Reducing the Cost of Breaking Audio CAPTCHAs by Active and Semi-Supervised Learning

Malte Darnstädt
Faculty of Mathematics
Ruhr-Universität Bochum
Bochum, Germany
Email: malte.darnstaedt@rub.de

Hendrik Meutzner, Dorothea Kolossa
Faculty of Electrical Engineering
Ruhr-Universität Bochum
Bochum, Germany
Email: {hendrik.meutzner, dorothea.kolossa}@rub.de

***Abstract*—CAPTCHAs are challenge-response tests that are widely used in the Internet to distinguish human users from machines. In addition to the well-known visual CAPTCHAs, most Internet services also provide an audio-based scheme, e.g., to enable access for visually impaired users. Recent research has shown that most CAPTCHAs are vulnerable as they can be broken by machine learning techniques. However, such automated attacks come at a relatively high cost as they require human experts to create labels for the unlabeled CAPTCHA samples collected from a website in order to train an attacking system.**

In this work we utilize active and semi-supervised learning methods for breaking audio CAPTCHAs. We show that these methods can reduce the labeling costs considerably, resulting in an increased vulnerability of audio CAPTCHAs as automated attacks are rendered even more worthwhile. In addition, our findings give insight into improvements to the design of CAPTCHAs, helping to harden prospective audio CAPTCHA schemes against active learning attacks in the future.

***Keywords*-active learning, semi-supervised learning, audio CAPTCHA, automatic speech recognition**

I. Introduction

CAPTCHAs[1] are widely used in the Internet for distinguishing human users from automated programs to limit the abuse in online services, e.g., automated account creation for sending spam mail. Therefore, CAPTCHAs should be easy to solve by humans, but difficult to break by machines. Audio CAPTCHAs are required for allowing access to visually impaired users and to enable the use of non-graphical devices.

For evaluating the security strength of a CAPTCHA scheme, one can build a system that attempts to solve the CAPTCHAs in an automated manner. Recent investigations [1]–[3] show that most audio CAPTCHAs are insecure as they can be broken by machine learning techniques. In this work, we utilize and improve an attack based on *automatic speech recognition* (ASR) pioneered by Sano, Otsuka and Okuno [3].

Training speech recognition systems, as any machine learning technique, requires labeled data examples; thus to break audio CAPTCHAs by means of ASR, the orthographic transcription of the audio signals must be available. Whereas collecting unlabeled data (i.e., downloading audio CAPTCHAs from a website) is nearly for free, labeling the data is very expensive and time consuming as it requires human interaction.

[1]Completely Automated Public Turing tests to tell Computers and Humans Apart

Therefore we investigate the usefulness of unlabeled data, by means of active and semi-supervised learning, to reduce the amount of necessary labels. We demonstrate that the labeling costs can be reduced considerably while achieving high success rates for breaking audio CAPTCHAs.

Our experiments are conducted on two different data sets, a development and an evaluation set. The development set is given by the Aurora-5 speech corpus [4] that contains thousands of noisy digit sequences. The digit sequences constitute real speech recordings spoken by various speakers. The evaluation set is given by the current version of reCAPTCHA [5] that consists of highly distorted digit sequences where the speech appears to be synthetic. We will show that results on the development set predominantly translate to the evaluation set, while analyzing differences in performance between the two data sets can give insights into the design of CAPTCHAs that are harder to break by active learning.

A. Structure of the Paper

We will first give a quick overview over relevant prior work in Sec. I-B. In Sec. I-C, we will define some vocabulary used in the rest of the paper. In Sec. II, we introduce our four learning algorithms: semi-supervised, active, and two combined learners. The setup and results of our experiments are given in Sec. III. The paper is finished with conclusions and open questions in Sec. IV.

B. Prior Work

While there is much prior work on breaking visual CAPTCHAs (e.g., see [6]–[9]), less focus has been given to audio CAPTCHAs. Chellapilla, Larson, Simard and Czerwinski establish in [7] that most attacks on character-based visual CAPTCHAs employ a two-stage approach: the image is first segmented into the parts where characters are located and then the individual characters are recognized using standard pattern recognition techniques. Chellapilla et al. consider the segmentation phase to be the hard part of the problem. A two-stage approach is also applied successfully to break audio CAPTCHAs by Tam, Simsa, Hyde and von Ahn [1] and Bursztein, Beauxis, Paskov, Perito, Fabry and Mitchell [2].

The viability of ASR to break audio CAPTCHAs was recently demonstrated by Sano, Otsuka and Okuno in [3]. Since ASR can cope with continuous speech, the aforementioned segmentation problem is avoided. For an introduction to ASR we refer the interested reader to Rabiner and Juang [10].

Our semi-supervised learning algorithm is an instance of the *hard expectation-maximization (EM) algorithm* [11], [12], which was previously applied for ASR by Zavaliagkos and Colthurst in [13], for the similar problem of sequence classification using Hidden Markov Models by Zhong in [14] and in the related field of computational linguistics by Spitkovsky, Alshawi, Jurafsky and Manning in [15].

Some remarks about the hard EM algorithm are in order: the well-known *expectation-maximization (EM) algorithm*, introduced by Dempster, Laird and Rubin in [11], iterativley finds a (local) maximum likelihood estimation for models depending on hidden variables, like the unknown labels of unlabeled CAPTCHAs in our case. In each iteration the algorithm maximizes the expected value of the likelihood function with respect to the distribution over the hidden variables conditioned on the model from the last iteration. The *hard* EM algorithm, a term coined by Kearns, Mansour and Ng in [12], is a simplification of standard EM, where the distribution is replaced by the mode, i.e. the most likely value, of the hidden variables. Hard EM is sometimes called *Viterbi EM* (e.g. see [15]); in this paper we will always refer to the Viterbi algorithm for finding the most likely path in a Hidden Markov Model [22] when we use the name 'Viterbi'.

The active learning algorithm that we employ as well as a method of combining active and semi-supervised learning in parallel were originally proposed in the context of ASR by Riccardi and Hakkani-Tür [16]. Similar combinations of active and semi-supervised learning for speech recognition were also investigated by Yu, Varadarajan, Deng and Acero [17].

C. Preliminaries

An *observation* is a representation of an audio signal (by an appropriate sequence of feature vectors), that consists of a variable number of spoken words, which are possibly separated by silence or background noise. The vocabulary is limited to digits between zero and nine and the speech may be either natural or synthetic. The observations may contain additional distortions, e.g., additive noise or reverberation effects. The *label* of an observation is the transcription (including silence and noise) of the audio signal.

A *model* represents a data structure that enables the estimation of labels—together with a measure of confidence for each estimated label—for a given set of observations. We regard an estimated label to be *correct* if its digit sequence is identical to the ground-truth label of the observation disregarding any occurrence of silence or noise.[2] Models used in ASR are often generative, i.e., they describe a distribution over observation-label pairs. The probability of a (fixed) model generating an observation-label pair is called the *likelihood* of that pair.

[2]For breaking reCAPTCHA this measure could be even more generous, as the reCAPTCHA system accepts answers having a Levenshtein distance of one between the user input and the true label of the CAPTCHA.

A *learner* is an algorithm that takes a sample of labeled observations as its input and outputs a model. The sample is drawn according to some unknown distribution (which is not necessarily generated by a model). We measure the error rate of a learner's model according to the same distribution.

When comparing two learners L and L', we call L *better* than L' if L achieves a lower average error rate for any fixed number of labeled words in the sample. Another possible criterion uses the number of labeled observations, which is unsuited for our purpose as observations can vary in length. Clearly, the effort of labeling an observation is proportional to its word count.

We assume that we have access to an ASR system that can learn models using labeled observations and that can estimate labels together with a measure of confidence for unlabeled observations. Please note that our learning algorithms, which are defined in the next section, are oblivious to the implementation of this underlying ASR backend (for a description of the backend used in our experiments see Sec.s II-D and II-E).

II. Algorithms

A. Semi-Supervised Learning

In semi-supervised learning the learner has access to two samples: a labeled sample (which we denote by S_L) and an unlabeled one (denoted by S_U). Both samples are drawn according to the same distribution over the observations, thus we can regard the sample S_U as providing additional information about the distribution to the learner. A detailed introduction to semi-supervised learning is given in [18].

We propose the following semi-supervised learning algorithm, which—as mentioned before—has already been applied in similar settings [13]–[15]:

1) Train initial model M on a (small) randomly drawn labeled sample S_L.
2) Draw an unlabeled sample S_U.
3) Repeat the following steps:
    a) Estimate the labels of S_U using the current model M.
    b) Retrain model M on the labeled set $S_L \cup S_U$.

In this, as in all following iterative learning algorithms, the number of iterations can be either determined by convergence or chosen in advance by the user.

In [13] Zavaliagkos and Colthurst removed the estimated labels with the least confidence scores from S_U before retraining the model; we will use a similar idea for the learning algorithms in section II-C, but instead of merely removing the estimations we will apply active learning to obtain the true labels.

B. Active Learning

An active learner, introduced by Cohn el al. in [19], also has access to an unlabeled sample, but here the learner is allowed to actively request true labels for selected observations.

The following active learner requests labels for those observations which the learner's model is least confident about:

1) Train initial model M on a (small) randomly drawn labeled sample S_L.
2) Repeat the following steps:
   a) Draw an unlabeled sample S_U and estimate its labels using the current model M.
   b) Request the true labels of the n observations from S_U with the least confidence and add them to S_L.
   c) Retrain model M on the labeled set S_L.

The difference between our method and the active learner from Riccardi and Hakkani-Tür [16] is that the latter one uses one pool of unlabeled observations, while we draw a new sample S_U in every iteration. Moreover, we use a different measure of confidence in our experiments than Riccardi and Hakkani-Tür (see Sec. II-E).

As noticed in [16], the choice of n is a trade-off between making optimal selection decisions (n should be small) and computational cost (n should be large).

C. Combined Learners

Since active learning algorithms typically do not request labels for all drawn unlabeled observations, it is a natural idea to use these leftovers for semi-supervised learning.

We study two methods to combine the active learner from Sec. II-B with the semi-supervised learner from Sec. II-A.

Our first combined learner was proposed by Riccardi and Hakkani-Tür in [16]—apart from the parts marked as 'optional'—and operates in *parallel*, i.e., in each iteration of the active learner we use the spare unlabeled data for semi-supervised learning:

1) Train initial model M on a (small) randomly drawn labeled sample S_L.
2) Set S_U to the empty set.
3) Repeat the following steps:
   a) Draw an unlabeled sample S'_U and estimate its labels using the current model M.
   b) Request the true labels of the n observations of S'_U with the least confidence and add them to S_L. Add the remaining observations of S'_U with their estimated labels to S_U.
   c) Retrain model M on the labeled set $S_L \cup S_U$ (*optional:* give S_L a higher weight than S_U)[3].
   d) *Optional:* Iterate the semi-supervised learning steps, i.e., estimate the labels of S_U and retrain M on $S_L \cup S_U$ several times.

During our experiments we had to include weighting and additional semi-supervised steps to achieve satisfactory results. The benefit of additional semi-supervised learning steps was previously recognized by Yu, Varadarajan, Deng and Acero [17].

For our second combined method, we propose to run the active and semi-supervised learners in a *serial* fashion:

1) Train model M using the active learner from Sec. II-B.
2) Use M as the initial model for the semi-supervised learner from Sec. II-A and skip the first step.

In spite of the simplicity of this "folklore" algorithm, we are not aware of prior applications in the field of automatic speech recognition.

The serial combined learner is computationally simpler than the parallel one, especially if the optional additional semi-supervised learning steps are executed. For a discussion of the parallel learner's practicability see Sec. III-B5.

D. Speech Recognition Backend

For the underlying ASR backend we employ the *Hidden Markov Model Toolkit* (HTK) [20], which provides a state-of-the-art implementation of the required training and recognition algorithms used for our approach.

We use the Baum-Welch algorithm [21] to train models given a labeled set of observations, and the estimation of labels—given the trained models—is based on the Viterbi algorithm [22]. Due to space constrains we omit the details of these algorithms and refer the interested reader to Rabiner and Juang [10]. HTK's implementation of the Viterbi algorithm, based on a token passing model, provides a list of the n most likely labels together their corresponding word likelihoods, and the overall likelihood of each label can be simply computed as the product of the individual word likelihoods.

E. Measures of Confidence

Let s be an unlabeled observation and let $(l_1, \vec{p}_1)$, $\ldots, (l_n, \vec{p}_n)$ be the output of the Viterbi algorithm using model M, i.e., l_i are the labels estimated by M for s and $\vec{p}_i$ are the associated word likelihood vectors (including likelihoods for parts consisting of silence or noise). We assume that the labels are ordered by their likelihoods.[4] We define the *normalized likelihood* q_i of $(l_i, \vec{p}_i)$ as

$$q_i := \left(\prod_{k=1}^{|l_i|} p_{i,k}^{\frac{1}{|w_{i,k}|}} \right)^{\frac{1}{|l_i|}}$$

where $p_{i,k}$ is the likelihood of the k-th word of label l_i, $|w_{i,k}|$ is its length (in frames) and $|l_i|$ is the word count of l_i.

While we always take l_1 to be the estimated label of s for semi-supervised learning, we employ these two methods to measure the confidence in l_1 for active learning:

A) Take the *normalized likelihood* q_1 of $(l_1, \vec{p}_1)$.
B) Let i be the first index such that l_1 and l_i differ in more than just the positions of silence/noise. Compute the confidence as the ratio $q_1 \,/\, q_i$. We call this quantity the *likelihood ratio*.

We give the following intuition why the normalized likelihood is a better measure of confidence than the plain likelihood: normalizing by $1/|w_{1,k}|$ counteracts against choosing observations with predominantly long words. The normalization by $1/|l_1|$ prevents the learner from selecting observations

[3]Integer weights can easily be implemented by duplicating the observations of S_L in the input of the training algorithm.

[4]i.e., it holds $\prod_{k=1}^{|l_i|} p_{i,k} \geq \prod_{k=1}^{|l_j|} p_{j,k}$ for $i < j$.

with a high word count, which would be a good strategy, in fact, if the effort of labeling was not directly dependent on the word count.

Using the likelihood ratio instead of the normalized likelihood is helpful in the following situation: it can happen that the model M is still unsure about the position of silence/noise in s, while the digits are already very certain; let us say the digits are in the same order in $l_1, \dots, l_m$ for some m. Let s' be a second observation where M has a low level of certainty about the order of digits: let $(l'_1, \vec{p_1}'), \dots, (l'_n, \vec{p_n}')$ be the output of the Viterbi algorithm for s' and let us say that already l'_1 and l'_2 differ in their digits and $q'_1 \approx q'_2$. It can occur in this case that q_1 is smaller than q'_1, so the active learner based on the normalized likelihood would request the label of s first. But we would like the learner to request the true label of s' instead of s, because we are only interested in the order of digits when breaking CAPTCHAs. Taking the quotient can mitigate this problem, since typically q_{m+1} is much smaller than q_1, or it may even hold that $m = n$, in which case we consider the likelihood quotient to be infinite.

For example, during our experiments the Viterbi algorithm estimated the following labels for an observation with the true label "noise-7-2-6-9-noise": "noise-7-2-6-9-noise", "noise-noise-7-2-6-9-noise", "noise-7-2-6-9-noise-noise", etc. We computed the sixteen most likely labels and all of them were correct, i.e. only differing in the position and number of noise segments. However, the normalized likelihood q_1 was lower than the normalized likelihood of other, still erroneously labeled observations.

III. Experiments

A. Data Sets

We use two different data sets: Aurora-5 [4] and a sample from Google's current audio CAPTCHA scheme (reCAPTCHA) [5].

The Aurora-5 data set contains real speech recordings that are mixed with natural background noise (e.g., airport, car engine or office noise) at different *signal-to-noise ratios* (SNRs). The speech recordings were obtained from different speakers (approx. 50 males and 50 females), each pronouncing several observations that comprise a sequence of digits and the number of digits per observation is varied between one and seven. The data set is provided with the respective labels for each observation. Our experiments are based on the noisy digit recordings distorted by the *interior noise* at 10 dB SNR.

We find that using Aurora-5 for system development is advantageous as the recordings are not only similar to digit-based audio CAPTCHAs but they offer the advantage of having access to a large pool of 9275 correctly labeled observations with 50554 words.

The reCAPTCHA data set is also constructed from a sequence of digits where the total number of digits per observation is varied between 6 and 12. The digits are spoken by both a male and a female voice and the speech appears synthetic. All signals exhibit the same stationary background noise. The CAPTCHAs were downloaded from the reCAPTCHA website in March 2014 and manually labeled at our institute. To obtain the CAPTCHA labels, we conducted a listening test in a controlled environment involving a group of 12 human listeners[5]. Each participant was asked to label a set of 50 CAPTCHA signals. The participants were briefed that the signals consist of a varying number of digits that are separated in time by distinct speech pauses. Each CAPTCHA was labeled by four different participants to identify inconsistent labelings. We only utilize those transcriptions that exhibit an agreement between at least three participants. This procedure resulted in an overall number of 336 labeled CAPTCHAs — corresponding to 4859 labeled words — that we utilize for our evaluation.

An important difference between the two data sets arises from the speaking pauses that separate the individual digits from each other. While for the Aurora-5 observations there are speaking pauses between all digits, the reCAPTCHA observations consists of blocks of digits that are spoken in an unnaturally fast succession. We assume that this is done to make the separation problem harder.

B. Setup and Learning

To examine and compare the performance of our learning algorithms we conducted a series of experiments. Because we put the focus on comparison, we did not try to thoroughly optimize parameters shared by all learners to achieve the best possible error rate.

However, we were able to reduce the error rate for solving the current reCAPTCHA challenges exactly (i.e., with a Levenshtein distance of zero) by 44%—from 83%, published by Sano, Otsuka and Okuno [3], to 39%—with a similar amount of training data.[6]

In this paragraph we specify details of our ASR backend (these details are not required for understanding the rest of the paper; again, we refer interested readers unfamiliar with ASR to Rabiner and Juang [10]). We used 39-dimensional *Mel frequency cepstral coefficients* (MFCCs) as features including their first and second order derivatives, where each feature vector corresponds to a window length of 25 ms of the audio signal. Each digit is modeled by an HMM that has 18 states and exhibits a left-to-right topology without state skips. The silence/noise model had five states and allowed backward transitions and skips between the first and the last state. The state emission probabilities were represented by a Gaussian mixture model (GMM) having four mixture components.

In all experiments, the ratio between the number of randomly drawn observations (labeled and unlabeled) and the number of labels used was kept constant. On the Aurora-5 data

[5]*Ethical Considerations:* Before performing the listening test, all participants were informed that they were to take part in a scientific study and that their data was used to analyze the properties of a CAPTCHA. All data collected during our study was used in an anonymized way so that there is no link between collected data and individual participants. Our institute does not fall under the jurisdiction of an IRB or similar ethics committee.

[6]The error rates in [3] were found to be 83% and 48%, for a Levenshtein distance of zero and one, respectively, with 400 labeled observations available in total.

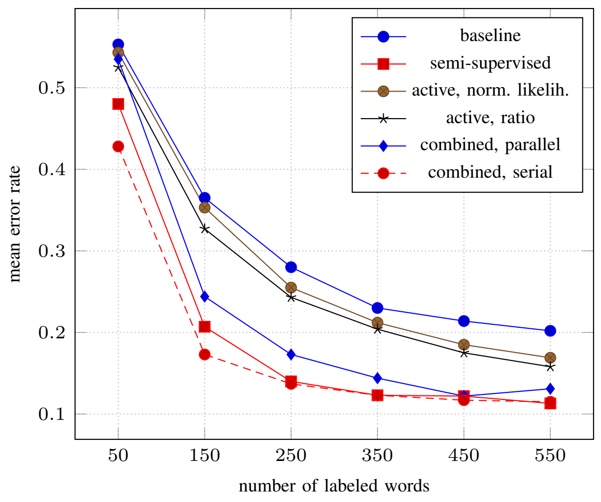

Fig. 1. Mean error rates for Aurora-5.

TABLE I
EXPERIMENTAL RESULTS FOR AURORA-5.

	baseline		**semi-supervised learner**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0– 99	0.553	0.103	0.480	0.187
100–199	0.365	0.071	0.207	0.094
200–299	0.280	0.051	0.140	0.036
300–399	0.230	0.041	0.123	0.032
400–499	0.214	0.036	0.122	0.026
500–599	0.202	0.035	0.113	0.020

	active, normalized likelihood		**active, likelihood ratio**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0– 99	0.543	0.109	0.525	0.119
100–199	0.353	0.065	0.327	0.067
200–299	0.255	0.039	0.243	0.035
300–399	0.212	0.030	0.204	0.036
400–499	0.185	0.025	0.175	0.025
500–599	0.169	0.026	0.158	0.019

	parallel combined learner		**serial combined learner**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0– 99	0.535	0.173	0.428	0.170
100–199	0.244	0.120	0.173	0.049
200–299	0.173	0.042	0.137	0.031
300–399	0.144	0.034	0.123	0.027
400–499	0.122	0.023	0.117	0.027
500–599	0.131	0.030	0.115	0.032

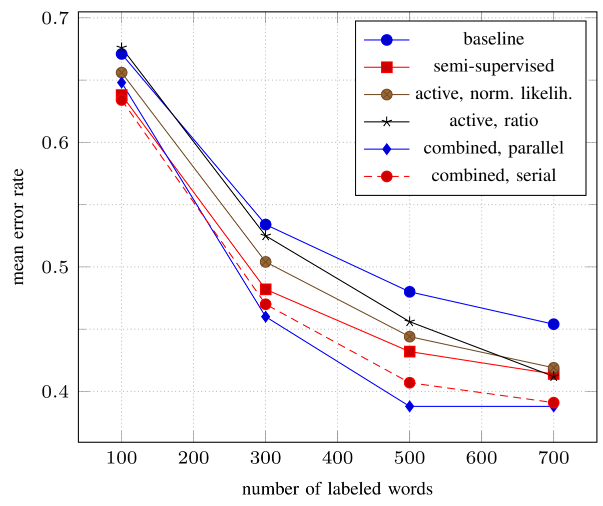

Fig. 2. Mean error rates for reCAPTCHA 2014.

TABLE II
EXPERIMENTAL RESULTS FOR RECAPTCHA.

	baseline		**semi-supervised learner**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0–199	0.671	0.104	0.638	0.144
200–399	0.534	0.098	0.482	0.136
400–599	0.480	0.100	0.432	0.136
600–799	0.454	0.103	0.414	0.135

	active, normalized likelihood		**active, likelihood ratio**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0–199	0.656	0.114	0.676	0.112
200–399	0.504	0.093	0.525	0.098
400–599	0.444	0.094	0.456	0.098
600–799	0.419	0.088	0.412	0.093

	parallel combined learner		**serial combined learner**	
# words	mean error rate	std. dev.	mean error rate	std. dev.
0–199	0.648	0.125	0.634	0.147
200–399	0.460	0.111	0.470	0.116
400–599	0.388	0.105	0.407	0.124
600–799	0.388	0.108	0.391	0.118

set this ratio is set to ten and on reCAPTCHA, due to the small amount of available labeled data, it is four. Intuitively, for optimal error rates this ratio should be as large as possible, but for comparisons any constant suffices. All iterations were done five times. Initial models were trained on fifty (Aurora-5) resp. twenty (reCAPTCHA) randomly chosen labeled observations.

1) Baseline: In addition to the learning algorithms given in Sec. II, we also run a purely supervised learner as a baseline. The baseline learner started—as all our learning algorithms—with an initial model and then added an increasing number of randomly drawn labeled observations using Baum-Welch training. We kept track of the number of additional labeled words and the empirical error rate (evaluated on an independent test set). After 10 (Aurora-5) resp. 20 (reCAPTCHA) runs, we calculated the mean error rates and standard deviations for several intervals of word counts. In the other experiments we proceeded in the same way and the results are shown in Tab. I (Aurora-5) resp. Tab. II (reCAPTCHA). A plot of the mean error rate against the word count is given in Fig. 1 resp. Fig. 2.

2) Semi-supervised learner: The semi-supervised learner operates in a similar fashion: it uses the model trained by the baseline as its initial model and then applies five rounds of semi-supervised training on randomly drawn unlabeled data (with a constant ratio between the number of unlabeled and labeled data as outlined above). As shown in Tab. I resp. Tab. II, the semi-supervised learner clearly outperforms the baseline, although the standard deviation is higher when the number of additional words is small.

A comparison with the supervised baseline allows us to estimate the value of unlabeled data: for example, on the Aurora-5 data set, we get a mean error rate of 21% by improving the initial model with 150 labeled and ca. 1350 unlabeled words[7], however we need over 500 labeled words

[7]Since on the Aurora-5 data set the ratio between all used observations and labeled observations is ten, and we designed the active learner in such a way that it has no bias in observation length.

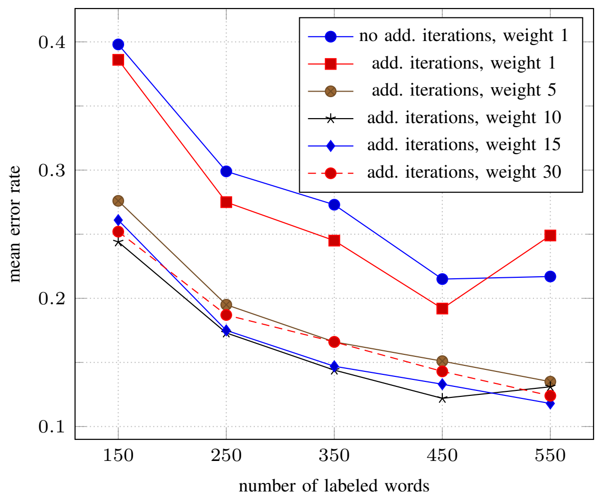

Fig. 3. Mean error rates for variants of the parallel combined learner using different weighting factors and options of additional semi-supervised learning iterations on Aurora-5.

to achieve the same performance when no unlabeled data are available.

3) Active learners: For active learning we examined both measures of confidence from Sec. II-E. As you can see in Fig. 1 and 2, both active learners are better than the baseline, which implies that both confidence measures can indeed identify observations whose label is more informative than the label of a randomly chosen one. Here we notice an important difference in the two data sets: on Aurora-5 the confidence measure based on the likelihood ratio outperforms the measure based on the normalized likelihood, while on reCAPTCHA we see the opposite behavior (except for a word count above 600, where the quotient is slightly better, which we interpret as an outlier.). The implications of this observation are discussed in Sec. III-C.

4) Combined learners: The combined learners are based on the superior confidence measure for each data set. We find that both parallel and serial learners are better than pure active learning, and on each data set one of the two combined learners is the best learner evaluated in this work. Again, the use of semi-supervised learning increases the standard deviation for small word counts.

5) Practicability and performance of the parallel combined learner: As seen in Fig. 3, we had to introduce weighting (with an empirically found optimal weight of ten for Aurora-5 and six for reCAPTCHA) and additional semi-supervised iterations to achieve a satisfactory performance for the parallel learner from Riccardi and Hakkani-Tür [16]. Please note that optimizing a parameter by experimentation—and by similar techniques like cross validation—is detrimental to active learning's goal of reducing the number of used labels, because an active learner will (typically) request the labels of a different fraction of the unlabeled data pool in each run. We are not aware of a method for optimizing the weight without using a large amount of labeled data, which is not readily available in situations where active learning is applied.

Furthermore, the optimized parallel learner is still worse than pure semi-supervised learning on Aurora-5, which implies that its active choice of labels is inferior to random sampling. While its performance on reCAPTCHA is better than the serial learner (except for small word counts), this gain seems too small to us to be worth the effort in real world attacks.

C. Implications for the Design of CAPTCHAs

The observation that active learning based on the likelihood ratio performs better than the one based on the normalized likelihood on the Aurora-5 data set, is best explained by the frequent confusion about the position of noise or silence (the example at the end of Sec. II-E was from Aurora-5). Further evidence is given by the fact that the normalized likelihood performs better on the reCAPTCHA data set, where we did not see these kind of confusions.

The computation of likelihood ratios is more expensive, since we have to determine the n best hypotheses instead of just one. To improve the resistance of audio CAPTCHAs against active learning, we would like to see this gap between the performances of the two methods on audio CAPTCHAs, and, at best, the gap should be much larger.

We suggest that both can be achieved by inserting superfluous words[8] from a preferably large vocabulary between the words that are relevant for solving the CAPTCHA (e.g., the digits in reCAPTCHA). These superfluous words play the role of the noise in Aurora-5.

Doing so should amplify the effect that occurred on Aurora-5: one simple silence/noise model does not suffice anymore; in the best case a different model must be trained for each superfluous word (the existence of a small number of models that recognize all superfluous words well must be prevented by the designer of the CAPTCHA scheme). Since the Viterbi algorithm must estimate which, where and how many superfluous words occurred, the confusion in the n best hypotheses will be much higher than it is now for Aurora-5. Therefore, even if the digit sequence is already very certain, the estimated labels will be assigned a low likelihood, and it becomes harder to distinguish already correctly recognized from uncertain digit sequences.

The introduction of superfluous words should not hamper human users too much, because they can focus on the necessary words (e.g., digits). On the other hand, the separation problem still has to be hard, so transitions between necessary and superfluous words have to be hard to detect.

IV. Conclusions and Open Questions

We gave several active and semi-supervised learning techniques and demonstrated that they are useful for reducing the demand of labeled data in breaking audio CAPTCHAs. We found that the results on our development set, Aurora-5, are generally transferable to reCAPTCHA. We argued that a serial combination of active and semi-supervised learning is preferable to a parallel one, especially in real world attacks.

[8]These do not necessarily have to be English words. Even noise signals could be used for the purpose.

We found evidence and theoretically motivated that the incorporation of superfluous words can hinder an active learning system and should therefore be included the design of audio CAPTCHA schemes.

We also raised open questions and gave directions for future work: What other measures of confidence are useful for active learning? How can we choose a good measure depending on the data set (without using much labeled data)? Is it possible to optimize parameters of active learning algorithms without using much labeled data? How well do the presented learning methods generalize to other CAPTCHA schemes?

Acknowledgments

The authors would like to thank Hans Ulrich Simon for helpful suggestions and all participants of the listening experiments at the Ruhr-Universität Bochum who provided the reCAPTCHA labels required for our work.

This research was supported in part by the DFG Research Training Group UbiCrypt (GRK 1817/1).

References

[1] J. Tam, J. Simsa, S. Hyde, and L. von Ahn, "Breaking Audio CAPTCHAs," in *NIPS*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008.

[2] E. Bursztein, R. Beauxis, H. S. Paskov, D. Perito, C. Fabry, and J. C. Mitchell, "The Failure of Noise-Based Non-continuous Audio Captchas," in *IEEE Symposium on Security and Privacy*, 2011.

[3] S. Sano, T. Otsuka, and H. G. Okuno, "Solving Google's Continuous Audio CAPTCHA with HMM-Based Automatic Speech Recognition," in *Proceedings of the International Workshop on Security*, 2013.

[4] D. Pearce and H.-G. Hirsch, "The Aurora Experimental Framework for the Performance Evaluation of Speech Recognition Systems under Noisy Conditions," in *Proc. ISCA ITRW ASR2000*, 2000.

[5] Google, "reCAPTCHA," http://www.recaptcha.net as of 03/2014.

[6] K. Chellapilla and P. Y. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)," in *NIPS*, 2004.

[7] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs)," in *Proceedings of the Second International Workshop on Human Interactive Proofs*, 2005.

[8] A. Hindle, M. W. Godfrey, and R. C. Holt, "Reverse Engineering CAPTCHAs," in *Proc. WCRE*, 2008.

[9] J. Yan and A. S. El Ahmad, "Breaking visual captchas with naive pattern recognition algorithms," in *Proc. ACSAC*, 2007.

[10] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, 1977.

[12] M. Kearns, Y. Mansour, and A. Y. Ng, "An Information-theoretic Analysis of Hard and Soft Assignment Methods for Clustering," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 1997.

[13] G. Zavaliagkos and T. Colthurst, "Utilizing Untranscribed Training Data To Improve Performance," in *DARPA Broadcast News Transcription and Understanding Workshop, Landsdowne*, 1998.

[14] S. Zhong, "Semi-supervised sequence classification with HMMs," *International Journal of Pattern Recognition & Artificial Intelligence*, 2005.

[15] V. I. Spitkovsky, H. Alshawi, D. Jurafsky, and C. D. Manning, "Viterbi Training Improves Unsupervised Dependency Parsing," in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, Stroudsburg, PA, USA, 2010.

[16] G. Riccardi and D. Z. Hakkani-Tür, "Active and unsupervised learning for automatic speech recognition," in *INTERSPEECH*, 2003.

[17] D. Yu, B. Varadarajan, L. Deng, and A. Acero, "Active learning and semi-supervised learning for speech recognition: A unified framework using the global entropy reduction maximization criterion," *Computer Speech & Language*, 2010.

[18] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. MIT Press, 2006.

[19] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning*, 1994.

[20] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[21] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, 1970.

[22] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, 1967.