Verifying Human Users in Speech-Based Interactions

Sajad Shirali-Shahreza, Yashar Ganjali, Ravin Balakrishnan

Department of Computer Science, University of Toronto, Canada

shirali@cs.toronto.edu, yganjali@cs.toronto.edu, ravin@dgp.toronto.edu

Abstract

Verifying that a live human is interacting with an automated speech based system is needed in some applications such as biometric authentication. In this paper, we present a method to verify that the user is human. Simply stated, our method asks the user to repeat a sentence. The reply is analyzed to verify that it is the requested sentence and said by a human, not a speech synthesis system. Our method is taking advantage of both speech synthesizer and speech recognizer limitations to detect computer programs, which is new, and potentially more accessible, way to develop CAPTCHA systems. Using an acoustic model trained on voices of over 1000 users, our system can verify the user's answer with 98% accuracy and with 80% success in distinguishing humans from computers.

Index Terms: Accessibility, CAPTCHA, Speech Recognition, Speech Synthesis

1. Introduction

Computers are becoming more intelligent and can mimic human behavior more naturally. There are some computer programs that aim to simulate human users. These programs are usually created to use services that are designed for human users. CAPTCHA (Completely Automatic Public Turing Test to Tell Computer and Human Apart) systems are a group of methods designed to distinguish between real human users and computer programs who are interacting with the system. Their goal is to ask questions which human users can easily answer, but current computers cannot. A complete survey of CAPTCHA methods and their applications in various domains is presented in [1]. CAPTCHA systems mostly use computer vision limitations to distinguish between human users and computer programs. Audio CAPTCHA is considered an accessible alternative to visual CAPTCHA. A recent user study shows that human users can correctly answer about 40% of the audio CAPTCHA tests on their first attempt; and more than 40% of the tests cannot be correctly answered even after the third attempt [2]. At the same time, specially designed programs can automatically answer more than 58% of the audio CAPTCHA tests [3]. On the whole, it seems that current audio CAPTCHAs are more difficult for humans than computers, and therefore, not good alternatives for visual CAPTCHAs in terms of accessibility.

In this paper, we propose a method to verify whether the user is a human or not. Our key idea is to use synthesized voice imperfections in addition to speech recognition limitations, which is used in current audio CAPTCHA systems, to detect computer programs. Our method plays a sentence for the user and asks him/her to repeat what he/she hears. Then, our system verifies that the response is the requested sentence and said by a human (Figure 1). This is a relatively easy task for human users, but difficult for computers, because it requires speech recognition and speech synthesis. This is the first time that speech synthesizer limitations are used to detect computer programs, which can be used as a new category of audio CAPTCHA systems. Remembering and repeating a sentence could be easier than memorizing and typing a sequence of digits or letters – the de facto in current audio CAPTCHA systems – which suggests our system might be more accessible[1]. This is the key contribution of the paper.

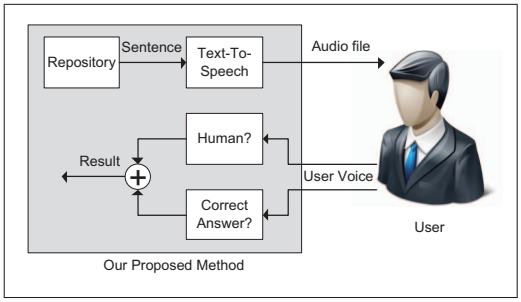

Figure 1: *Overall structure of the proposed method*

Our system can be incorporated in biometric authentication systems that use speech, to verify the "liveness" of user that is the design goal of systems described in [4, 5]. Those systems usually try to use the visual information – such as synchronism between lip movement and the audio – in addition to audio information to prevent *Replay Attack* [6]. As mentioned in [5], they do not verify the sentence that the user said. However, with new speech synthesized systems like [7] that can adapt their output to a specific person's voice, it can be a serious security problem [8], specially if there is no visual information available, such as interactions over the phone. One of the contributions of this paper is showing that the user's response can be verified with high accuracy. For example, with the acoustic model trained with voices of more than 1000 users, our method can verity whether the user said the requested sentence or not with more than 98% accuracy.

The structure of this paper is as follows: in the next section, we describe the overall structure of our system and each component architecture. Section three presents data sets that we created and used to train and test our system. The experimental results of our system are reported in section four. In the last section, we review the main features of our proposed method, provide our plan for future works, and offer a conclusion.

2. System Structure

Our system focuses on computer imperfections in synthesizing human voices to detect computer bots. This is the main idea

[1] Our preliminary pilot study supports this claim. We are in the process a formal user study to compare the accessibility properties of the proposed system with current audio CAPTCHA systems.

behind our work. We ask the user to say a sentence and analyze the response to detect computer generated voices. The overall structure of our method is shown in Figure 1. A sentence is chosen from a repository. A Text-To-Speech (TTS) component is used to convert this sentence into audio format that plays the sentence and asks the user to repeat it. The user's answer goes into two different components: *answer verification* and *group identification*. The goal of the answer verification component is to verify whether the user has said the requested sentence. The group identification component aims to determine whether the user is human or a computer program. These components are described in the following subsections. Their output is combined to produce a final output result. We create fuzzy output instead of binary output, which is more useful for the applications which will use our method. For example, it can be combined with the speaker verification result in an authentication system to create the final decision.

2.1. Answer Verification

The duty of the answer verification component is to decide whether the user has said the requested sentence. Although a general speech recognizer can fulfill our requirements, we do not necessarily need a full speech recognizer. In our method, we want to verify whether the user said what he/she is asked to say or not. So we have a candidate sentence and we can measure how similar the user answer is to our expected sentence.

As shown in Figure 2, the sentence is converted into phonemes using a pronunciation dictionary. At this point, an alignment algorithm aligns the phonemes into voice, using an acoustic model that contains the shapes of all phonemes and is created from a corpus of human voices. After alignment, we can compute the similarity of each part of the user's voice to the corresponding phoneme in the acoustic model. Then, we calculate the average phoneme alignment scores. The logarithm of this number is used as the alignment score of the answer.

This approach has a number of advantages. First, instead of a binary output that simply reports whether the user said the requested sentence, we have a score that shows the similarity between the user's answer and what he/she is asked to say. Second, this process is simpler than normal speech recognition which requires searching a large list of words. Third, we can better handle variations such as different accents or background noise, because in such cases, the alignment score will usually have small changes, while in a normal speech recognition task another word will be chosen as the output.

2.2. Group Identification

In the group identification component, the goal is to distinguish between a human voice and a computer generated – synthesized – voice. The answer verification component prevents cheating by playing a prerecorded human voice, regardless of what the system asks the user to say, which a kind of Replay attack [6].

Because this component aims to identify computer generated voices, it must consider the limitations of Text-To-Speech (TTS) methods. One of the known limitations of concatenative speech synthesizers – which is the common method in current TTS systems – is noticeable artifacts in phoneme conjunctions [9]. So we focus on phoneme conjunctions points to detect computer synthesized voices.

Figure 2 shows how fixed length windows of phoneme conjunctions are created from the waveform alignment. These conjunctions are then used to create feature vectors and classify the user's answer. After performing the alignment, in common with answer verification, we have the phoneme's borderlines. Now we select a fixed length of waveform around each phoneme conjunction point. Each of these fixed length audio samples is used to extract Mel-Frequency Cepstral Coefficient (MFCC) These features are preprocessed, for example their means and variances are normalized. They are preprocessed because different MFCC features have different ranges of values; usually lower features have greater values, and this will bias the classifier. A second reason is that they reduce the effect of the microphone used [10]. After preprocessing, we have a fixed length feature vector for each phoneme conjunction point. A classifier is used to classify each feature vector as human or computer-generated. The ratio of phoneme conjunction points classified as human (a number between 0 and 1) is the output of this component.

3. Datasets

We use two datasets in our work: a dataset of human voices and a dataset of computer synthesized voices.

3.1. Human Dataset

For the human dataset, we need a dataset with samples from a range of users and recorded in different conditions. However, most available speech datasets recorded in a controlled environment by a few users. After reviewing available speech corpuses, we selected the VoxForge[2] data project to create a dataset of human voices. The VoxForge collects Free – GPL license – transcribed speech for different speech processing researches and uses. In June 2010, we downloaded the available English samples. Then we cleaned and processed on the downloaded data. The final dataset has more than 44,000 utterances by more than 1,000 users. Each utterance is one or two short sentences, and there are more than 9,000 sentences. Because users uploaded their voices through the project website, there is a variety of speech samples in the dataset: both native and non-native speakers, speakers with different accents such as American and British, different microphones used to record the samples, and varying amounts of background noise.

This dataset was used to train an acoustic model required for alignment. Considering the variations in the dataset, we can assume that this acoustic model is speaker-independent, microphone-independent, and text-independent. This dataset is also used to train the answer verification and test both components of the system. We will refer to this dataset as the human dataset.

3.2. Computer Dataset

We also needed a dataset of computer generated voice to train and test our system. Considering that there are many different TTS products and each uses special data to train, it was difficult to collect TTS systems and use them to generate samples to create the dataset. Additionally, using the output of systems trained with different voices to train the classifier will result in the creation of a classifier that can distinguish between systems, not between a computer generated voice and a human voice, because there will be considerable difference between the outputs of systems based on the differences between their training data. In addition, commercial products do not necessarily take advantage of state-of-the-art methods.

The Blizzard Challenge[3] evaluates various TTS systems on

[2] http://voxforge.org/

[3] http://festvox.org/blizzard/

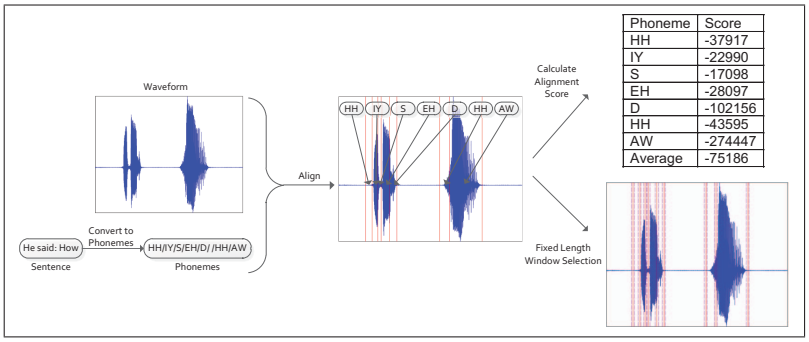

Figure 2: *Waveform alignment with phonemes for answer verification score calculation and group identification feature extraction*

one dataset. Samples of a human voice were provided to researchers developing TTS systems. The researchers used this data to train their systems and synthesize voices that are as similar to the original sample as possible, and maximizing the naturalness of the voice. In 2008 and 2009, the contest released samples that were generated by different groups.

We used the released Blizzard Challenge 2009 data to create a dataset of computer generated voices and will refer to it as the computer dataset. This English section of the Blizzard challenge data contains the samples of the original human voice along with the synthesized output of 18 different groups. We performed some clean up on the data and removed parts that we did not need. Our resulting dataset has 19 groups: one human and 18 TTS systems. About 1,000 sentences are said by each group, which creates a total of more than 16,000 utterances. We use this dataset to train the group identification component and test both component.

4. Experimental Results

In this section, we discuss the tools we developed to implement our method and provide the results for each of the two components described in the previous section. We use C++ in the Linux environment to implement our method. For parts of the MFCC feature extraction and alignment process, we use CMU SPHINX[4] tools and libraries. For the classification part of the group identification component, we use the LIBSVM[5] implementation of Support Vector Machines (SVM).

4.1. Answer Verification

As mentioned previously, we calculate an alignment score for each user's voice based on the sentence that we asked him/her to say. In this section, we want to evaluate whether it is possible to choose a single threshold value to determine whether the user repleated the requested sentence or not.

Assume that alignment score is increased as the similarities between the user's answer and the requested sentence are decreased. Also assume that the answers with alignment scores less than the threshold are marked as correct. We can easily measure the true positive – the ratio of correct answers which we marked as correct – of our system, by counting the number of samples in which the alignment score of user's voice with the utterance transcription is less than the threshold. But we also need a way to measure the false positive of our system: the probability that the user said something other than we asked, and we incorrectly marked it as correct.

To measure this, we must create a series of possible cases in which the user said something other than we asked, and we try to align it to the correct answer. To create this set, we select a subset of each dataset. For the human dataset, we select the top 100 sentences that were said by most users. Then, for each sentence, we randomly choose 30 speakers. For each sample, we consider the case of aligning the sample with the correct sentence – the sentence that the user really said – as the Human-CorrectAnswer, and then the case of aligning the sample with one of the remaining 99 sentences as the Human-IncorrectAnswer. In other word, the Human-CorrectAnswer cases represent when the user said what we asked and we must mark it as correct, and the Human-IncorrectAnswer cases represent when the user said something other than what we asked and we must mark it as incorrect. We refer to this subset as the Human Dataset Subset. The same is done for the computer dataset; we select 78 random sentences, and for each sentence, we have the output of 18 TTS systems plus the original human voice.

Figure 3 shows the results of choosing different thresholds on the ratio of answers in each group that are accepted as correct. As expected, when the threshold is increased, more samples in each case are considered as correct. But the figure shows considerable space betwen when the correct answer series reaches 1 and when the incorrect series starts to rise from 0. For example, with a threshold value of 16.81, more than 98% of correct answers are marked as correct and less than 1% of incorrect answer are marked as correct. These results show that this component performs very well in verifying whether the user said the requested sentence.

4.2. Group Identification

In the previous section, we described how we create the feature vectors from the user's answer. Before we can use the classifier to classify new user's answer, we must train a classifier. To do this, we need a training data that have a series of samples from human voices and a series of samples from computer generated voices. An important note here is that we must try to reduce unwanted effects. For example, while the computer dataset is nearly noise-free, the samples in the human dataset usually has some sort of background noise. So we cannot use a number of samples from human data set and a number of samples from the computer data set and mix them to create training data, because in that case, the classifier learns to ditstinguish between noisy and noise-free voices, not human and synthesized voices.

To avoid such problems, we decided to only use the com-

[4]http://cmusphinx.sourceforge.net/
[5]http://www.csie.ntu.edu.tw/˜cjlin/libsvm/

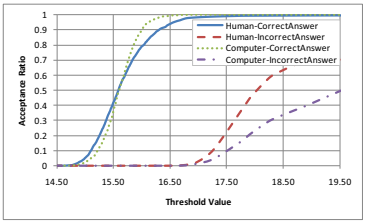

Figure 3: *Answer verification threshold effect*

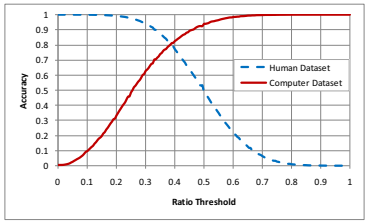

Figure 4: *Effect of human verification threshold on accuracy*

puter dataset, where we have samples of human voice and samples of computer generated voices. A good point about the data set we created is that because the TTS outputs are generated from a model that was trained from the human voice, the differences between TTS outputs and human voice are mainly due to the TTS limitations. So if the classifier can distinguish these changes, we can say that the classifier is distinguishing between a human voice and a computer generated voice.

When the classifier is trained, it is used to classify the class of each phoneme conjunction in the user's answer. The ratio of conjunctions marked as human are counted and considered as the output. If we want to select a single threshold to decide whether the user is human, we should mark all answers where the ratio of the phoneme conjunctions marked as human is higher than the threshold as human.

Due to the limitations for training a SVM model on huge number of samples, we cannot use the entire computer dataset. So we randomly select 1500 samples. We extract the feature vectors as described in the previous section for each sample. There is a second problem that must also be considered. As mentioned, there is one human voice sample and 18 TTS systems. If we use all samples, the number of computer-generated samples will be 18 times greater than the number of human samples, which will result in a biased classifier. So we again randomly select 5000 feature vectors from each group and use them to train the classifier.

To measure the relation between the threshold and the accuracy on human dataset, we use all 18 TTS system samples and the human samples to train the classifier. But we use a cross validation schema to measure the accuracy in the computer dataset, because if we use all the TTS outputs to train the classifier, then it is possible that the trained classifier will identify each individual TTS system instead of distinguishing between human and computer-generated voices. To prevent this, we divide the 18 TTS systems into three random groups, each containing 6 systems, to create a 3-fold cross validation. Then we train the classifier on the two groups and test it on the third group. Then the average of the accuracy for three different runs is selected. Figure 4 shows the results in both the human and the computer dataset. As we see, threshold values between 0.35 and 0.40 provide about 80% accuracy in both datasets.

5. Conclusion

In this paper, we present a system to verify that the user is a human, using the TTS limitations to identify computer programs. Our system asks the user to repeat what he/she hears. Then the user's answer is analyzed to verify that it is the requested sentence and said by a human. We provide the experimental results of our system on two large datasets of human voices and computer synthesized voices. In a binary decision mode, our system can verify the user's answer with more than 98% accuracy and can correctly identify about 80% of user's group – human vs. computer – in each dataset.

Our method can be combined with other authentication or identification systems to provide more security. For example, it can be used by authentication systems as a kind of liveness check to further enhance voice-based authentication. It can also be used in over telephone services to detect bots such as those used by customer supports in companies. Repeating a sentence seems to be easier than entering a series of digits and numbers, which can result in a more usuable and accessible system. Currently we are preparing a formal user study to test this. We are also developing a VOIP spam detection system using this method.

6. References

[1] S. Shirali-Shahreza and M. Shirali-Shahreza, "A survey of human interactive proof systems," *International Journal of Innovative Computing, Information and Control (IJICIC)*, vol. 6, no. 3(A), pp. 855–876, 2010.

[2] J. P. Bigham and A. C. Cavender, "Evaluating existing audio CAPTCHAs and an interface optimized for non-visual use," in *27th Intl. Conf. on Human factors in computing systems - CHI '09*, p. 1829.

[3] J. Tam, J. Simsa, D. Huggins-Daines, L. von Ahn, and M. Blum, "Improving audio CAPTCHAs," in *The Symposium on Accessible Privacy and Security (SOAPS '08)*.

[4] G. Chetty, "Robust audio visual biometric person authentication with liveness verification," in *Intelligent Multimedia Analysis for Security Applications 2010*, pp. 59–78.

[5] N. Eveno and L. Besacier, "A speaker independent "Liveness" test for Audio-Visual biometrics," in *Interspeech 2005*, Lisbon, Portugal, pp. 3081–3084.

[6] H. Bredin, A. Miguel, I. Witten, and G. Chollet, "Detecting replay attacks in audiovisual identity verification," in *2006 International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 1, pp. 621–624.

[7] M. Kurimo, et al., "Personalising speech-to-speech translation in the EMIME project," in *ACL 2010 System Demonstrations*, pp. 48–53.

[8] P. Perrot, G. Aversano, and G. Chollet, "Voice disguise and automatic detection," in *Workshop on Nonlinear Speech Processing, WNSP 2005*, pp. 101–117.

[9] Y. Pantazis and Y. Stylianou, "On the detection of discontinuities in concatenative speech synthesis," in *Workshop on Nonlinear Speech Processing 2005*, pp. 89–100.

[10] M. Shirali-Shahreza and S. Shirali-Shahreza, "Effect of MFCC normalization on vector quantization based speaker identification," in *10th Intl. Symp. on Signal Processing and Information Tech.(ISSPIT 2010)*, pp. 250–253