Automatic Assessment of Mother Tongue Reading Ability

Shreyas Premkhede, Piyush Kothawade, Soham Jiddewar, Prathamesh Kadam

Automatic Speech Recognition Model
Marathi group
Indian Institute of Technology
Hyderabad, India

June 26, 2024

Aim of the Project

- We aspire to develop an app that would test analyze the speech of children in their regional languages.
- To make it more interactive, we would display the accuracy fluency scores of the input speech out of 100, whilst also displaying the weak points(e.g. specific vowels/consonants) where the user struggles.

What are we working on..

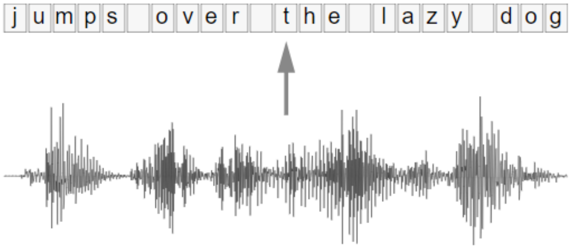
- We are using a SPRING-INX ASR Fine-tuned model(trained on the *Marathi language*).
- Till now, we have been able to do the forced alignment of the input audio with the transcript.
- We have also tested the model on varied sentences(towards the end of this ppt.), testing the confidence scores getting the expected results too!

Difficulties faced

- Since a lot of us were new to this field in speech processing, we took some time reading articles to understand what we were actually working on.
- We used the *Forced Alignment in Wave2Vec2* colab as a reference. We had to tackle some issues later too, since the alignment of the input speech with the time frames was a bit ambiguous.

Brief Explanation of the CTC Algorithm

- What are we doing?
  - We want to map input audio to the transcript
  - We use the CTC algorithm to map these audio inputs to transcripts.



Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

Brief Explanation of Working of Model

- What are the major steps?
  - We are mapping X (Audio input).

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

  - To Y (Transcript)

$$Y = [y_1, y_2, \ldots, y_U]$$

- How is CTC better than other algorithms?
  - It overcomes the challenges of varying inputs and outputs.
  - Gives distribution over all Y's, so we can infer which Y is most probable.

Brief Explanation of Working of Model

- Some properties of CTC
  - 'ϵ' token which is a blank token. Used to denote silence, the distinction between characters in the script (Could be the same character).

$$Z = [\epsilon, y_1, \epsilon, y_2, \ldots, \epsilon, y_U, \epsilon]$$

  - We'll divide CTC into two broad categories,
    - Loss Function
    - Inference

Forward Propagation

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

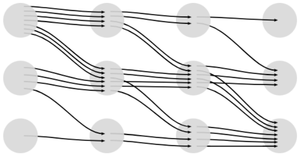
The CTC conditional **probability**

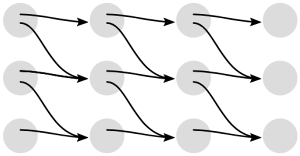
marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

- We need to use Dynamic programming to calculate these per-time step probabilities because a number of possible Y's is exponential.
- Example to show we can use DP: Consider the word "there". Now for the first three letters that is "the" we can get it from sequences like ϵ + t + ϵ + h + ϵ + e or t + h + ϵ + ϵ + e.

Brief Explanation of Working of Model


Summing over all alignments can be very expensive.


Dynamic programming merges alignments, so it's much faster.

- We define a new 'forward' variable called α. Which is the probability of the sequence from 1^{st} character to the s^{th} character of the output transcript being tracked by the algorithm at time step t.
- With the help of this alpha variable, we will have a whole grid of probabilities and we can set some rules to find possible answers.

Brief Explanation of Working of Model

- Such rules will help us in finding paths in a faster and more efficient way.
- All these probabilities are added to get the probability of Y.

$$\sum_{(X,Y)\in D} -\log(p(Y|X))$$

- Here, D is the set of all X's and Y's used for training the model.

Greedy Decoding vs Beam Search

- The greedy approach would be taking all scores that are best for the current time step but this approach is flawed.

$$A^* = \arg\max_A \prod_{t=1}^{T} p_t(a_t|X)$$

Greedy Decoding vs Beam Search

- Example: the alignments [a, a, ϵ] and [a, a, a] may have a lower probability than [b, b, b]. But the sum of their probabilities may be greater than that of [b, b, b]. The naive approach will incorrectly propose $Y = [b]$. It should have chosen $Y = [a]$.
- To infer over multiple alignments, we will use the beam search algorithm.

Brief Explanation of Working of Model

- Properties of CTC:
  - Alignment-free:
    - Monotonic
    - Many-one
    - Y should be smaller than X
    - No mapping from X to Y for single audio signals.
  - Conditional independence:

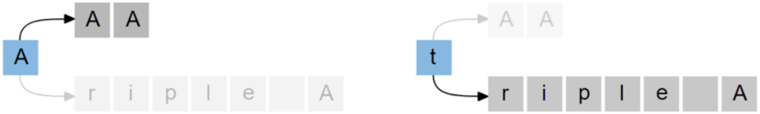


Figure: (a) First letter 'A' (b) First letter 't'
Probability Of "AA" and "riple A" are independent of the first letter.

Emission Matrix

- An emission matrix specifies the probability of observing a particular output given a specific state.
- During the decoding process, the emission matrix is used to calculate the probability of the observed data sequences given the current state. These probabilities are used in conjunction with the transition probabilities (probabilities of moving from one state to another) to update the trellis matrix.

Trellis Matrix

- **Purpose:** The trellis matrix helps to keep track of the possible state sequences and their associated probabilities at each time step.
- **Function:** Each cell in the matrix represents the highest probability of being in a particular state at a particular time step, given the observations up to that time step.

Example Workflow

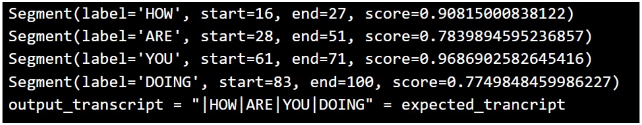
- **Audio Input:** A sentence like "Hello world" is input as raw audio.
- **Feature Extraction with wav2vec 2.0:** The audio is processed to produce feature vectors at each time step.
- **Constructing Emission Matrix:** For each feature vector, calculate the probability of each possible phoneme (e.g., /h/, /e/, /l/, /o/, etc.). Populate the emission matrix with these probabilities.
- **Dynamic Programming:** Use the Viterbi algorithm to traverse the trellis matrix, utilizing the emission matrix to compute the most likely alignment path.
- **Alignment Result:** The final output is the alignment of the audio with the phonemes "h-e-l-l-o w-o-r-l-d", showing precisely where each phoneme occurs in the audio.

Testing the model

- The model, as expected, gave high confidence scores for correctly pronounced sentences.
- Meanwhile, for mispronounced words, the model gave slightly lower confidence scores, again, as expected.
- We also tested the model against sentences with noisy background, expecting the confidence scores to be lower.
- We also tried perturbing the speech rate of the sentences, with lower confidence scores for higher speech rates.

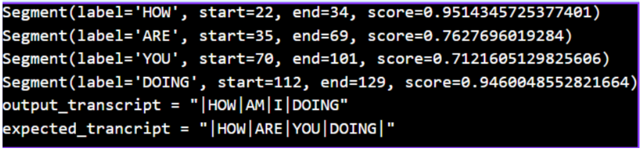
Sentences pronounced correctly and incorrectly

Given the word level confidence scores :



```
Segment(label='HOW', start=16, end=27, score=0.90815000838122)
Segment(label='ARE', start=28, end=51, score=0.7839894595236857)
Segment(label='YOU', start=61, end=71, score=0.9686902582645416)
Segment(label='DOING', start=83, end=100, score=0.7749848459986227)
output_transcript = "|HOW|ARE|YOU|DOING" = expected_trancript
```

Figure: Sentence Pronounced Correctly



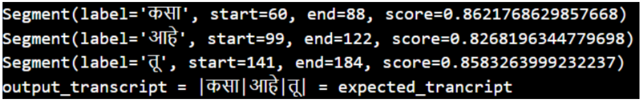
```
Segment(label='HOW', start=22, end=34, score=0.9514345725377401)
Segment(label='ARE', start=35, end=69, score=0.7627696019284)
Segment(label='YOU', start=70, end=101, score=0.7121605129825606)
Segment(label='DOING', start=112, end=129, score=0.9460048552821664)
output_transcript = "|HOW|AM|I|DOING"
expected_trancript = "|HOW|ARE|YOU|DOING|"
```

Figure: Sentence Pronounced Incorrectly

Observations: The confidence scores for the words 'ARE' 'YOU' have lowered in the second case, which is expected since the 'AM' 'I' was fed into the model instead.

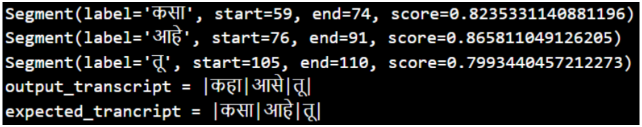
Sentences pronounced correctly and incorrectly

Given the word level confidence scores :



```
Segment(label='कसा', start=60, end=88, score=0.8621768629857668)
Segment(label='आहे', start=99, end=122, score=0.8268196344779698)
Segment(label='तू', start=141, end=184, score=0.8583263999232237)
output_transcript = |कसा|आहे|तू| = expected_trancript
```

Figure: Sentence Pronounced Correctly



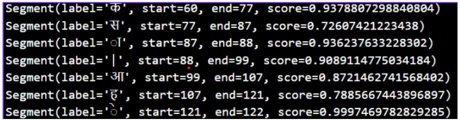
```
Segment(label='कसा', start=59, end=74, score=0.8235331140881196)
Segment(label='आहे', start=76, end=91, score=0.865811049126205)
Segment(label='तू', start=105, end=110, score=0.7993440457212273)
output_transcript = |कहा|आसे|तू|
expected_trancript = |कसा|आहे|तू|
```

Figure: Sentence Pronounced Incorrectly

Observations: Since only the first two words were pronounced incorrectly, I was expecting the confidence level scores to decrease for both, but it only decreased for the first word, but increased for the second word. Analyzing it further(the character-level scores:)

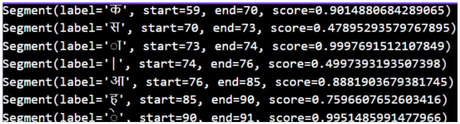
Anomaly in the previous example

Given the word level confidence scores :



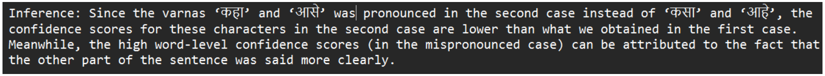
```
Segment(label='क', start=60, end=77, score=0.9378807298840804)
Segment(label='स', start=77, end=87, score=0.72607421223438)
Segment(label='ा', start=87, end=88, score=0.93623763228302)
Segment(label='|', start=88, end=99, score=0.9089114775034184)
Segment(label='आ', start=99, end=107, score=0.8721462741568402)
Segment(label='ह', start=107, end=121, score=0.7885667443896897)
Segment(label='े', start=121, end=122, score=0.9997469782829285)
```

Figure: Sentence Pronounced Correctly



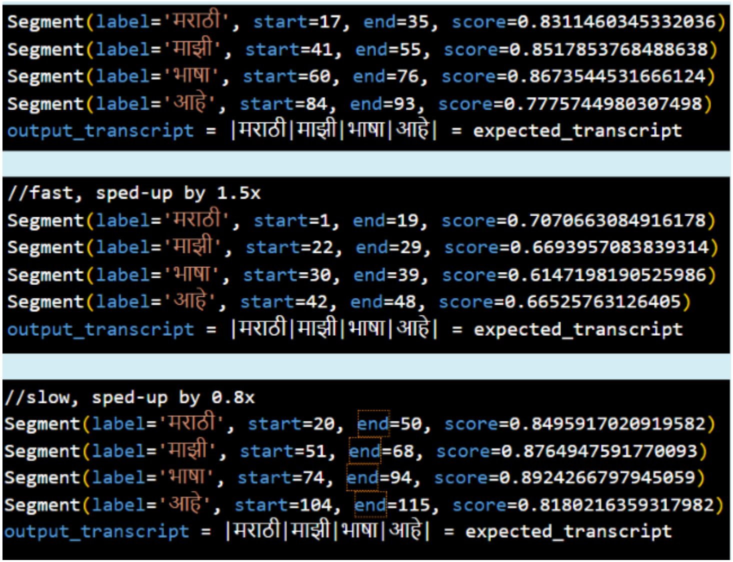
```
Segment(label='क', start=59, end=70, score=0.9014880684289065)
Segment(label='स', start=70, end=73, score=0.47895293579767895)
Segment(label='ा', start=73, end=74, score=0.9997691512107849)
Segment(label='|', start=74, end=76, score=0.4997393193507398)
Segment(label='आ', start=76, end=85, score=0.8881903679381745)
Segment(label='ह', start=85, end=90, score=0.7596607652603416)
Segment(label='े', start=90, end=91, score=0.9951485991477966)
```

Figure: Sentence Pronounced Incorrectly



Inference: Since the varnas 'कहा' and 'आसे' was pronounced in the second case instead of 'कसा' and 'आहे', the confidence scores for these characters in the second case are lower than what we obtained in the first case. Meanwhile, the high word-level confidence scores (in the mispronounced case) can be attributed to the fact that the other part of the sentence was said more clearly.

Sentence with added perturbation



```
Segment(label='मराठी', start=17, end=35, score=0.8311460345332036)
Segment(label='माझी', start=41, end=55, score=0.8517853768488638)
Segment(label='भाषा', start=60, end=76, score=0.8673544531666124)
Segment(label='आहे', start=84, end=93, score=0.7775744980307498)
output_transcript = |मराठी|माझी|भाषा|आहे| = expected_transcript
```

```
//fast, sped-up by 1.5x
Segment(label='मराठी', start=1, end=19, score=0.7070663084916178)
Segment(label='माझी', start=22, end=29, score=0.6693957083839314)
Segment(label='भाषा', start=30, end=39, score=0.6147198190525986)
Segment(label='आहे', start=42, end=48, score=0.66525763126405)
output_transcript = |मराठी|माझी|भाषा|आहे| = expected_transcript
```

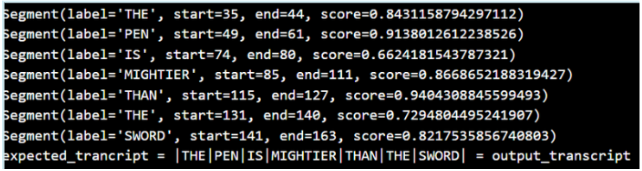
```
//slow, sped-up by 0.8x
Segment(label='मराठी', start=20, end=50, score=0.8495917020919582)
Segment(label='माझी', start=51, end=68, score=0.8764947591770093)
Segment(label='भाषा', start=74, end=94, score=0.8924266797945059)
Segment(label='आहे', start=104, end=115, score=0.8180216359317982)
output_transcript = |मराठी|माझी|भाषा|आहे| = expected_transcript
```

Observations from the previous testcase

- I was expecting the confidence scores to lessen in the second case when I had sped-up the input audio by 1.5x.
  - Reason: Faster speech can blur the transitions between phonemes, making it more challenging for the model to distinguish between them clearly.
  - The model is trained on normal speech rates, so the compressed features at 1.5x speed do not match well with what the model has learned.
- Meanwhile, in the third case, when I had slowed down the input audio by 0.8x, I was instead expecting the confidence scores to be better (for all the words).
  - Reason: Slower speech often results in clearer recognization of phonemes and words, making it easier for the model to distinguish between them.

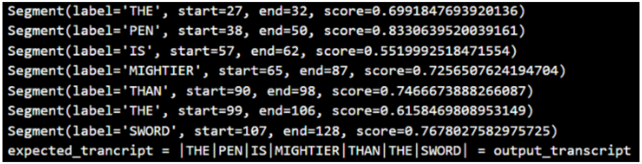
Sentences with/without noisy background

Given the word level confidence scores :



```
Segment(label='THE', start=35, end=44, score=0.8431158794297112)
Segment(label='PEN', start=49, end=61, score=0.9138012612238526)
Segment(label='IS', start=74, end=80, score=0.6624181543787321)
Segment(label='MIGHTIER', start=85, end=111, score=0.8668652188319427)
Segment(label='THAN', start=115, end=127, score=0.9404308845599493)
Segment(label='THE', start=131, end=140, score=0.7294804495241907)
Segment(label='SWORD', start=141, end=163, score=0.8217535856740803)
expected_trancript = |THE|PEN|IS|MIGHTIER|THAN|THE|SWORD| = output_transcript
```

Figure: Sentence Spoken in clear background



```
Segment(label='THE', start=27, end=32, score=0.6991847693920136)
Segment(label='PEN', start=38, end=50, score=0.8330639520039161)
Segment(label='IS', start=57, end=62, score=0.5519992518471554)
Segment(label='MIGHTIER', start=65, end=87, score=0.7256507624194704)
Segment(label='THAN', start=90, end=98, score=0.7466673888266087)
Segment(label='THE', start=99, end=106, score=0.6158469808953149)
Segment(label='SWORD', start=107, end=128, score=0.7678027582975725)
expected_trancript = |THE|PEN|IS|MIGHTIER|THAN|THE|SWORD| = output_transcript
```

Figure: Sentence Spoken in noisy background

Observations from the previous test cases

- In the first case (in a clear background), I was expecting the confidence scores to be higher than the second case.
  - Reason: Since the model can easily identify phonetic and acoustic features, the model can align these clear features with its learned patterns more accurately, leading to higher confidence scores.
- Meanwhile, in the second case: In a noisy environment, the speech signal is mixed with background noise, making it harder to distinguish the speech from the noise.
  - This makes it harder to align the noisy features with the learned patterns, leading to lower confidence in its predictions.

Thank you