11	PROB3	int(11)			No	0
12	PROB4	int(11)			No	0
13	PROB5	int(11)			No	0
14	NEWP	int(11)			No	0
15	COUNT	int(11)			No	0

Fig. 6 Structure of Audiodb Table

#	Name	Type	Collation	Attributes	Null	Default
1	AID	int(3)			No	None
2	SUGGEST	varchar(20)	latin1_swedish_ci		No	None
3	SVAL	float			No	None

Fig. 7 Structure of Suggest Table

#	Name	Type	Collation	Attributes	Null	Default
1	AID_NO	int(3)			No	None
2	NAME	varchar(20)	latin1_swedish_ci		No	None
3	START	int(3)			No	None
4	END	int(3)			No	None

Fig 8 Structure of Range Table

#	Name	Type	Collation	Attributes	Null	Default
1	RID	int(3)			No	None

Fig . 9 Structure of Random Table

1. Creating Database Tables

Table Audiodb is the core of the database, where the text processing is done. Following are brief details of each fields in the table:

- AID
- ASAMPLE
- FLAG
- KEY1
- KEY2
- KEY3
- KEY4
- KEY5
- PROB1
- PROB2
- PROB3
- PROB4
- PROB5
- NEWP
- COUNT

a) **AID:** An AID (Audio identifier) is the unique identifier for audio samples that are going to get stored in the database. It is set as a primary key to facilitate various operations on the database.
b) **ASAMPLE:** An ASAMPLE (Audio sample) is the path o the audio file.
c) **FLAG:** A flag is the counter that monitors the insertion of the keys for the specific ASAMPLE .
d) **KEY1 to KEY5**: The values entered by the end-user are stored in these keys (1-5). The purpose of using only 5 values will be seen in the later section of the document.
e) **PROB1 to PROB5:** PROB (i) computes the repetition of occurrence of KEY (i). Depending on this PROB (i) value ranking algorithm will manipulate the ranking of the audio sample.
f) **COUNT:** The total number of attempts to enter the keys values. These attempts are limited to an extent, and the reason is explained in the later section of the document.
g) **NEWP:** The NEWP is the chance given for the new key value, which is the value not present in the KEY1 to KEY5.

Ranking Algorithm requires another table which can give grades to the sample files so the table SUGGEST serve this purpose. Following are brief details of each fields in the table.

- AID
- SUGGEST
- SVAL

a) **AID:** It references the AID of Audiodb table.
b) **SUGGEST:** Stores the top ranked KEY (i) .
c) **SVAL:** value that helps in assigning the colour code.

Before and after the required processing to be done by Ranking Algorithm, there is a need to maintain the records of range of audio files. Each row stores the range of the audio samples which later assist in randomly allocating the files to the end user. Another use of this table is that when text generation function is invoked, a specified range of audio samples can be converted. Thus it provides flexibility with the text conversion. Following are brief details of each fields in the table.

- AID_NO
- NAME
- START
- END

a) **AID_NO:** Uniquely identifies the range of audio files.
b) **NAME:** Name given to the specific set of samples.
c) **START:** Starting AID of range of Audio samples.
d) **END:** Ending AID of range of Audio samples.

The purpose of Table Random is for randomization of allocation audio samples to the users. It has only one field RID which randomize the range of audio samples present in Range Table.

a) Structure of databases

1) Default table looks likes Table I
2) The very first insertion in the table of sample < a > in Table II
3) The series of insertion of audio samples on table Audiodb in Table III
4) The flag value only changes if there is a new word associated with the audio file. Possible values for flag- 0,1,2,3,4,5,-1

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
-	-	0	####	####	####	####	####	0	0	0	0	0	0	0

Table I: Default table

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	0	####	####	####	####	####	0	0	0	0	0	0	0

Table II: The very first insertion in the table of sample < a >

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	0	####	####	####	####	####	0	0	0	0	0	0	0
22	< b >	0	####	####	####	####	####	0	0	0	0	0	0	0
33	< c >	0	####	####	####	####	####	0	0	0	0	0	0	0
...	< ... >	0	####	####	####	####	####	0	0	0	0	0	0	0
n	< N >	0	####	####	####	####	####	0	0	0	0	0	0	0

Table III: The series of insertion of audio samples on table Audiodb

b) Deploying at the user interface sites

1) When the first user hears the audio sample (AID = 11) and he enters a word "**hello**".
The overall changes that happen in the table (see Table IV) are explained below:

a. FLAG is incremented by 1.
b. KEY1 is set to value "**hello**".
c. PROB1 which points to the repetition of occurrence of the KEY1 is incremented to 1.
d. COUNT counter is incremented to 1.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	1	Hello	####	####	####	####	1	0	0	0	0	0	1
22	< b >	0	####	####	####	####	####	0	0	0	0	0	0	0
33	< c >	0	####	####	####	####	####	0	0	0	0	0	0	0
...	< ... >	0	####	####	####	####	####	0	0	0	0	0	0	0
n	< N >	0	####	####	####	####	####	0	0	0	0	0	0	0

Table IV Audiodb Table

2) After randomization some other audio sample is rendered (say AID=22).The user enters a word "**good**". The overall changes which happen in the table (see Table V) are explained below:

a. FLAG is incremented by 1.
b. KEY1 is set to value "**good**".
c. PROB1 which points to the repetition of occurrence of the KEY1 is incremented to 1.
d. COUNT counter is incremented to 1.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	1	Hello	####	####	####	####	1	0	0	0	0	0	1
22	< b >	1	Good	####	####	####	####	1	0	0	0	0	0	1
33	< c >	0	####	####	####	####	####	0	0	0	0	0	0	0
...	< ... >	0	####	####	####	####	####	0	0	0	0	0	0	0
n	< N >	0	####	####	####	####	####	0	0	0	0	0	0	0

Table V Audiodb Table

Step 3) After randomization the sample whose AID = 11 is rendered. The user enters a word "**helo**". The overall changes that happens in the table (see Table VI) are explained below:-

a. FLAG is incremented to 2.
b. KEY2 is set to value "**helo**" (as KEY1 is occupied by "**hello**").
c. PROB2 which points to the repetition of occurrence of the KEY2 is incremented to 1.
d. COUNT counter is incremented to 2.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	2	Hello	helo	####	####	####	1	1	0	0	0	0	2
22	< b >	1	Good	####	####	####	####	1	0	0	0	0	0	1
33	< c >	0	####	####	####	####	####	0	0	0	0	0	0	0
...	< ... >	0	####	####	####	####	####	0	0	0	0	0	0	0
n	< N >	0	####	####	####	####	####	0	0	0	0	0	0	0

Table VI Audiodb Table

Step 4) Similarly in this way the various samples are rendered and the scenario which is aroused let's see it for only sample AID=11 in Table VII.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	Heno	4	3	3	2	1	0	13

Table VII Audiodb Table

c) Deciding when to stop rendering the Audio sample

Explanation: When the FLAG field becomes -1 then that audio sample is stopped from being rendered. The purpose of using this FLAG is to prevent indefinitely rendering of a audio sample. Let's see how it works.
The FLAG field can be set to -1 in the following cases:-

Case 1) If any of the PROB(i) hits to value 5 (see Table VIII)

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	-1	Hello	helo	Halo	hallo	Heno	5	3	3	2	1	0	14

Table VIII Audiodb Table

Case 2) If the COUNT hits to value 25 (see Table IX)

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	-1	Hello	helo	Halo	hallo	heno	4	4	4	3	1	3	25

Table IX Audiodb Table

When the above such situation arises then **NEWP** field of the table comes into picture. Let's see its role in the algorithm.

d) Role of NEWP field

NEWP: As mentioned in the introductory part of the Audiodb table, the NEWP is the chance given for the new key value, which is the value that currently not present in the KEY1 to KEY5.That means NEWP is the value which comes into picture when all KEY1 to KEY5 are filled up. Depending on the condition given below it is decided whether to entertain it or not:

Setup =>

- Scan through database and find key with lowest probability.
- Use FILO (first In Last Out) manner in case of conflict.
- Now the key found should be swapped or not??

the key found could get more hits later, we cannot allow every new player.

Steps to follow:

- Find key with lowest prob
- If prob <=2 then allow new player.
  else discard new player and set newp=5
- Check newp if newp==5 discard new player
- Else allow and increment newp by 1 and discard lowest key value, discard new player and set newp=5

case 1) :Assume that a user enters word "**asd**". There could be chances that the new word is the apt word for the audio sample that user just heard. So its PROB value becomes 1 which is equal to the PROB5 value (see Table X)

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	heno	4	3	3	2	1	0	13

Table X Audiodb Table

In this case the "**asd**" value is replaced at the KEY5 field, and NEWP is incremented to 1. Also not to forget the COUNT value which is incremented by 1.
NOTE: FLAG value isn't changed (see Table XI).

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	asd	4	3	3	2	1	1	14

Table XI Audiodb Table

case 2): Assume the following scenario (see Table XII)

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	heno	4	3	3	1	1	0	12

Table XII Audiodb Table

Now if user enters word "**asd**". There could be chances that the new word is the apt word for the audio sample that user just heard. So its PROB value becomes 1 which is equal to the PROB5 and PROB4 value. Now the question that pops up is which one to replace (KEY4 or KEY5). Solution to this is by LIFO (Last In First Out) method. Since the last entry obviously was of KEY5 hence it is replaced with the "**asd**" NEWP was incremented to 1.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	asd	4	3	3	1	1	1	13

Table XIII Audiodb Table

case 3): The new player can replace the already occupant KEY(i) as long as the KEY(i) has its corresponding PROB(i) value less than 3. In other way the new player will not be entertained if minimum of PROB(1-5) is 3 or more. As the current key in the database has greater probability than the new player (see Table XIV).

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB1	PROB2	PROB3	PROB4	PROB5	NEWP	COUNT
11	< a >	5	Hello	helo	Halo	hallo	heno	4	4	3	4	3	1	19

Table XIV Audiodb Table

Here in the above scenario the new word "**asd**" will not be entertained since the minimum value of PROB(i) is 3 which according to condition has higher probability.

2. Ranking of Audio Samples in the Database

Assume the following:
Audio File contents: "**Hello Good Morning to all**"

Assume the following that we have 5 audio files which contain the following words
< **hello** >,< **good** >,< **morning** >,< **to** >,< **all** >.
We are just considering this for explanation purpose though as per the system the developer may or may not know contents of audio files.
Assume the following table (see Table XV) that is generated after rendering above audio file to the end-users.

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB 1	PROB 2	PROB 3	PROB 4	PROB5	NEWP	COUNT
11	< ***hello*** >	-1	Hello	Helo	hallo	halo	Heno	5	4	2	1	1	0	13
22	< ***good*** >	-1	Good	gud	gude	Good e	Jude	5	4	2	2	2	0	15
33	<***morning***>	-1	Mornni ng	morning	moning	mour ning	Marnin g	4	5	4	4	1	0	18
44	< ***to*** >	-1	Too	two	to	Tu	Toe	4	4	4	4	4	2	25
55	< ***all*** >	-1	Awl	owl	all	Ol	oul	3	3	4	3	1	3	25

Table XV Audiodb Table

Ranking Algorithm requires another table which can give grades to the sample files so the table SUGGEST serve this purpose. Its structure is given in Fig. 10.

#	Name	Type	Collation	Attributes	Null	Default
1	AID	int(3)			No	*None*
2	SUGGEST	varchar(20)	latin1_swedish_ci		No	*None*
3	SVAL	float			No	*None*

Fig 10 Structure of Suggest Table

Ranking of keys which were entered by users can be done by storing the keys that have the maximum PROB(i) values (see Table XVI).

AID	ASAMPLE	FLAG	KEY1	KEY2	KEY3	KEY4	KEY5	PROB 1	PROB 2	PROB 3	PROB 4	PROB5	NEWP	COUNT
11	< ***hello*** >	-1	Hello	helo	hallo	halo	Heno	5	4	2	1	1	0	13
22	< ***good*** >	-1	Good	gud	gude	Good e	Jude	5	4	2	2	2	0	15
33	<***morning***>	-1	Mornni ng	morning	moning	mour ning	Marnin g	4	5	4	4	1	0	18
44	< ***to*** >	-1	Too	two	to	Tu	Toe	4	4	4	4	4	2	25
55	< ***all*** >	-1	Awl	owl	all	Ol	oul	3	3	4	3	1	3	25

Table XVI Audiodb Table

In the case of conflicts such as more than one PROB(i) are having same max value as we can see in the < "**to**"> AID=44 , then such problems are resolved on the basis of first come first serve.

3. Color Code Generation

The main motto of the Audio Based Recaptcha is to digitalize the audio file. This aim is accompanied by colour code method which emphasize on the precision of the text word generated. Each colour has its own significance. The colour coding method is complemented by the SUGGEST table.

Before that SVAL of the SUGGEST table needs to compute which is done by the following way:

Let max (i) is the maximum value in array PROB[j], and count is the value attained by the COUNT field for AID=i.

Here i is the AID of the SUGGEST table and j varies from 1 to 5. Therefore SVAL (i) is given by the following formula:

- SVAL(i)=max(i)/count;
- SVAL=SVAL*10;

The colour code's mapping with the SVAL is given in the following table (see Table XVII).

SVAL Range	HEX COLOR	COLOR
0	#000000	HELLO
0.1 - 1	#FF0000	HELLO
1.1 - 2	#FE2E2E	HELLO
2.1 - 3	#FA5858	HELLO
3.1 - 4	#FA8258	HELLO
4.1 - 5	#FFFF00	HELLO
5.1 - 6	#F4FA58	HELLO
6.1 - 7	#81F781	HELLO
7.1 - 8	#2EFE2E	HELLO
8.1 - 9	#01DF01	HELLO
9.1 - 10	#088A08	HELLO

Table XVII

At the end when a text file is generated with the font color mapped according to the code codes, the beneficiary comes to know that to what extent the text generated is clear-cut one and eventually can help them in numerous ways.

GUI

The proposed Audio based ReCAPTCHA GUI looks like in Fig. 11.

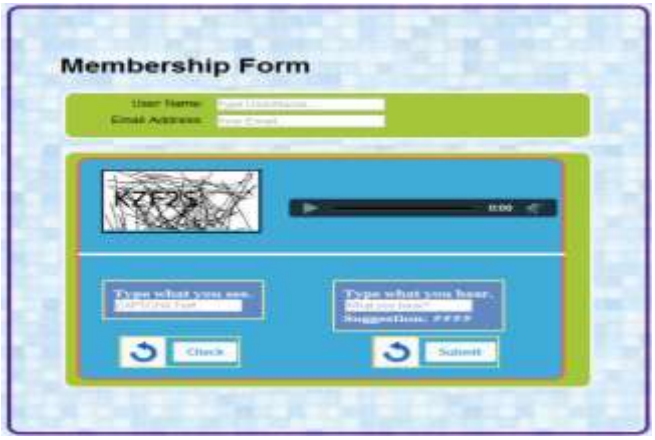

Fig. 11 GUI of Audio Based ReCAPTCHA

Conclusion

This paper has given a broad view of Captcha, Recaptcha and Audio Captcha which are sufficient enough to understand the proposed Audio Based ReCAPTCHA. The paper gives the deep explanation on ranking algorithm on which the audio based ReCAPTCHA runs. This paper tells how we can utilize the human hearing capability to digitalize the audio files.

References

[1]. L. von Ahn, M. Blum, J. Langford, "Telling Humans and Computers Apart Automatically Communications of the ACM", vol. 47, no. 2, pp. 57-60, Feb. 2004.
[2]. Coates, A.L., Baird, H.S., Fateman, R. J. Pessimal, "A Reverse Turing Test", in Proceedings of the International Conference on Document Analysis and Recognition (ICDAR '01), Seattle, WA, 2001, 1154–115.
[3]. Retrieved from: http://www.geek.com/articles/google/audio-captchas-hacked-with-high-success-rate-20081210.
[4]. Retrieved from: http://www.google.com/recaptcha/learnmore.
[5]. Retrieved from: http://www.html5rocks.com/en/.
[6]. Retrieved from: http://docs.webplatform.org/wiki/html.
[7]. Retrieved from: http://www.josscrowcroft.com/projects/motioncaptcha-jquery-plugin.