RUHR-UNIVERSITÄT BOCHUM

RUB

Using Automatic Speech Recognition for Attacking Acoustic CAPTCHAs: The Trade-off between Usability and Security

Hendrik Meutzner*, Viet-Hung Nguyen, Thorsten Holz, Dorothea Kolossa

ACSAC’14, 11. December, 2014, New Orleans, Louisiana, USA

hgi
Horst Görtz Institute
for IT-Security

Outline

Outline

The Concept of CAPTCHAs

- *Completely Automated Public Turing Tests to Tell Computers and Humans Apart*

Example of Google’s image-based CAPTCHA scheme.

The Concept of CAPTCHAs

- *Completely Automated Public Turing Tests to Tell Computers and Humans Apart*

Example of Google's image-based CAPTCHA scheme.

- Distinguish humans from computers to limit or even prevent the abuse in Internet services, e.g.,
  - automated account creation for sending spam mail.

The Concept of CAPTCHAs

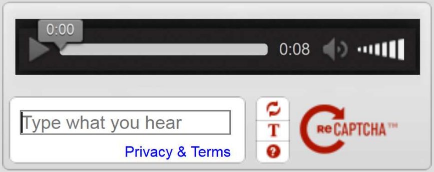
- *Completely Automated Public Turing Tests to Tell Computers and Humans Apart*

Example of Google's image-based CAPTCHA scheme.

- Distinguish humans from computers to limit or even prevent the abuse in Internet services, e.g.,
  - automated account creation for sending spam mail.
- CAPTCHAs should be easy to solve by humans but difficult to break by computers.

Acoustic CAPTCHAs

- Acoustic CAPTCHAs are beneficial for
  - visually impaired people,
  - hands-free operation,
  - non-graphical devices.



Example of Google's audio-based CAPTCHA scheme.

Usability and Security of CAPTCHAs

- Breaking CAPTCHAs represents a machine learning problem.

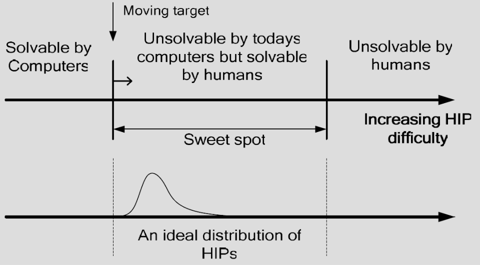
Usability and Security of CAPTCHAs

- Breaking CAPTCHAs represents a machine learning problem.
- A CAPTCHA is said to be broken if the success rate for automatic solving exceeds
  - 5 % [1], 1 % [2], 0.01 % [3],

[1] J. Tam et al., "Breaking Audio CAPTCHAs," NIPS 2008.
[2] E. Bursztein et al., "The Failure of Noise-Based Non-Continuous Audio Captchas," S&P 2011.
[3] K. Chellapilla et al., "Building segmentation based humanfriendly Human Interactive Proofs," HIP2005.

Usability and Security of CAPTCHAs

- Breaking CAPTCHAs represents a machine learning problem.
- A CAPTCHA is said to be broken if the success rate for automatic solving exceeds
  - 5 % [1], 1 % [2], 0.01 % [3],
- "For good usability the human success rate should approach 90 %." [3]

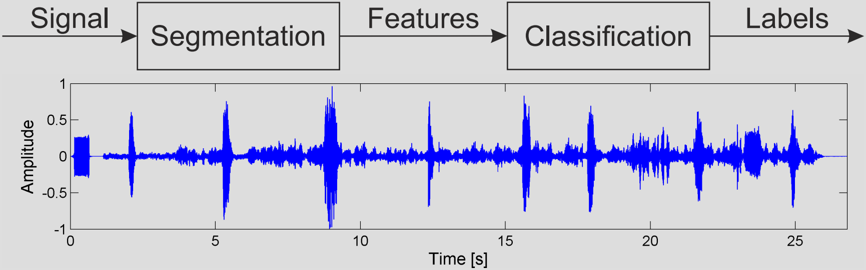


Regions of feasibility as a function of HIP difficulty for humans and computers algorithms. [3]

[1] J. Tam et al., "Breaking Audio CAPTCHAs," NIPS 2008.
[2] E. Bursztein et al., "The Failure of Noise-Based Non-Continuous Audio Captchas," S&P 2011.
[3] K. Chellapilla et al., "Building segmentation based humanfriendly Human Interactive Proofs," HIP2005.

Attacks on Acoustic CAPTCHAs

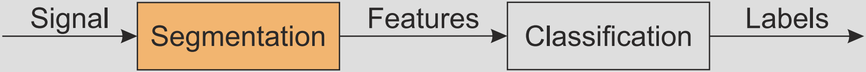
- Most previous attacks (e.g., [1,2]) are based on a two-stage approach:

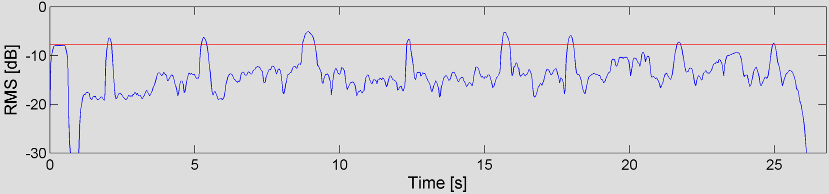


[1] E. Bursztein et al., "The Failure of Noise-Based Non-Continuous Audio Captchas," S&P 2011.
[2] J. Tam et al., "Breaking Audio CAPTCHAs," NIPS 2008.

Attacks on Acoustic CAPTCHAs

- Most previous attacks (e.g., [1,2]) are based on a two-stage approach:
  1. Computation of short-time signal energy and identification of peaks that exceed a specific energy threshold.

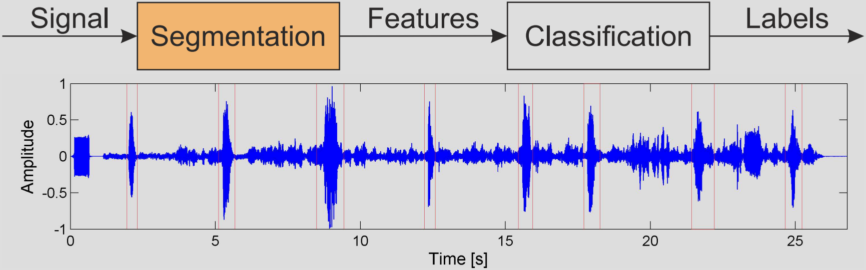




[1] E. Bursztein et al., “The Failure of Noise-Based Non-Continuous Audio Captchas,” S&P 2011.
[2] J. Tam et al., “Breaking Audio CAPTCHAs," NIPS 2008.

Attacks on Acoustic CAPTCHAs

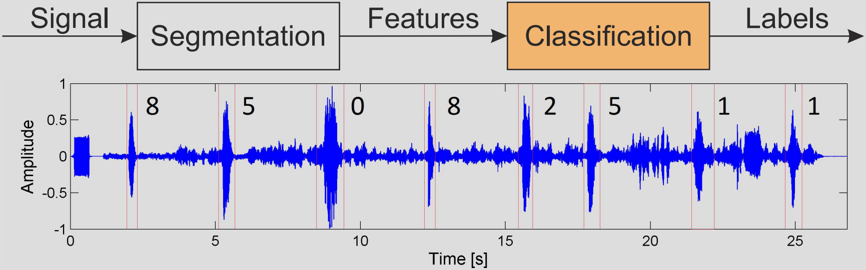
- Most previous attacks (e.g., [1,2]) are based on a two-stage approach:
  1. Computation of short-time signal energy and identification of peaks that exceed a specific energy threshold.
     ⇒ Energy peaks are used for signal segmentation.



[1] E. Bursztein et al., “The Failure of Noise-Based Non-Continuous Audio Captchas,” S&P 2011.
[2] J. Tam et al., “Breaking Audio CAPTCHAs," NIPS 2008.

Attacks on Acoustic CAPTCHAs

- Most previous attacks (e.g., [1,2]) are based on a two-stage approach:
  1. Computation of short-time signal energy and identification of peaks that exceed a specific energy threshold.
     $\Rightarrow$ Energy peaks are used for signal segmentation.
  2. Classification (Least Squares, SVMs) of isolated *word* segments.



[1] E. Bursztein et al., "The Failure of Noise-Based Non-Continuous Audio Captchas," S&P 2011.
[2] J. Tam et al., "Breaking Audio CAPTCHAs," NIPS 2008.

Outline

CAPTCHA Solver

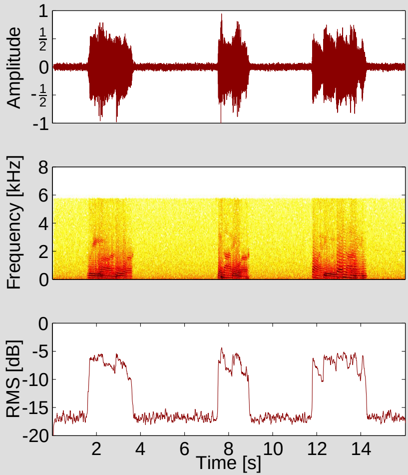
- We use a hidden Markov model (HMM) based recognizer[1].
- Each word is modeled by an HMM that has 3 emitting states per phoneme and exhibits a left-to-right topology without state skips.
- The speech pauses (i.e., silence/noise) are represented by an additional model that has 3 emitting states and allows backward transitions and skips between the first and the last state.
- The state emission probabilities are represented by a Gaussian mixture model (GMM) having 8 mixture components.
- The features are given by 39-dimensional perceptual linear prediction (PLP) coefficients including their first and second order derivatives.
- Each feature vector corresponds to a window length of 25 ms of the audio signal.

[1] S. Young, “The HTK Hidden Markov Model Toolkit: Design and Philosophy,” Entropic Cambridge Research Laboratory, Ltd, 1994.

Outline

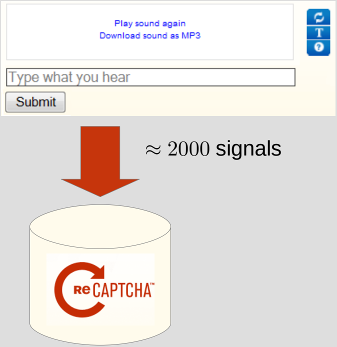
reCAPTCHA

Example: “314-694-5279”

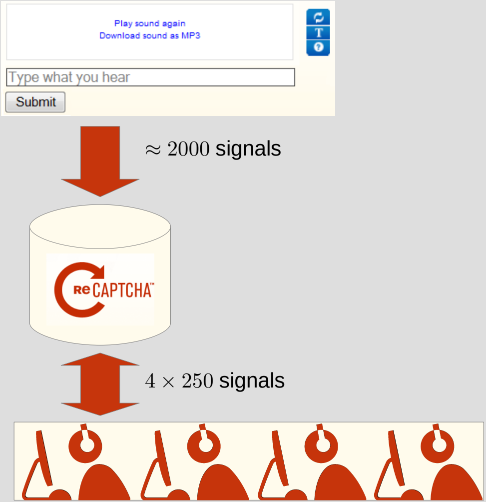


- The words are given by digits between “0” and “9”.
- The digits are spoken in a block-wise manner.
- The number of digits is varied between 6 and 12.
- Some of the digits are overlapping in time.
- The speech is synthetic and consists of a single female voice.
- The overall voice quality is comparatively low.
- All signals exhibit the same stationary background noise.

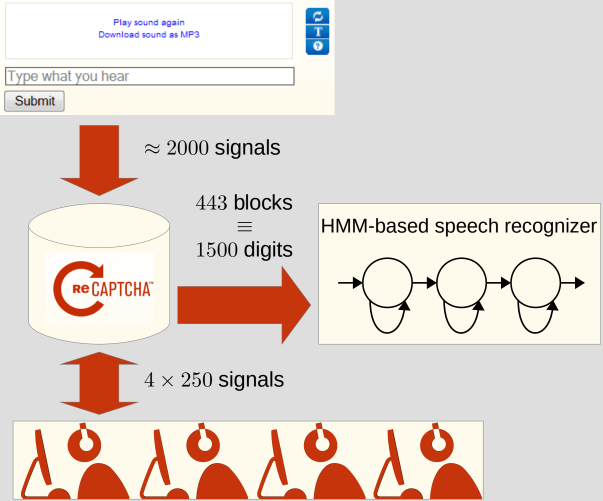
Downloading CAPTCHAs



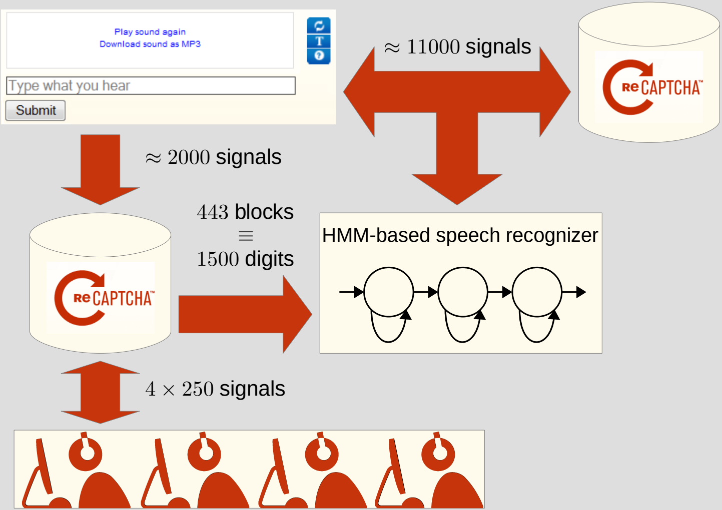
Obtaining Transcriptions



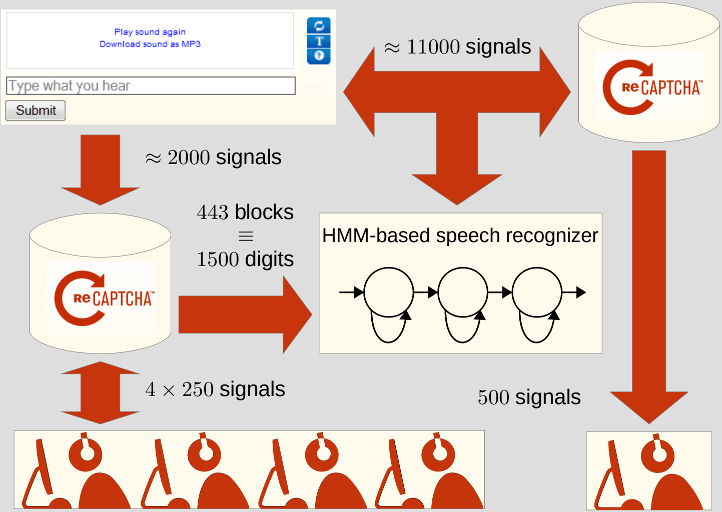
Training the Speech Recognizer



Security Analysis



Assessing Human Usability



Analysis Results

- Inter-labeler agreement (training corpus):

	# Agreements			
	1 (No)	2	3	4 (All)
Digit blocks	11.20 %	29.73 %	31.87 %	27.20 %
Full transcription	49.20 %	36.00 %	10.00 %	4.80 %

Analysis Results

- Inter-labeler agreement (training corpus):

	# Agreements			
	1 (No)	2	3	4 (All)
Digit blocks	11.20 %	29.73 %	31.87 %	27.20 %
Full transcription	49.20 %	36.00 %	10.00 %	4.80 %

- Success rate for automated CAPTCHA solving (attack): 63 %.

Analysis Results

- Inter-labeler agreement (training corpus):

	# Agreements			
	1 (No)	2	3	4 (All)
Digit blocks	11.20 %	29.73 %	31.87 %	27.20 %
Full transcription	49.20 %	36.00 %	10.00 %	4.80 %

- Success rate for automated CAPTCHA solving (attack): 63 %.
- Human success rate (listening test): 24 % (σ =17.35 %).

Analysis Results

- Inter-labeler agreement (training corpus):

	# Agreements			
	1 (No)	2	3	4 (All)
Digit blocks	11.20 %	29.73 %	31.87 %	27.20 %
Full transcription	49.20 %	36.00 %	10.00 %	4.80 %

- Success rate for automated CAPTCHA solving (attack): 63 %.
- Human success rate (listening test): 24 % ($\sigma = 17.35$ %).
- Previous attacks:

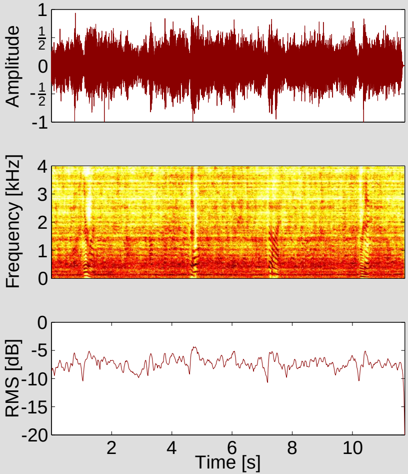
Authors	Method	Success rate
Bursztein et al. [1]	Classification	1.5 %
Sano et al. [2]	Speech recognition	52 %

[1] E. Bursztein et al., "The Failure of Noise-Based Non-Continuous Audio Captchas," S&P 2011.
[2] S. Sano, et al., "Solving Google's Continuous Audio CAPTCHA with HMM-Based Automatic Speech Recognition," Advances in Information and Computer Security, Springer, 2013.

Outline

Proposed CAPTCHA

Example: “01-64-75-36”



- The words are given by digits between “0” and “9”.
  $\Rightarrow$ Fair comparison, usability.
- **Real speech recordings** from **different speakers** (m/f).
- Two consecutive words are overlapping in time.
- **Non-stationary background noise**, scaled such that the signal energy is constant.
  $\Rightarrow$ Prevent isolation of words.
  $\Rightarrow$ Confuse speech recognizers.
- **Artificial reverberation**
  $\Rightarrow$ Automatic speech recognition is more challenging.
  $\Rightarrow$ Intelligibility remains good.

Creating CAPTCHAs 1/2

- Signal generation is based on a subset of the TIDIGITS speech corpus.
  - Single-digit recordings of 25 male and 25 female speakers, corresponding to 1000 individual digits.

Creating CAPTCHAs 1/2

- Signal generation is based on a subset of the TIDIGITS speech corpus.
  - Single-digit recordings of 25 male and 25 female speakers, corresponding to 1000 individual digits.
- Random digits are chosen from the database, alternating male and female speakers.

Creating CAPTCHAs 1/2

- Signal generation is based on a subset of the TIDIGITS speech corpus.
  - Single-digit recordings of 25 male and 25 female speakers, corresponding to 1000 individual digits.
- Random digits are chosen from the database, alternating male and female speakers.
- Two consecutive digits are superimposed in time.
  - The superposition of digits is based on their short-time power.

Creating CAPTCHAs 1/2

- Signal generation is based on a subset of the TIDIGITS speech corpus.
  - Single-digit recordings of 25 male and 25 female speakers, corresponding to 1000 individual digits.
- Random digits are chosen from the database, alternating male and female speakers.
- Two consecutive digits are superimposed in time.
  - The superposition of digits is based on their short-time power.
- The number of digit blocks per CAPTCHA is varied between 4–5 (8–10 digits per CAPTCHA).

Creating CAPTCHAs 1/2

- Signal generation is based on a subset of the TIDIGITS speech corpus.
  - Single-digit recordings of 25 male and 25 female speakers, corresponding to 1000 individual digits.

- Random digits are chosen from the database, alternating male and female speakers.

- Two consecutive digits are superimposed in time.
  - The superposition of digits is based on their short-time power.

- The number of digit blocks per CAPTCHA is varied between 4–5 (8–10 digits per CAPTCHA).

- All digit blocks are separated by speech pauses of random length.

Creating CAPTCHAs 2/2

- All speech pauses are superimposed by a multi-talker babble noise.
  - The noise signal is scaled such that the the short-time energy of the resulting signal is somewhat constant over time.

RUB

Creating CAPTCHAs 2/2

- All speech pauses are superimposed by a multi-talker babble noise.
  - The noise signal is scaled such that the the short-time energy of the resulting signal is somewhat constant over time.
- The mixture signal is reverberated by a randomly generated impulse response:

$$\begin{aligned} y(t) &= x(t) * h(t) \\ &= x(t) * \left(w(t) e^{-t/\tau} \right) \end{aligned}$$

$w(t)$: white Gaussian noise (random) τ: decay time (fixed)

Creating CAPTCHAs 2/2

- All speech pauses are superimposed by a multi-talker babble noise.
  - The noise signal is scaled such that the the short-time energy of the resulting signal is somewhat constant over time.

- The mixture signal is reverberated by a randomly generated impulse response:

$$\begin{aligned} y(t) &= x(t) * h(t) \\ &= x(t) * \left(w(t)e^{-t/\tau}\right) \end{aligned}$$

$w(t)$: white Gaussian noise (random) τ: decay time (fixed)

- We create and compare CAPTCHAs for two different decay times, i.e.,
  - $\tau \hat{=} T_{60} = 100$ ms,
  - $\tau \hat{=} T_{60} = 300$ ms.

Analysis Results

Speech recognition results (Attack):

# Train	T_{60} [ms]	Sent. [%]	Word [%]
200	0	15.86	77.03
200	100	5.33	64.49
200	300	1.25	56.11
400	0	17.42	78.38
400	100	5.06	65.32
400	300	2.34	60.20
800	0	20.38	79.71
800	100	6.87	67.21
800	300	3.14	62.88
1600	0	26.43	82.43
1600	100	6.26	67.37
1600	300	4.11	64.66

- All scores are based on 10,000 CAPTCHAs (sentences), corresponding to 90,140 words.

Analysis Results

Speech recognition results (Attack):

# Train	T_{60} [ms]	Sent. [%]	Word [%]
200	0	15.86	77.03
200	100	5.33	64.49
200	300	1.25	56.11
400	0	17.42	78.38
400	100	5.06	65.32
400	300	2.34	60.20
800	0	20.38	79.71
800	100	6.87	67.21
800	300	3.14	62.88
1600	0	26.43	82.43
1600	100	6.26	67.37
1600	300	4.11	64.66

- All scores are based on 10,000 CAPTCHAs (sentences), corresponding to 90,140 words.

Listening test results:

	Sent. [%]	Word [%]
μ	56.38	91.74
σ	21.47	7.18

$T_{60} = 100$ ms

	Sent. [%]	Word [%]
μ	37.81	86.88
σ	17.65	7.90

$T_{60} = 300$ ms

- The results were obtained from 16 individual participants for each reverberation time.
- The scores correspond to 800 CAPTCHAs (sentences) and 7,280 words.

Outline

Conclusions

reCAPTCHA vs. Proposed Scheme:

	ASR	Human
Proposed CAPTCHA ($T_{60} = 100$ ms)	5.33 %	56.38 %
reCAPTCHA (as of March 2014)	62.8 %	24.40 %

Conclusions

- Conservative CAPTCHAs can potentially be learned by machines at a relatively low cost.
- Increased CAPTCHA security (using signal distortions) comes at the cost of lower human pass rates.

Conclusions

- Conservative CAPTCHAs can potentially be learned by machines at a relatively low cost.
- Increased CAPTCHA security (using signal distortions) comes at the cost of lower human pass rates.
- We assume that the theoretical sweet-spot, i.e.,
  - high success rates for humans ($\geq 90\,\%$),
  - low success rates for machines ($\leq 1\,\%$ or even $\leq 0.01\,\%$).

  can not be achieved by using conventional methods.

Conclusions

- Conservative CAPTCHAs can potentially be learned by machines at a relatively low cost.

- Increased CAPTCHA security (using signal distortions) comes at the cost of lower human pass rates.

- We assume that the theoretical sweet-spot, i.e.,
  - high success rates for humans ($\geq 90\,\%$),
  - low success rates for machines ($\leq 1\,\%$ or even $\leq 0.01\,\%$).

  can not be achieved by using conventional methods.

- It is necessary to investigate into more sophisticated CAPTCHAs, e.g.,
  - CAPTCHAs that are based on context-dependent questions, requiring **intelligence** and/or **previous knowledge**.

Thank you!

Thank you!

Questions?

Winograd Schemas

Example 1:

Question: The trophy doesn’t fit into the brown suitcase because it’s too [small/large]. What is too [small/large]?

Answer: The [suitcase/the trophy].

Winograd Schemas

Example 1:

Question: The trophy doesn't fit into the brown suitcase because it's too [small/large]. What is too [small/large]?

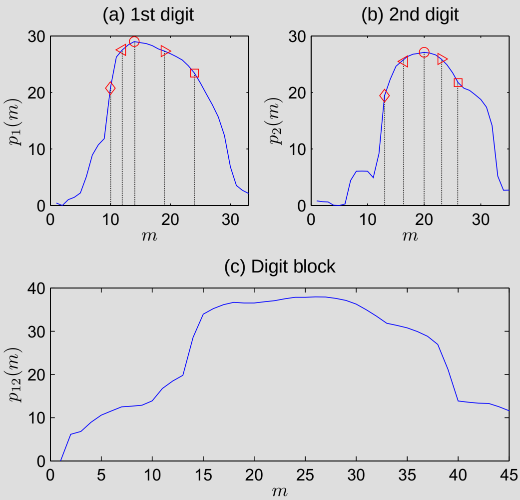
Answer: The [suitcase/the trophy].

Example 2:

Questions: The man couldn't lift his son because he was so [weak/heavy]. Who was [weak/heavy]?

Answer: The [man/the son].

Creating Digit Blocks

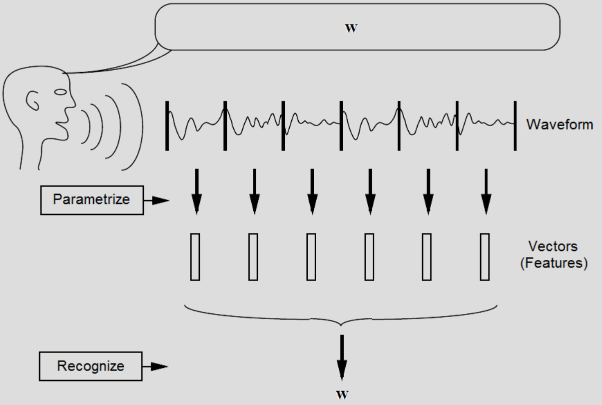


Metrics

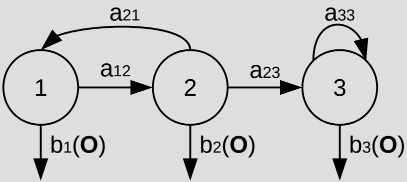
$$\text{Word Acc.} = 100 \cdot \frac{W - W_D - W_I - W_S}{W},$$
$$\text{Sent. Acc.} = 100 \cdot \frac{S_C}{S},$$

W:	Number of words
W_D:	Word deletions
W_I:	Word insertions
W_S:	Word substitutions
S:	Number of sentences
S_C:	Number of correctly transcribed sentences

General Task



Introduction of Hidden Markov Models



Example of an HMM

a_{ij}:	state transition probabilities
b_j:	output probabilities
$\mathbf{o}$:	observations (features)
π_j:	initial state probabilities

- The HMM consists of states and links.
  - Each link allows a transition between two states.
  - An observation is generated with a stochastic transition from one state to another.
  - Each observation $\mathbf{o}$ is one of the symbols in $V = \{v_1 \ldots v_K\}$
- The complete HMM is defined by the parameter set $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\Pi})$.

The Three Basic Problems for HMMs

1. Given the observation sequence $\mathbf{O} = \mathbf{o}_1\, \mathbf{o}_2 \ldots \mathbf{o}_T$ and the model λ, how to compute

$$P(\mathbf{O}|\lambda) \text{ ?}$$

2. Given the observation sequence $\mathbf{O} = \mathbf{o}_1\, \mathbf{o}_2 \ldots \mathbf{o}_T$ and the model λ, how to choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$, i.e.,

$$Q^* = \arg\max_{Q} P(Q|\mathbf{O}, \lambda)$$

that best explains the observations? $\rightarrow$ “Recognition”

3. How to adjust $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\Pi})$ to maximize $P(\mathbf{O}|\lambda)$ (maximum likelihood estimation)? $\rightarrow$ “Training”

$$\lambda_{\text{ML}} = \arg\max_{\lambda} \mathrm{P}(\mathbf{O}|\lambda)$$

Problem 1 - Computation of $P(\mathrm{O}|\lambda)$

- Naive approach: enumerate every possible state sequence

$$
\begin{aligned}
P(\mathbf{O}|Q,\lambda) &= \prod_{t=1}^{T} P(\mathbf{o}_t|q_t,\lambda) \quad \text{(statistical independent observations)} \\
&= b_{q_1}(\mathbf{o}_1) b_{q_2}(\mathbf{o}_2) \cdots b_{q_T}(\mathbf{o}_T) \\
P(Q|\lambda) &= \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \\
P(\mathbf{O}|\lambda) &= \sum_{\forall Q} P(\mathbf{O}|Q,\lambda) P(Q|\lambda) \\
&= \sum_{q_1, q_2, \ldots, q_T} \pi_{q_1} b_{q_1}(\mathbf{o}_1) a_{q_1 q_2} b_{q_2}(\mathbf{o}_2) \cdots a_{q_{T-1} q_T} b_{q_T}(\mathbf{o}_T)
\end{aligned}
$$

- Computational infeasible due to $2TN^T$ calculations.

Problem 1 - Computation of $P(\mathrm{O}|\lambda)$

- More efficient approach: forward-backward procedure

$$\alpha_t(i) = P(\mathbf{o}_1\mathbf{o}_2\cdots\mathbf{o}_t, q_t = S_i|\lambda) \quad \text{(forward variables)}$$

1. Initialization

$$\alpha_1(i) = \pi_i b_i(\mathbf{o}_1), \quad 1 \leq i \leq N$$

2. Recursion

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(\mathbf{o}_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

3. Termination

$$P(\mathbf{O}|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad \text{(terminal forward variables)}$$

- N^2T calculations, rather than $2TN^T$ as for the naive approach.

RUB

Problem 1 - Computation of $P(\mathrm{O}|\lambda)$

- Backward procedure:

$$\beta_t(i) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}\cdots\mathbf{o}_T, q_t = S_i|\lambda) \quad \text{(backward variables)}$$

1. Initialization

$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. Recursion

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{o}_{t+1}) \beta_{t+1}(j), \quad t = T-1, \ldots, 1, \quad 1 \leq i \leq N$$

- N^2T calculations

RUB

Problem 2 - Optimal State Sequence

- Find $Q^* = \arg\max_{Q} P(Q|\mathbf{O}, \lambda)$.
- Maximize the expected number of correct individual states.
- Probability of being in state S_i at time t, given $\mathbf{O}$ and λ:

$$\begin{aligned}\gamma_t(i) &= P(q_t = S_i|\mathbf{O}, \lambda) \\ &= \frac{\alpha_t(i)\beta_t(i)}{P(\mathbf{O}|\lambda)} \\ &= \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}, \quad \left(\sum_{i=1}^{N} \gamma_t(i) = 1\right)\end{aligned}$$

- The most likely state q_t^* at time t is then given by

$$q_t^* = \arg\max_{1\leq i\leq N} \{\gamma_t(i)\}, \quad 1 \leq t \leq T.$$

- How to find the single best state sequence?

Viterbi Algorithm

- Note that $Q^* = \arg\max_Q P(Q|\mathbf{O},\lambda) = \arg\max_Q P(Q,\mathbf{O}|\lambda)$
- Define a score along a single path at time t that ends in state S_i:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1 q_2 \ldots q_t = i, \mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_t|\lambda)$$

1. Initialization ($1 \leq i \leq N$):

$$\delta_1(i) = \pi_i b_i(\mathbf{o}_1) \qquad \Psi_1(i) = 0$$

2. Recursion ($2 \leq t \leq T, 1 \leq j \leq N$):

$$\delta_t(j) = \left(\max_{i=1\ldots N} \delta_{t-1}(i)a_{ij}\right) b_j(\mathbf{o}_t) \qquad \Psi_t(j) = \arg\max_{i=1\ldots N} \delta_{t-1}a_{ij}$$

3. Termination:

$$P^*(\mathbf{o}_1\ldots\mathbf{o}_T|\lambda) = \max_{i=1\ldots N} \delta_T(i) \qquad q_T^* = \arg\max_{i=1\ldots N} \delta_T(i)$$

4. Path backtracking: $q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, \cdots, 1.$

Problem 3 - Adjust the Model Parameters λ

- There is no known way to analytically solve for the model, which maximizes the probability of the observation sequence.
- Choose $\lambda = (\mathbf{A}, \mathbf{B}, \boldsymbol{\Pi})$ such that $P(\mathbf{O}|\lambda)$ is (locally) maximized.

Expectation maximization (EM) algorithm

- General method (not only for HMMs).
- Start with some λ^0.
- Iteratively compute:
  1. $F(\lambda, \lambda^{t-1}) := E_Q \left[\log P(\mathbf{O}, Q|\lambda) \,|\, \mathbf{O}, \lambda^{t-1}\right]$ (E-step)
  2. $\lambda^t = \mathrm{argmax}_\lambda F(\lambda, \lambda^{t-1})$ (M-step)
- An increase of F provably increases the likelihood $P(\mathbf{O}|\lambda)$.
- Provably converges to a local maximum of $P(\mathbf{O}|\lambda)$.
- For estimating HMM parameters, an instance of the EM algorithm is used, namely the Baum-Welch algorithm.

RUB

Baum-Welch Algorithm

- Probability of being in state S_i at time t, state S_j at time $t+1$, given $\mathbf{O}$ and λ:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j | \mathbf{O}, \lambda)$$
$$= \frac{\alpha_t(i) a_{ij} b_j(\mathbf{O}_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O}|\lambda)} \qquad = \frac{\alpha_t(i) a_{ij} b_j(\mathbf{O}_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(\mathbf{O}_{t+1}) \beta_{t+1}(j)}$$

- Express $\gamma_t(i)$ in terms of $\xi_t(i,j)$:

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i,j)$$

Reestimation formulae:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \qquad \hat{b}_j(k) = \frac{\sum_{\substack{t=1 \\ \text{s.t.} \mathbf{o}_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \qquad \hat{\pi}_i = \gamma_1(i)$$

Baum-Welch Algorithm

$$\hat{a}_{ij} = \frac{\text{expected \# of transitions from state } S_i \text{ to } S_j}{\text{expected \# of transitions from state } S_i}$$

$$\hat{b}_j(k) = \frac{\text{expected \# of times being in state } S_j \text{ and observing symbol } v_k}{\text{expected \# of times being in state } S_j}$$

$$\hat{\pi}_i = \text{expected \# of times being in state } S_i \text{ at time } t = 1$$

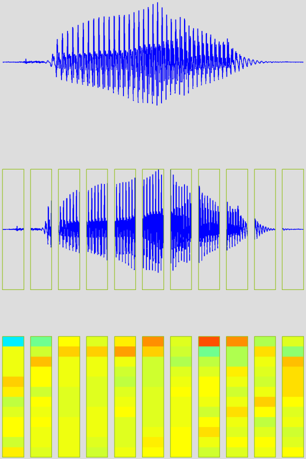
Using HMMs for Speech Recognition

- Represent each word or phoneme by an individual HMM.
- A common model topology is a left-to-right model (possibly with skips).
- The output probabilities $b_q(\mathbf{o})$ are modeled by using continuous density multivariate distributions, e.g., Gaussian mixture models (GMMs):

$$b_q(\mathbf{o}) = \sum_{\kappa=1}^{K} c_{\kappa,q} \mathcal{N}(\mathbf{o}|\boldsymbol{\mu}_{\kappa,q}, \underline{\boldsymbol{\Sigma}}_{\kappa,q}), \tag{1}$$

$$\sum_{\kappa=1}^{K} c_{\kappa,q} = 1 \quad \forall q. \tag{2}$$

Extracting Feature Vectors



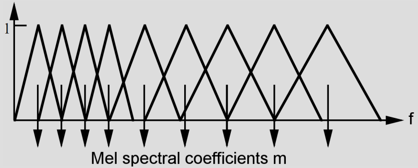
1. Segmentation of the input signal into (overlapping) frames
   - Typical frame lengths ≈ 25 ms
   - Overlap between frames ≈ 50–75 %
2. Extraction of features for each frame, e.g.,
   - Mel Frequency Cepstral Coefficients (MFCC),
   - Perceptual Linear Prediction (PLP),
   - Considering dynamics by incorporating 1st and 2nd order derivatives (Δ, $\Delta\Delta$).

Feature Extraction (MFCCs)

1. Compute the short-time Fourier transform (STFT) for each frame:

$$X(m,n) = \sum_{i=0}^{L-1} x(mR+i)h(i)e^{-j\frac{2\pi i}{L}n}$$

2. Warp spectral components onto the Mel scale:



3. Apply the discrete cosine transform to the log-Mel spectrum:

$$\tilde{X}(m,c) = \sum_{m=0}^{L'-1} \ln\left(\hat{X}(m,m)\right) \cos\left[\frac{\pi}{L'}\left(m+\frac{1}{2}\right)c\right].$$

4. Observation vector: $\mathbf{o}_t = \left[\tilde{X}(m,0) \quad \tilde{X}(m,1) \quad \cdots \quad \tilde{X}(m,L'-1)\right]^{\mathrm{T}}.$

Training using Sentences

- Reestimation algorithm, e.g., Baum-Welch, remains unchanged by using sentences, i.e., sequences of words, for training.

  1. For training, the corresponding transcriptions for each sentence (“labels”) have to be known.

  2. The respective models for each sentence are concatenated, which results in a larger HMM.

  3. The resulting larger HMM is trained by using the Baum-Welch reestimation procedure.

Recognition of Continuous Speech

- Combine individual word models into compound HMM.
  - Adjust the compound HMM to the underlying grammar, e.g.: <“one” or “two” or “three” or ... or “silence”>