Confidence Estimation for Trustworthy and Efficient Speech Systems - Part II

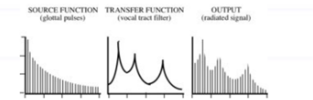
Interspeech 2025

Nagarathna Ravi, Thishyan Raj T, Vipul Arora

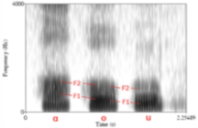
Indian Institute of Technology - Kanpur

August 17, 2025

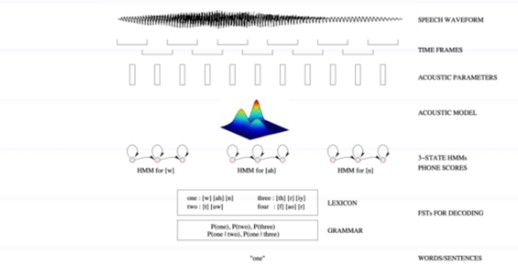
Evolution of Speech Technology



[Image from Jacques Koreman, 1995]



[Image from Ken Stevens, 2000]



- Linguistic Rules and Signal Processing
- GMM-HMM
- DNN-HMM

Machine Learning: End-to-End

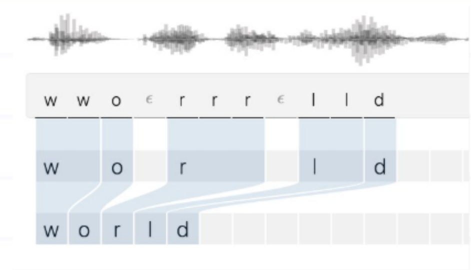


Image from https://distill.pub/2017/ctc/

Hello, I am giving this talk

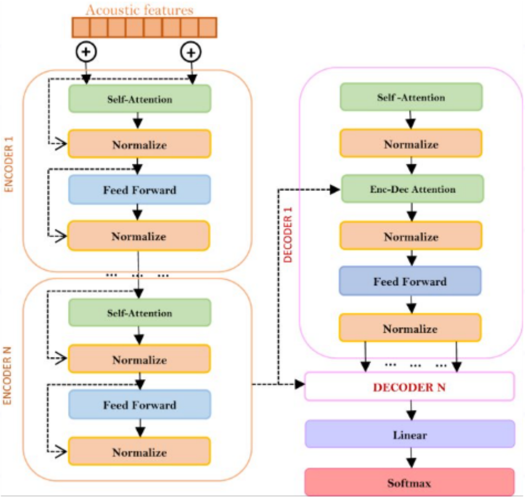


Image from https://doi.org/10.1038/s41598-022-12260-y

Automatic Speech Recognition - ASR

Example of ASR with Confidence Estimation

Ground Truth: Cats are cute

ASR Output: Cats are cube

Token-level confidence:
_cats (0.96), _are (0.94), _cu (0.65), be (0.42)

Word-level confidence:
cats (0.96), are (0.94), cube (0.42)

ASR Model — Dataset and Notations

- The speech dataset $\mathcal{D}$ contains speech–transcript pairs:

$$\{(X_i, z_i)\}$$

  where:
  - X_i — representation of a speech audio file
  - z_i — corresponding ground truth transcription
- Each audio representation:

$$X = [x_1, x_2, \ldots, x_T]$$

  where:
  - $x_t \in \mathbb{R}^D$ — D-dimensional frame at time $t \in \{1, \ldots, T\}$
- Each transcription:

$$z = [z_1, z_2, \ldots, z_U], \quad z_u \in \mathcal{V}$$

  where:
  - $\mathcal{V}$ — set of tokens (phonemes, characters, word-pieces, etc.)

ASR Model — Words and Prediction

- Tokens can be grouped into words:

$$w = [w_1, w_2, \dots, w_N]$$

Example:

$$[w_1, w_2, \dots, w_N] = [[z_1, z_2], [z_3, z_4, z_5], \dots, [z_U]]$$

- ASR model F_Θ is trained on $\mathcal{D}' \subset \mathcal{D}$ to predict:

$$\hat{z} = F_\Theta(X)$$

where:

$$\hat{z} = [\hat{z}_1, \hat{z}_2, \dots, \hat{z}_{U'}]$$

- Predicted tokens can also be grouped into words:

$$\hat{w} = [\hat{w}_1, \hat{w}_2, \dots, \hat{w}_{N'}]$$

ASR Model — CTC

- The ASR model F_Θ is trained using the **Connectionist Temporal Classification (CTC)** loss.
- Let $\mathcal{Z}^{\text{CTC}}_{\text{align}}(z)$ be the set of all valid alignments of z (with blanks $\varnothing$) to a sequence of length T.
- For an alignment $a = [a_1, a_2, \ldots, a_T]$ where $a_t \in \mathcal{V} \cup \{\varnothing\}$:

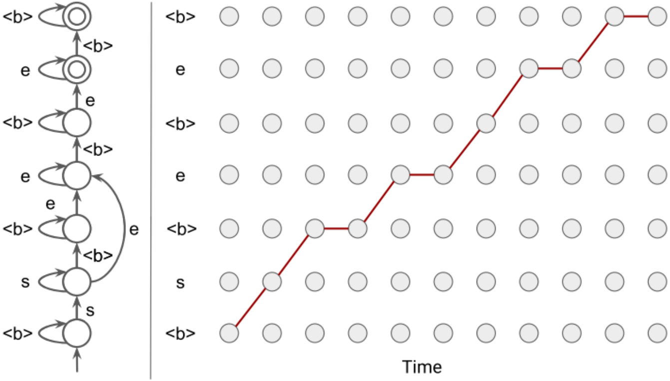
$$P(a \mid X) = \prod_{t=1}^{T} P(a_t \mid x_t; \Theta)$$

- The CTC loss for one (X, z) pair is:

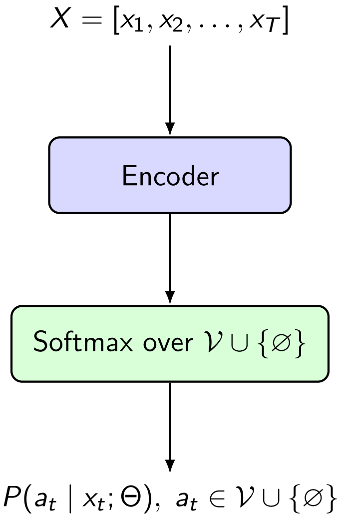
$$\mathcal{L}_{\text{CTC}}(X, z) = -\log \sum_{a \in \mathcal{Z}^{\text{CTC}}_{\text{align}}(z)} P(a \mid X)$$

ASR Model — CTC Decoding

- **Greedy decoding:** Select $\hat{a}_t = \arg\max_{v \in \mathcal{V} \cup \{\varnothing\}} P(v \mid x_t)$, then remove repeats and blanks.
- **Beam search decoding:** Keep top-K candidate sequences.
- Predicted tokens $\hat{z}$ can be grouped into words $\hat{w}$ as described earlier.



ASR Model — CTC Block Diagram



ASR Model — RNN-T

- The ASR model F_Θ is trained using the **Recurrent Neural Network Transducer (RNN-T)** loss.
- RNN-T jointly models acoustic and linguistic context via:

$$P(\hat{z} \mid X) = \sum_{\pi \in \mathcal{Z}_{\text{align}}^{\text{RNN-T}}(\hat{z})} P(\pi \mid X; \Theta)$$

  where $\pi = [\pi_{1,1}, \pi_{1,2}, \ldots, \pi_{T,U}]$ is a valid alignment path including blanks $\varnothing$.
- Each probability is factorized using:

$$P(k \mid t, u) = \mathrm{Softmax}(\mathbf{W}\, h_{t,u} + \mathbf{b})$$

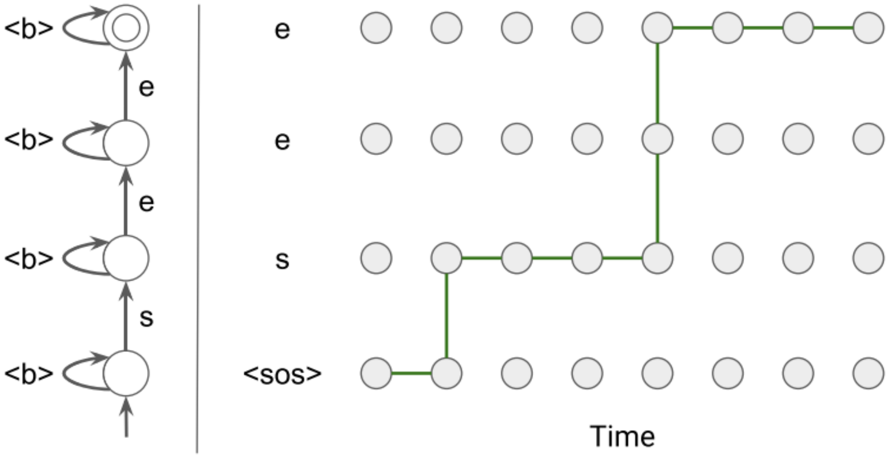
  where:
  - t — acoustic time step, u — output token index
  - $h_{t,u}$ — joint network output from encoder and prediction network
- The RNN-T loss is:

$$\mathcal{L}_{\text{RNNT}}(X, z) = -\log P(z \mid X)$$

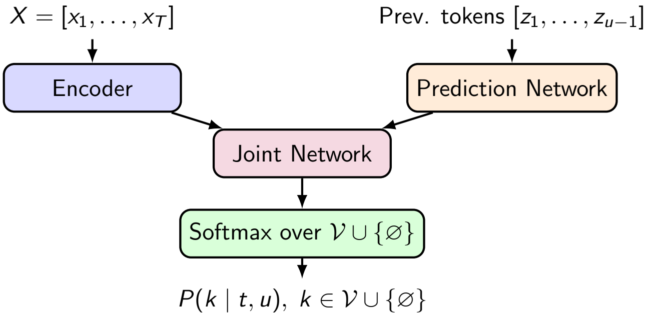
ASR Model — RNN-T Decoding

- At inference, decoding proceeds step-by-step:
  1. Encoder processes $X = [x_1, \ldots, x_T]$ into hidden states.
  2. Prediction network takes previous output token to produce linguistic state.
  3. Joint network combines both to output a distribution over $\mathcal{V} \cup \{\varnothing\}$.
- **Greedy decoding:** At each (t, u), pick the highest-probability symbol; if blank, advance t; otherwise, emit token and advance u.
- **Beam search decoding:** Maintain top-K partial hypotheses for better accuracy.
- Final output sequence $\hat{z}$ is obtained, and can be grouped into predicted words $\hat{w}$.

RNN-T



ASR Model — RNN-T Block Diagram



ASR Model — Encoder–Decoder Formulation

- The encoder–decoder model directly estimates:

$$P(z \mid X) = \prod_{u=1}^{U} P(z_u \mid z_{1:u-1}, X)$$

where:

  - $X = [x_1, x_2, \ldots, x_T]$ — acoustic feature sequence
  - $z = [z_1, z_2, \ldots, z_U]$ — token sequence, $z_u \in \mathcal{V}$

- **Encoder:**

$$h_t^{\mathrm{enc}} = f_{\mathrm{enc}}(x_t; \Theta_{\mathrm{enc}})$$

produces high-level acoustic representations.

- **Decoder:**

$$h_u^{\mathrm{dec}} = f_{\mathrm{dec}}(z_{1:u-1}, c_u; \Theta_{\mathrm{dec}})$$

where c_u is the attention context vector.

ASR Model — Encoder–Decoder Formulation

- Final token distribution:

$$P(z_u \mid z_{1:u-1}, X) = \mathrm{Softmax}(W_o\, g(h_u^{\mathrm{dec}}, c_u) + b_o)$$

Levenshtein Alignment — Example

- **Reference:** `the cat jumped here and there`
- **Hypothesis:** `the kat jumbled hair the ear`

Ref	Hyp	Op
the	the	correct
cat	kat	substitution
jumped	jumbled	substitution
here	hair	substitution
and	and	correct
there	the	substitution
	ear	insertion

Uncertainty Estimation Metrics (1/2)

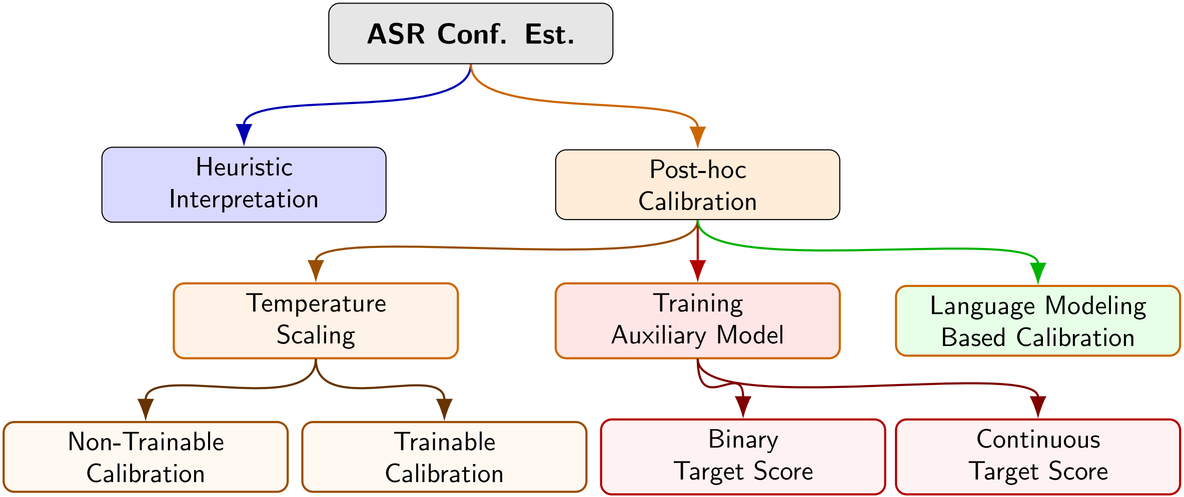
- **Mean Absolute Error (MAE):** Average absolute difference between c and $\hat{c}$.
- **Kullback–Leibler Divergence (KLD):** Measures how much the estimated confidence distribution differs from the true one.
- **Jensen–Shannon Divergence (JSD):** Symmetric, smoothed version of KLD; bounded between 0 and 1.
- **Normalized Cross Entropy (NCE):** Cross entropy between c and $\hat{c}$, normalized by the entropy of c.

Uncertainty Estimation Metrics (2/2)

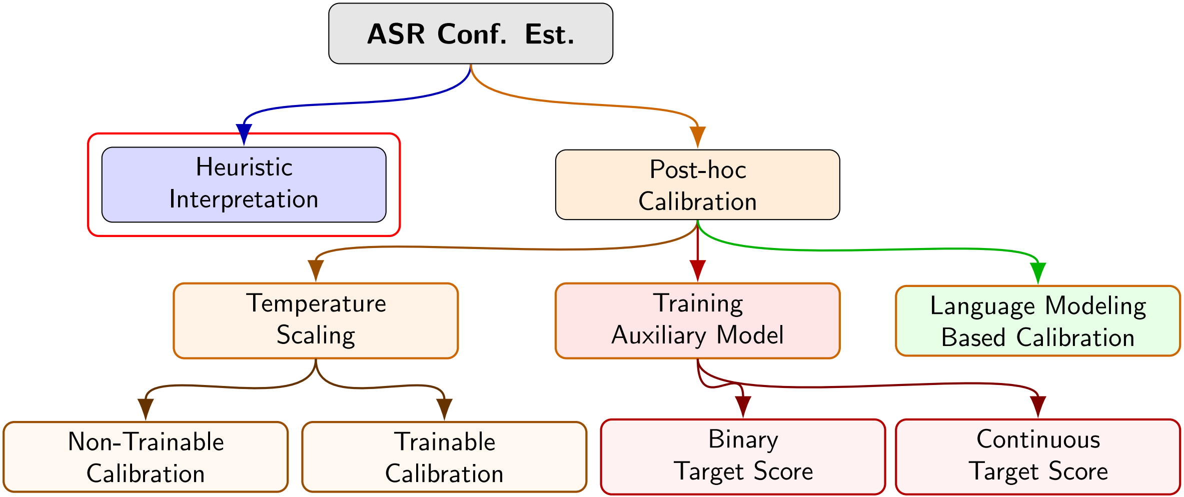
- **Expected Calibration Error (ECE):** Weighted average of $|\text{accuracy} - \text{confidence}|$ over bins of $\hat{c}$.
- **Maximum Calibration Error (MCE):** Maximum calibration gap across all bins.
- **RMSE – Word Correctness Ratio:** Root mean square error between estimated confidence and actual word correctness ratio.

ASR Uncertainty Estimation

ASR Uncertainty Estimation



ASR Uncertainty Estimation



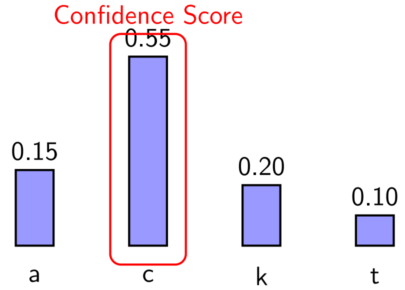
Heuristic Interpretation — Token-level Confidence

- **Idea:** Use the model's own output probabilities as a proxy for confidence.
- For each predicted token $\hat{z}_u \in \mathcal{V}$, the ASR model produces a probability distribution:

$$p(\hat{z}_u = v \mid X), \quad v \in \mathcal{V}$$

- The **confidence score** for the predicted token is taken as:

$$c_u = \max_{v \in \mathcal{V}} \; p(\hat{z}_u = v \mid X)$$

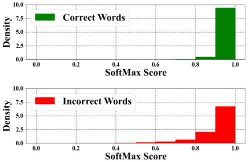


Heuristic Interpretation — Word-level Confidence

- For a predicted word $\hat{w}_n = [\hat{z}_{u_1}, \ldots, \hat{z}_{u_k}]$, aggregate token confidences, e.g.,

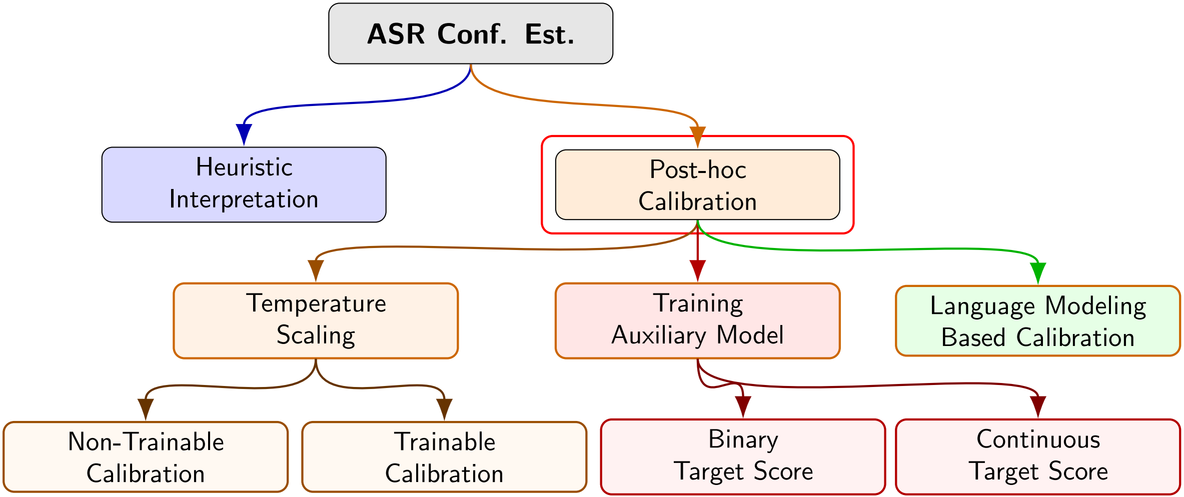
$$c_{\hat{w}_n} = \frac{1}{k} \sum_{j=1}^{k} c_{u_j}$$

- Softmax probabilities can be over-confident and poorly calibrated.

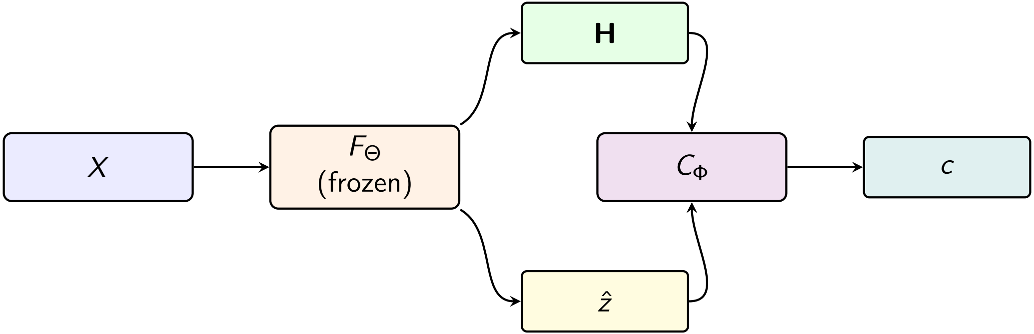


D. S. Park *et al.*, "Improved noisy student training for automatic speech recognition," *Interspeech*, 2020.
Y. Chen *et al.*, "Semi-supervised ASR by end-to-end self-training," *Interspeech*, 2020.

ASR Uncertainty Estimation

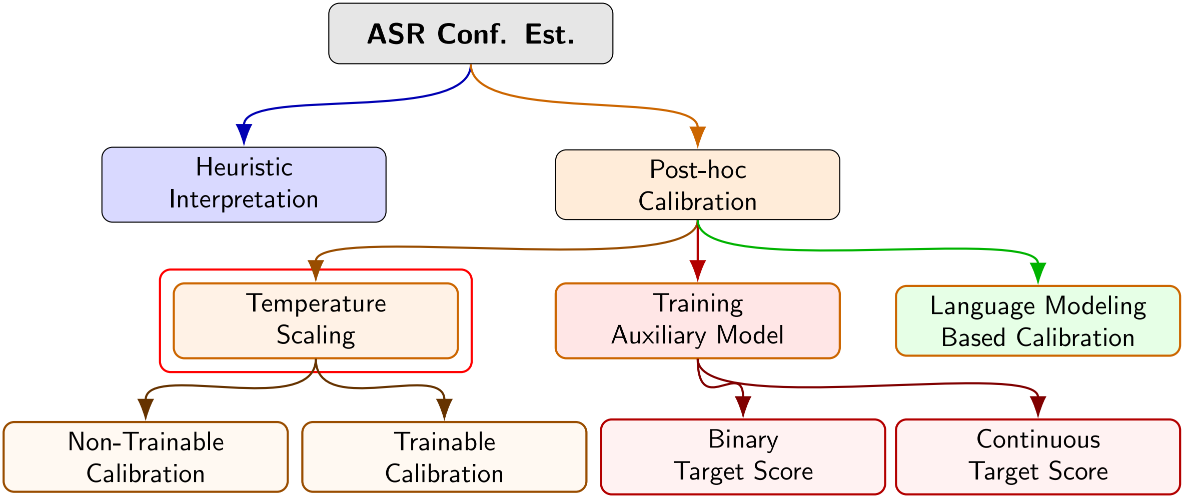


Post-hoc Calibration

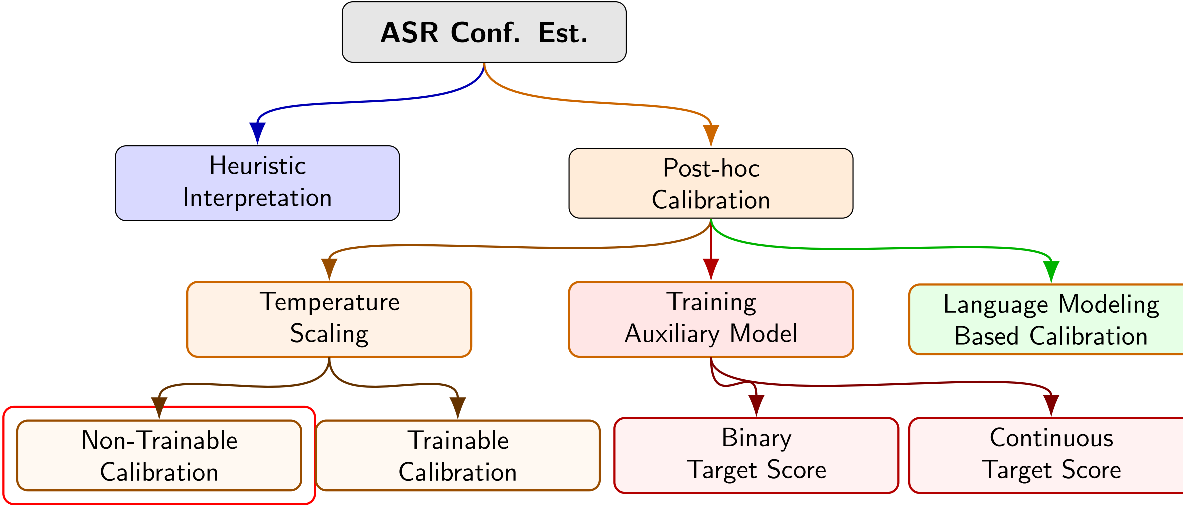


- C_Φ — post-hoc calibration function, can be **trainable** or **non-trainable**.
- $\mathbf{H}$ — combination of one or more intermediate representations from F_Θ.
- c — token-level or word-level confidence scores.

ASR Uncertainty Estimation



ASR Uncertainty Estimation



Temperature Scaling — Non-trainable Calibration (1/2)

Class probabilities from F_Θ :

$$\mathbf{H} = [p_v]_{v \in \mathcal{V}}, \quad p_v = P(\hat{z}_u = v \mid X)\,.$$

Tsallis entropy (parameter $\alpha \in (0,1)$):

$$H_{\mathrm{ts}}(\mathbf{H};\alpha) = \frac{1}{\alpha - 1}\left(1 - \sum_{v \in \mathcal{V}} p_v^\alpha\right).$$

Maximum Tsallis entropy (uniform distribution):

$$H_{\mathrm{ts}}^{\mathsf{max}}(\alpha, |\mathcal{V}|) = \frac{1}{\alpha - 1}\left(1 - |\mathcal{V}|^{\,1-\alpha}\right) = \frac{|\mathcal{V}|^{\,1-\alpha} - 1}{1 - \alpha}.$$

Example: $\mathcal{V} = \{a, c, k, t\}$

$$\mathbf{H} = [0.1,\ 0.8,\ 0.05,\ 0.05], \quad \alpha = 0.5$$

Step 1: Compute p_v^α

$$[0.3162,\ 0.8944,\ 0.2236,\ 0.2236]$$

Step 2: Sum:

$$\Sigma = 1.6578$$

Step 3: Tsallis entropy:

$$H_{\mathrm{ts}} = \frac{1 - 1.6578}{-0.5} \approx 1.3156$$

Step 4: Max Tsallis entropy:

$$H_{\mathrm{ts}}^{\mathsf{max}} \approx \frac{4^{0.5} - 1}{0.5} \approx 2$$

Temperature Scaling — Non-trainable Calibration (2/2)

Normalized confidence (calibrator C_Φ):

$$c_u = C_\Phi(\alpha, \mathbf{H}) = 1 - \frac{H_{\text{ts}}(\mathbf{H}; \alpha)}{H_{\text{ts}}^{\max}(\alpha, |\mathcal{V}|)}$$

$$c \in [0, 1]$$

Example (continued): $\mathcal{V} = \{a, c, k, t\}$
From before:

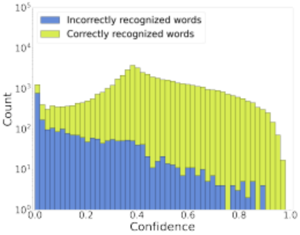
$$H_{\text{ts}} = 1.3156, \quad H_{\text{ts}}^{\max} \approx 2$$

$$c = 1 - \frac{1.3156}{2} \approx 0.3422$$

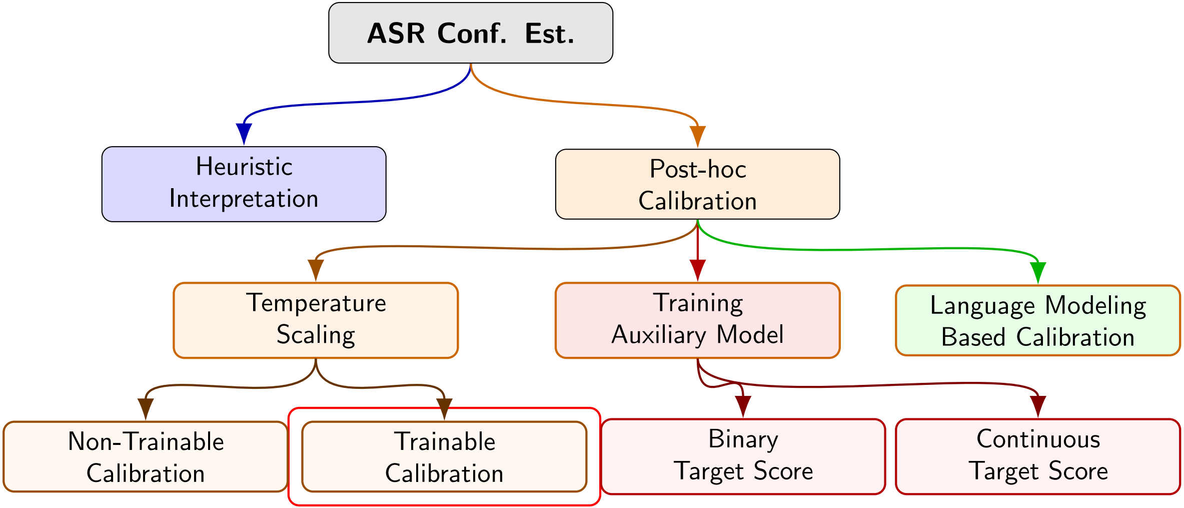
$\rightarrow$ **Confidence of prediction:** $\boxed{0.3422}$

A. Laptev and B. Ginsburg, "Fast Entropy-Based Methods of Word-Level Confidence Estimation for End-to-End Automatic Speech Recognition," *IEEE SLT*, 2022.

- Word-level confidence by aggregating framewise confidence: *min*, *max*, *mean* aggregation
- Tsallis entropy with $\alpha = \frac{1}{4}$, and min aggregation



ASR Uncertainty Estimation



Temperature Scaling — Trainable Calibration (1/2)

- C_Φ uses a **trainable** neural network to predict a token-dependent temperature.
- Input features are extracted from intermediate representations of F_Θ:

$$\mathbf{H}_u = [\mathbf{v}_u, \mathbf{q}_u]$$

  where:
  - $\mathbf{v}_u$ — context vector for token u
  - $\mathbf{q}_u$ — decoder state for token u
- The inverse temperature for token u is:

$$T_u^{-1}(\mathbf{H}_u) = \max\left(0, \mathrm{DNN}(\mathbf{H}_u)\right)$$

Temperature Scaling — Trainable Calibration (2/2)

- DNN is trained with:

$$\mathcal{L}_{\text{NLL}} = -\sum_{u=1}^{|\hat{z}|} \log\left(\text{softmax}\left(\mathbf{l}_u \cdot T_u^{-1}(\mathbf{H}_u)\right)[z_u]\right)$$

where:

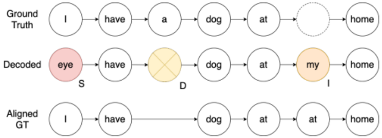
- $\mathbf{l}_u$ — logits for token u
- z_u — ground truth token

- Calibrated confidence scores:

$$c_u = C_\Phi\left(T_u^{-1}(\mathbf{H}_u)\right) = \text{entropy}\left(\text{softmax}\left(\mathbf{l}_u \cdot T_u^{-1}(\mathbf{H}_u)\right)\right)$$

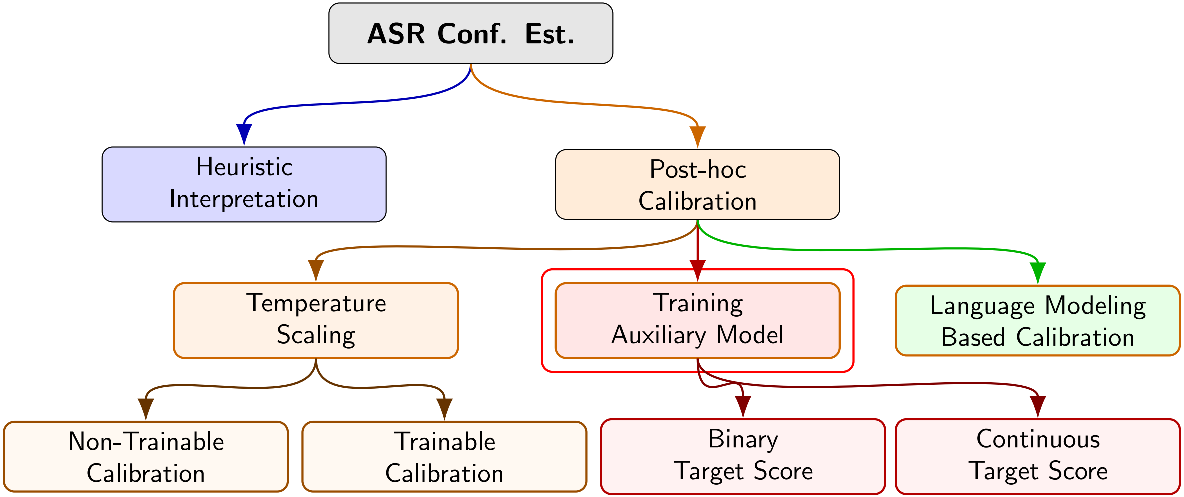
To train DNN:

- Align the ground truth and predicted tokens
- Assign closest correct token for insertions

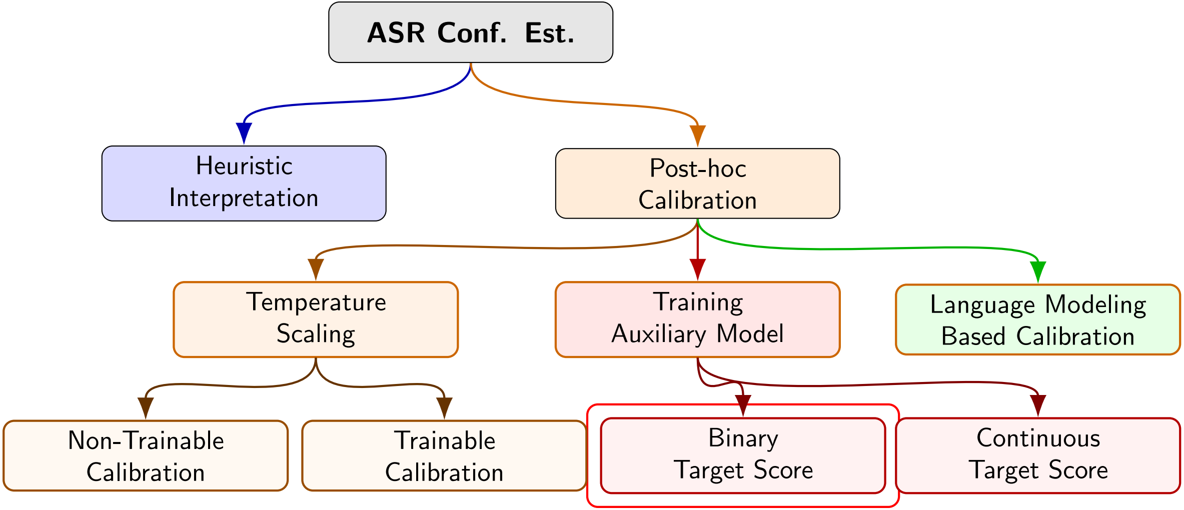


Woodward, A. et al., "Confidence Measures in Encoder-Decoder Models for Speech Recognition",. Proc. Interspeech 2020.

ASR Uncertainty Estimation



ASR Uncertainty Estimation



Auxiliary Model - Binary Target Score (1/3)

Li, Qiujia, et al. "Confidence estimation for attention-based sequence-to-sequence models for speech recognition." ICASSP 2021

- For an input utterance:

$$X = [x_1, x_2, \ldots, x_T]$$

  the encoder produces:

$$\mathbf{e}_{1:T} = \mathrm{ENCODER}(X)$$

- At decoding step u:

$$\mathbf{a}_u = \mathrm{ATTENTION}(\mathbf{a}_{u-1}, \mathbf{d}_{u-1}, \mathbf{e}_{1:T})$$
$$\mathbf{d}_u = \mathrm{DECODER}(\mathbf{a}_u, \mathbf{d}_{u-1}, \mathrm{EMB}(\hat{z}_{u-1}))$$
$$p(\hat{z}_u \mid \hat{z}_{1:u-1}, X) = \mathrm{SOFTMAX}(\mathbf{d}_u)$$

Confidence Estimation Module (CEM) — Prediction (2/3)

- Intermediate representation for token u:

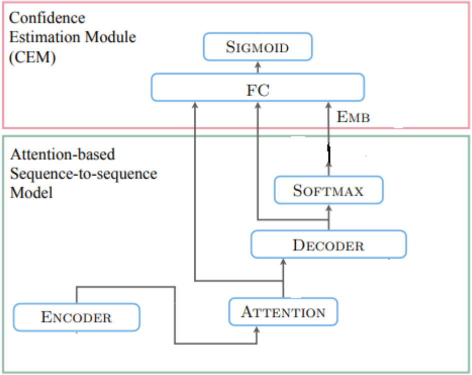
$$\mathbf{H}_u = \left(\mathbf{a}_u, \mathbf{d}_u, \mathrm{EMB}(\hat{z}_u)\right)$$

- Predicted confidence:

$$\hat{c}_u = C_\Phi(\mathbf{H}_u) = \sigma\left(\mathrm{FC}(\mathbf{H}_u)\right)$$

where:

  - $\hat{c}_u \in [0, 1]$ — estimated confidence for token $\hat{z}_u$



Confidence Estimation Module (CEM) — Training (3/3)

- Training objective — Binary Cross-Entropy:

$$\mathcal{L}(c, \hat{c}) = -\frac{1}{U}\sum_{u=1}^{U}\left[c_u \log \hat{c}_u + (1 - c_u)\log(1 - \hat{c}_u)\right]$$

  where:

  - c_u — target confidence label
  - $\hat{c}_u$ — predicted confidence score

- Example target confidence generation - align:

Reference: “A B C D”

Hypothesis: “A C C D”

$$\Rightarrow c = [1, 0, 1, 1]$$

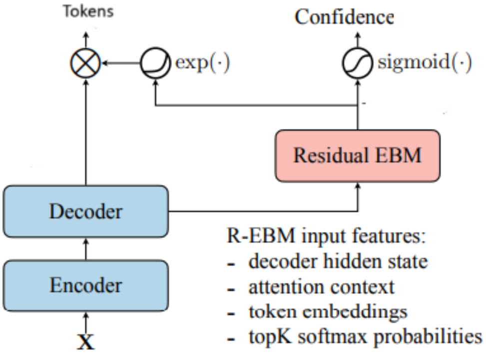
Residual Energy-Based Model (R-EBM) (1/1)

- The R-EBM takes an acoustic sequence $X = [x_1, \ldots, x_T]$ and a hypothesis token sequence $\hat{z} = [\hat{z}_1, \ldots, \hat{z}_U]$ to produce a scalar energy value $-E_\theta(X, \hat{z})$.
- The loss function is:

$$\mathcal{L} \approx \frac{1}{|\mathcal{Y}^+|} \sum_{\hat{z} \in \mathcal{Y}^+} \log \frac{1}{1 + \exp\left(E_\theta(X, \hat{z})\right)} + \frac{1}{|\mathcal{Y}^-|} \sum_{\hat{z} \in \mathcal{Y}^-} \log \frac{1}{1 + \exp\left(-E_\theta(X, \hat{z})\right)}$$

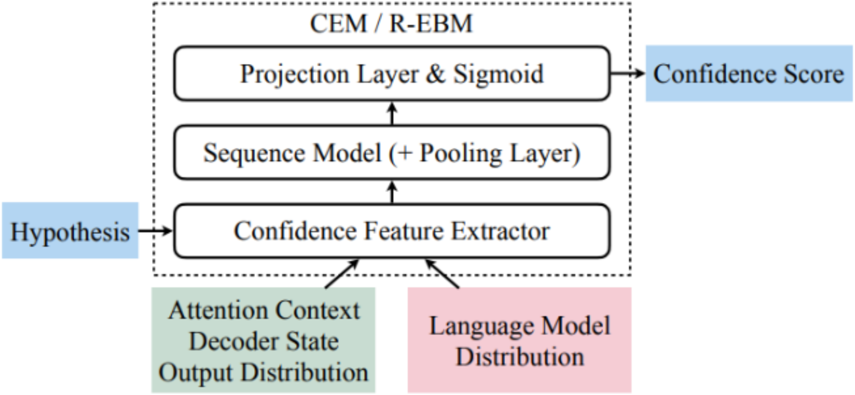
where:

$$\mathcal{Y}^+ \cup \mathcal{Y}^- = \{z, \mathrm{BEAMSEARCH}(X, n)\}$$



Li, Q., et al., "Residual Energy-Based Models for End-to-End Speech Recognition.". Interspeech 2021.

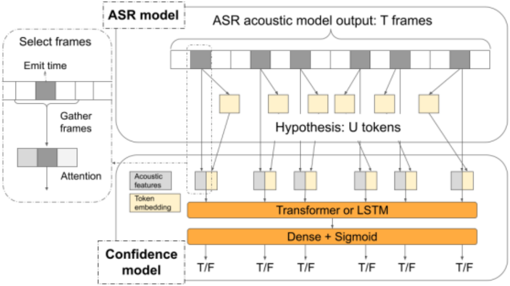
OOD Data (1/1)



Li, Qiujia, et al. "Improving confidence estimation on out-of-domain data for end-to-end speech recognition." ICASSP 2022.

Word-level Confidence Estimation for RNN-T Models (1/2)

- Word-level confidence estimation for RNN-T models
- Form a $2k + 1$ vector of encoder outputs from the left and right context of a subword emission as the acoustic feature
- CEM input Feature formation methods:
  - Attention-based
  - Direct concatenation



Wang, Mingqiu, et al. "Word-level confidence estimation for RNN transducers." 2021 ASRU.

Word-level Confidence Estimation for RNN-T Models (2/2)

- Targets are mapped at both token and word levels.
- Start of words are denoted by special symbols (e.g., ⟨_⟩).

Hypothesis	_lov	-	ely	_son	
Reference	_lo	ve	ly	_son	g
Corresponding targets for confidence model					
Morpheme	False	False	False	True	False
Word	-	-	True	-	False
Word (ours)	True	True	True	False	False

Word-level Confidence Estimation for CTC Model (1/3)

- Let a word w_i be composed of predicted tokens $\hat{z}_{u:u+m}$ from the CTC model:

$$w_i = [\hat{z}_u, \hat{z}_{u+1}, \ldots, \hat{z}_{u+m}]$$

- Define the combined feature vector H for the word w_i as:

$$H = [\mathrm{logits}(\hat{z}_{u:u+m}); \quad \mathrm{softmax}(\hat{z}_{u:u+m}); \quad \rho(w_i); \quad \eta(w_i)]$$

where:

- $\mathrm{logits}(\hat{z}_{u:u+m})$ — logits of the predicted tokens in w_i
- $\mathrm{softmax}(\hat{z}_{u:u+m})$ — softmax probabilities of logits
- $\rho(w_i)$ — summation of one-hot vectors representing each token in w_i
- $\eta(w_i)$ — number of tokens in the word w_i

Word-level Confidence Estimation for CTC Model (2/3)

- The post-hoc CEM model C_Φ takes H as input:

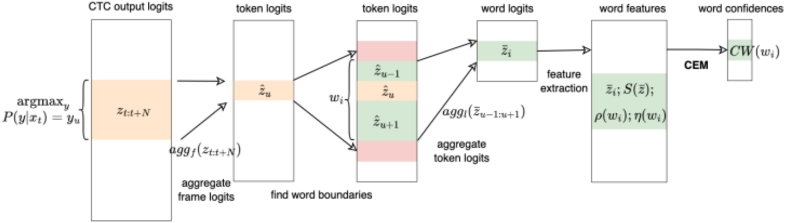
$$\hat{c}_{w_i} = C_\Phi(H)$$

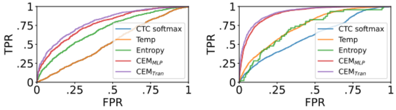
  where:
  - $\hat{c}_{w_i}$ — predicted word-level confidence score
- C_Φ is either MLP or transformer encoder (takes whole sentence as context)
- Binary Cross-Entropy loss for training
- Example target confidence generation:

$$\begin{aligned} \text{Reference:} &\quad \text{“How are you”} \\ \text{Hypothesis:} &\quad \text{“How are ou”} \\ \Rightarrow &\quad c = [1, 1, 0] \end{aligned}$$

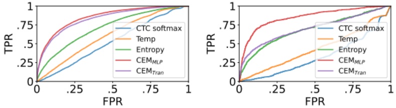
Naowarat, Burin et al., “Word-level confidence estimation for CTC models,” Interspeech, 2023.

Word-level Confidence Estimation for CTC Model (3/3)

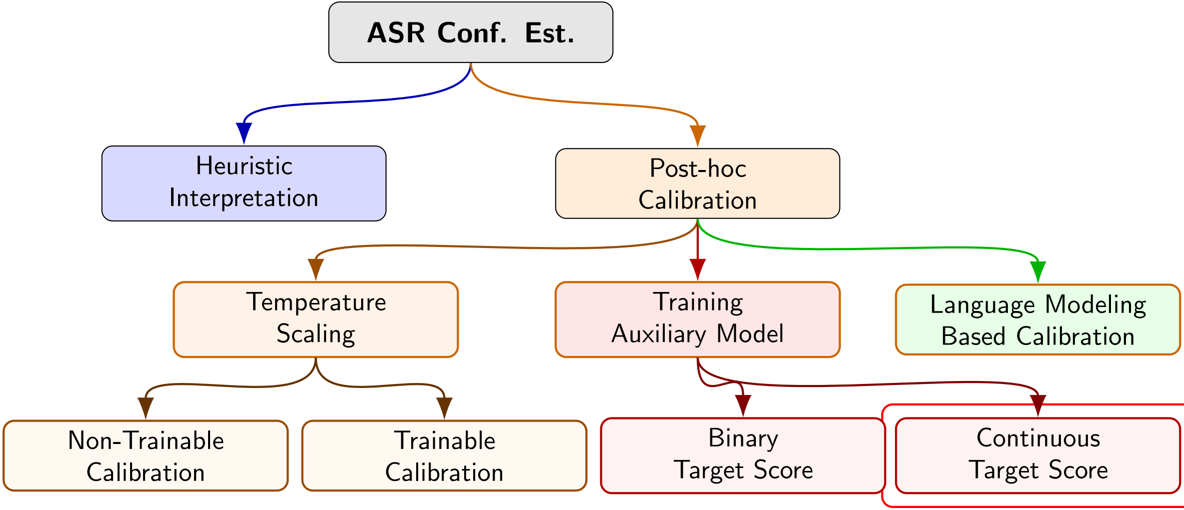




(a) *LibriSpeech test-other* (b) *CommonVoice*

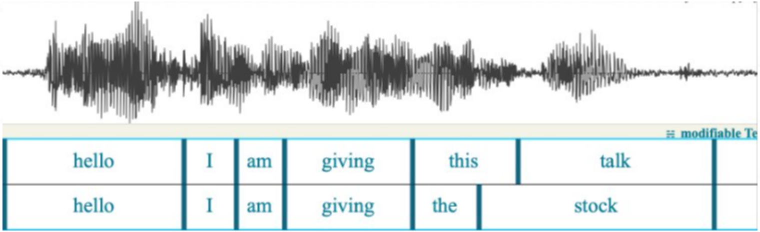


ASR Uncertainty Estimation



TeLeS: Temporal Lexeme Similarity Score (1/6)

N. Ravi, Thishyan Raj T and V. Arora, "TeLeS: Temporal Lexeme Similarity Score to Estimate Confidence in End-to-End ASR," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2024.



The binary target score does not consider the temporal alignment
Ref.: (/the/ /quick/ /brown/ /fox/)
Hyp.: (/the/ /quik/ /bright/ /focks/)
Target Scores: [1, 0, 0, 0]
The second word suffers from a character deletion (`quik`), the auxiliary CEM with binary target score attempts to penalize the estimation and push the score towards zero.

TeLeS: Temporal Lexeme Similarity Score (2/6)

- perform word level alignment between the reference and the hypothesis

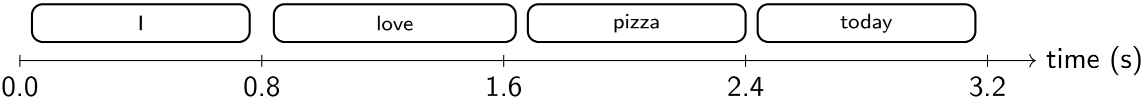
$$G = \textsc{AliGen}(\mathbf{w}, \hat{\mathbf{w}})$$

where $G = [g_1, g_2, \ldots g_P]$ with $g_p \in \{C, S, I, D\}$, which denotes correct, substitution, insertion and deletion respectively.

- the temporal agreement score c^{T} as

$$c^{\mathrm{T}} = \begin{cases} \max\left(0, 1 - \frac{\left|w_i^{ST} - \hat{w}_j^{ST}\right| + \left|w_i^{ET} - \hat{w}_j^{ET}\right|}{\left|w_i^{ET} - w_j^{ST}\right|}\right) & \text{if } g_p = C, S \\ 0 & \text{if } g_p = I, D \end{cases}$$

Where ST denotes start time of the word and ET denotes end time of the word computed using forced alignment.



TeLeS: Temporal Lexeme Similarity Score (3/6)

- When the words are substituted, the words are not necessarily entirely wrong but they may be wrong due to spelling mistakes. Compute the lexical similarity score, c^{L},

$$c^{\mathrm{L}} = \begin{cases} \frac{\left|w_i \cap \hat{w}_j\right|}{\left|w_i \cup \hat{w}_j\right|} & \text{if } g_p = C, S \\ 0 & \text{if } g_p = I, D \end{cases}$$

The TeLeS target score is as follows.

$$c = \begin{cases} \alpha \times c^{\mathrm{L}} + (1-\alpha) \times c^{\mathrm{T}} & \text{if } g_p = C \\ \beta \times c^{\mathrm{L}} + (1-\beta) \times c^{\mathrm{T}} & \text{if } g_p = S \\ 0 & \text{if } g_p = I, D \end{cases}$$

where, $\alpha \in [0,1]$ and $\beta \in [0,1]$ are weights for temporal and lexeme similarity scores and they are tunable hyper-parameters.

TeLeS: Temporal Lexeme Similarity Score (4/6)

Using $\hat{\mathbf{z}}$, for each word $\hat{w}_n$, we find start and end indices of $\hat{w}_n$, $(b_{\hat{w}_n}, l_{\hat{w}_n})$, where $b_{\hat{w}_n}$ is the start index and $l_{\hat{w}_n}$ is the end index. For each $\hat{w}_n$, we can now compute,

$$\bar{\mathbf{a}}_{\hat{w}_n} = \text{mean}(\mathbf{a}_{b_{\hat{w}_n}}, \ldots, \mathbf{a}_{l_{\hat{w}_n}})$$
$$\bar{\mathbf{h}}_{\hat{w}_n} = \text{mean}(\mathbf{h}_{b_{\hat{w}_n}}, \ldots, \mathbf{h}_{l_{\hat{w}_n}})$$
$$\bar{\mathbf{s}}_{\hat{w}_n} = \text{mean}(\mathbf{s}_{b_{\hat{w}_n}}, \ldots, \mathbf{s}_{l_{\hat{w}_n}})$$

The input to the CEM model is as follows.

$$H = [\bar{\mathbf{a}}_{\hat{w}_n}, \bar{\mathbf{h}}_{\hat{w}_n}, \bar{\mathbf{s}}_{\hat{w}_n}]$$

CEM model is as follows.

$$\hat{c} = C_\phi(H, c)$$

TeLeS: Temporal Lexeme Similarity Score (5/6)

Trained using shrinkage loss,

$$\mathcal{L} = \left[\frac{\frac{1}{N'} \sum_{\hat{w}_n \in \hat{\mathbf{w}}} \left(\hat{c}_{\hat{w}_n} - c_{\hat{w}_n} \right)^2 \cdot e^{\hat{c}_{\hat{w}_n}}}{1 + e^{\gamma \cdot \left(\kappa - \frac{1}{N'} \sum_{\hat{w}_n \in \hat{\mathbf{w}}} \left| \hat{c}_{\hat{w}_n} - c_{\hat{w}_n} \right| \right)}} \right]$$

γ and κ are hyper-parameters.

Challenge:

- In ASR confidence estimation, **most words are correct**.
- Severe class imbalance $\Rightarrow$ standard MSE focuses on easy cases and ignores rare errors (substitutions / insertions).

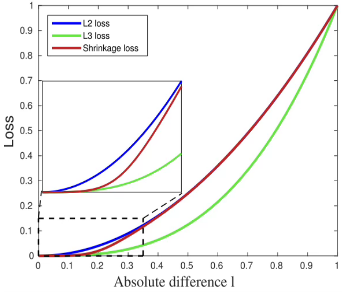
Effect on Imbalance:

- Down-weights/shrinks loss from frequent correct predictions.
- Amplifies loss for rare incorrect predictions.

TeLeS: Temporal Lexeme Similarity Score (6/6)

Effect on Imbalance:

- Down-weights/shrinks loss from frequent correct predictions.
- Amplifies loss for rare incorrect predictions.



Lu, Xiankai, et al.
"Deep object tracking with shrinkage loss."
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020.

Metrics	Class-Prob	Entropy	Binary	TeLeS
MAE ↓	0.5026	0.3998	0.2650	**0.1817**
KLD ↓	0.7912	0.6333	1.1799	**0.1785**
JSD ↓	0.2313	0.1818	0.1920	**0.0467**
RMSE-WCR ↓	0.2883	0.2684	0.2593	**0.1457**
NCE ↑	-0.2641	-0.0100	-0.0055	**0.1363**
ECE ↓	0.2472	0.2253	0.2627	**0.0260**
MCE ↓	0.3904	0.3663	0.4294	**0.1011**

TruCLeS: True Class Lexical Similarity Score (1/5)

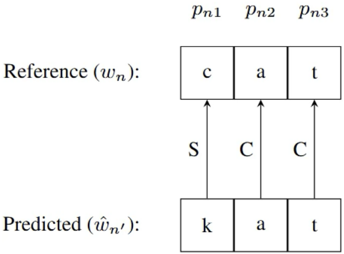
N. Ravi, Thishyan Raj T, Chaganti Ravi Teja and V. Arora, "ASR Confidence Estimation using True Class Lexical Similarity Score," in Interspeech, 2025.

- Perform Levenshtein Alignment between:
    - Predicted word sequence: $\hat{\mathbf{w}}$
    - Reference word sequence: $\mathbf{w}$
- Obtain alignment mapping $g_{n'}$, where n' corresponds to indices in $\hat{\mathbf{w}}$ excluding deletions (words not predicted)
- Interpretation of $g_{n'}$: **C**: Correct match; **S**: Substitution error; **I**: Insertion error

$\mathbf{w}$: cat likes to play
$\hat{\mathbf{w}}$: kat likes to ply
$g_{n'}$: S C C S

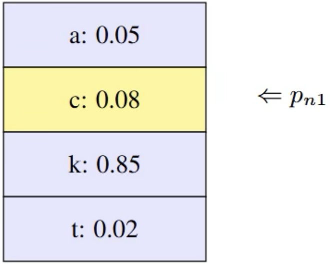
TruCLeS: True Class Lexical Similarity Score (2/5)

- Repeat Levenshtein Alignment at the token level, mapping $\hat{w}_{n'}$ to the reference w_n, where n is obtained from $g_{n'}$
- Denote token-level alignment as $k_{n'j'}$, where:
  - j' indexes tokens of $\hat{w}_{n'}$ (ignores deletions)
  - $k_{n'j'} \in \{\mathrm{C}, \mathrm{S}, \mathrm{I}\}$:
    - **C**: Correct match
    - **S**: Substitution error
    - **I**: Insertion error



- Define the token-level score $\eta_{n'j'}$:

$$\eta_{n'j'} = \begin{cases} p_{nj}, & \text{if } k_{n'j'} \in \{\mathrm{C}, \mathrm{S}\} \\ 0, & \text{if } k_{n'j'} = I \end{cases}$$



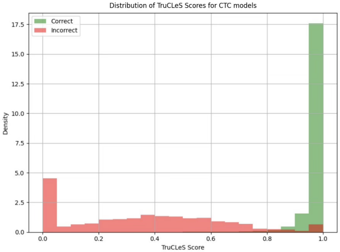
- Here, p_{nj} is the true class probability of the jth token (from the ASR model) aligned with the j'th token in word $w_{n'}$.

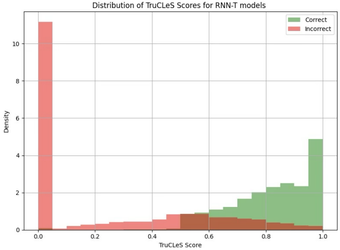
TruCLeS: True Class Lexical Similarity Score (3/5)

Word-level target TruCLeS score, $c_{n'}$ for output words $\hat{w}_{n'}$

$$c_{n'} = \begin{cases} \frac{\sum_{j'} \eta_{n'j'}}{\sum_{j'} 1} \times \delta(\hat{w}_{n'}, w_n) & \text{if } g_{n'} \in \{C, S\} \\ 0 & \text{if } g_{n'} = I \end{cases}$$

where, $\delta(\hat{w}_{n'}, w_n)$ - lexical similarity between $\hat{w}_{n'}$ and w_n





TruCLeS: True Class Lexical Similarity Score (4/5)

Using $\hat{\mathbf{z}}$, for each word $\hat{w}_n$, we find start and end indices of $\hat{w}_n$, $(b_{\hat{w}_n}, l_{\hat{w}_n})$, where $b_{\hat{w}_n}$ is the start index and $l_{\hat{w}_n}$ is the end index. For each $\hat{w}_n$, we can now compute,

$$\bar{\mathbf{a}}_{\hat{w}_n} = \text{mean}(\mathbf{a}_{b_{\hat{w}_n}}, \ldots, \mathbf{a}_{l_{\hat{w}_n}})$$
$$\bar{\mathbf{h}}_{\hat{w}_n} = \text{mean}(\mathbf{h}_{b_{\hat{w}_n}}, \ldots, \mathbf{h}_{l_{\hat{w}_n}})$$
$$\bar{\mathbf{s}}_{\hat{w}_n} = \text{mean}(\mathbf{s}_{b_{\hat{w}_n}}, \ldots, \mathbf{s}_{l_{\hat{w}_n}})$$

The input to the CEM model is as follows.

$$H = [\bar{\mathbf{a}}_{\hat{w}_n}, \bar{\mathbf{h}}_{\hat{w}_n}, \bar{\mathbf{s}}_{\hat{w}_n}]$$

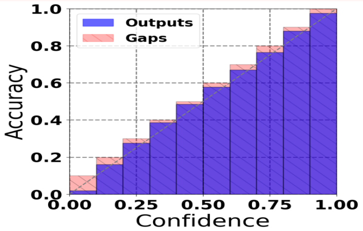
TruCLeS: True Class Lexical Similarity Score (5/5)

CEM model is as follows.

$$\hat{c} = C_\phi(H, c)$$

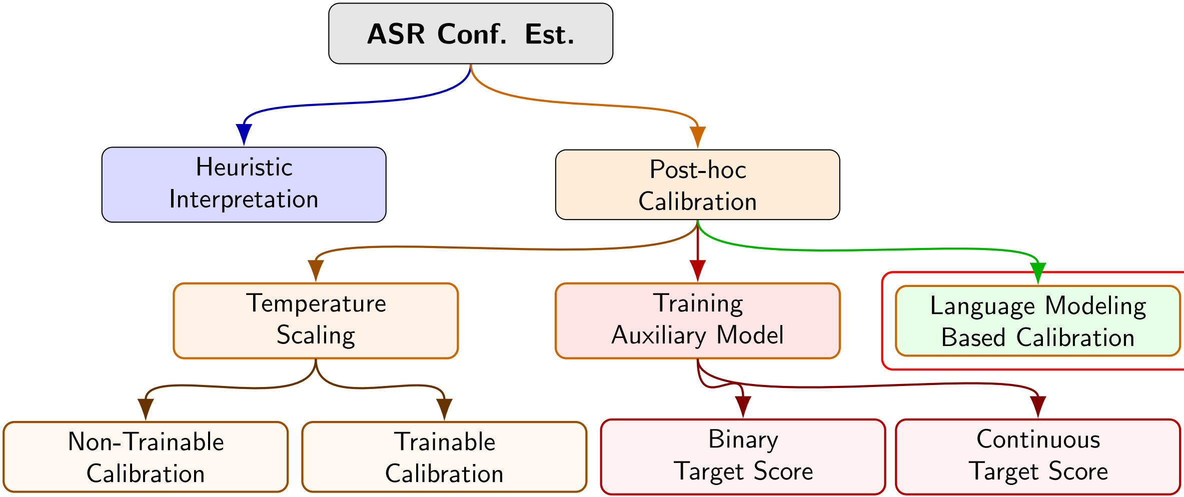
Trained using shrinkage loss,

$$\mathcal{L} = \left[\frac{\frac{1}{N'} \sum_{\hat{w}_n \in \hat{\mathbf{w}}} \left(\hat{c}_{\hat{w}_n} - c_{\hat{w}_n} \right)^2 \cdot e^{\hat{c}_{\hat{w}_n}}}{1 + e^{\gamma \cdot \left(\kappa - \frac{1}{N'} \sum_{\hat{w}_n \in \hat{\mathbf{w}}} |\hat{c}_{\hat{w}_n} - c_{\hat{w}_n}| \right)}} \right]$$



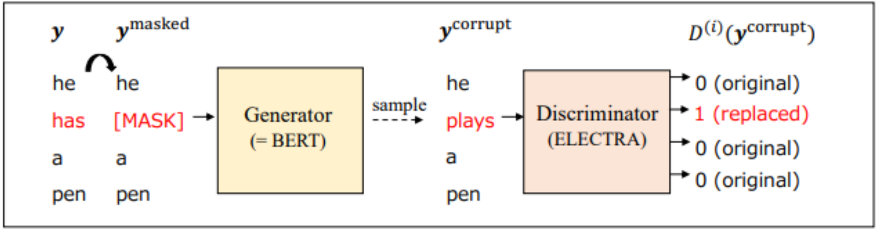
γ and κ are hyper-parameters.

ASR Uncertainty Estimation



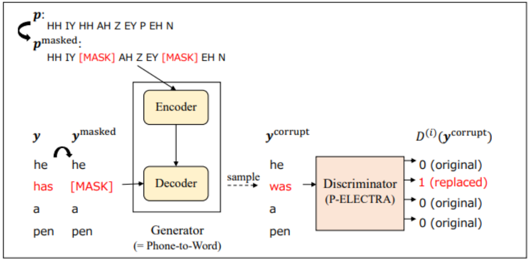
Language Modeling based Calibration: Electra (1/2)

$\boldsymbol{y}$:	_i _don ' t _believe **_ann _knew** _any _magic _or _she ' d _have **_worked** _it _before
$\boldsymbol{y}^{\text{masked}}$:	_i _don ' t _believe [MASK] [MASK] _any _magic _or _she ' d _have [MASK] _it _before
$\boldsymbol{y}^{\text{corrupt}}$ in ELECTRA:	_i _don ' t _believe **_there _is** _any _magic _or _she ' d _have **_used** _it _before
$\boldsymbol{y}^{\text{corrupt}}$ in P-ELECTRA:	_i _don ' t _believe **_and _you** _any _magic _or _she ' d _have **_worked** _it _before

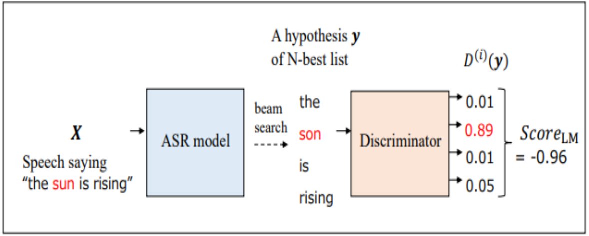


(a) Pre-training of ELECTRA

Language Modeling based Calibration: Electra (2/2)



(b) Pre-training of Phone-attentive ELECTRA (P-ELECTRA)



- **Token-level confidence:**

$$c = 1 - D(y)$$

where $D(y)$ is the discriminator output for token y.

- **Word-level confidence:** Minimum confidence among tokens forming the word:

$$c_{\text{word}} = \min_{y \in \text{token}} c(y)$$

Futami, Hayato, et al. "Asr rescoring and confidence estimation with electra." 2021 ASRU.

Downstream Application - Uncertainty Estimation

Active Learning using Confidence Scores (1/3)

N. Ravi, Thishyan Raj T and V. Arora, "TeLeS: Temporal Lexeme Similarity Score to Estimate Confidence in End-to-End ASR," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2024.

$$\hat{\mathbf{w}} = \mathcal{F}_{\theta}(X_i) \tag{1}$$

$$\hat{c}_{\hat{w}_n} = \mathcal{C}_{\phi}(H) \tag{2}$$

$$A_{\mathrm{X}_i} = \frac{\sum_{n=1}^{|\hat{\mathbf{w}}|} \hat{c}_{\hat{w}_n}}{|\hat{\mathbf{w}}|} \tag{3}$$

$$\hat{\mathrm{X}} = \mathrm{sort}([A_{\mathrm{X}_i}]_{i=1}^{N}) \tag{4}$$

The top samples from $\hat{\mathrm{X}}$ within the labelling budget are annotated and added to the train set. The audio-pseudo-label pairs of the samples with $A_{\mathrm{X}_i} \geq \delta$ (δ is predefined threshold) are added to the train set.

Active Learning using Confidence scores (2/3)

- Active Learning
- 10x faster annotation workflows

Active Learning using Confidence Scores (3/3)

Hindi

Acquired Data Proportion	Random (WER/CER ↓)	Path Prob. (WER/CER ↓)	SMCA (WER/CER ↓)	LMC (WER/CER ↓)	TeLeS (WER/CER ↓)
1/10	38.83/13.84	43.53/15.22	59.97/23.01	92.55/53.16	**32.96/11.09**
1/7	33.97/12.97	31.71/10.59	33.09/11.00	36.93/12.42	**28.78/9.51**

Tamil

Acquired Data Proportion	Random (WER/CER ↓)	Path Prob. (WER/CER ↓)	SMCA (WER/CER ↓)	LMC (WER/CER ↓)	TeLeS (WER/CER ↓)
1/10	52.92/12.98	50.05/10.52	49.63/10.43	50.64/10.71	**47.38/9.83**
1/7	49.90/10.43	45.62/9.28	45.11/9.13	45.49/9.19	**44.35/8.87**

Kannada

Acquired Data Proportion	Random (WER/CER ↓)	Path Prob. (WER/CER ↓)	SMCA (WER/CER ↓)	LMC (WER/CER ↓)	TeLeS (WER/CER ↓)
1/10	58.74/13.56	54.29/12.22	99.97/85.58	99.95/93.97	**47.98/10.71**
1/7	49.84/11.62	43.75/9.57	60.87/13.67	74.47/17.69	**41.91/9.15**

- Random: Selects random audio samples.
- Length-normalized path probability
- SMCA: Based on different models with dropout.
- LMC: Based on 3-gram LM.

Mvaak demo

- Demo video URL - Mvaak tool
- For further details visit - Link

Thank you