Explainability & Localization Write Up

Team sheep(Yang Xu)

1. Methodology

Our method is based on OpenAI's Whisper model, enhanced with a modified loss function that verifies the accuracy of each token. This approach allows us to quickly identify erroneous words. Next, we will demonstrate our method in three steps: tokenization, the model and loss function, and prediction and localization.

1.1 Tokenization

First, we tokenize the text input using Whisper's tokenizer and insert special tokens to ensure that the model behaves similarly to an ASR task. Rather than generating tokens step by step or using beam search, we directly check whether each word in the expected text—namely, what the child is asked to say—is pronounced correctly. Figure 1 illustrates an example of the tokenization process.

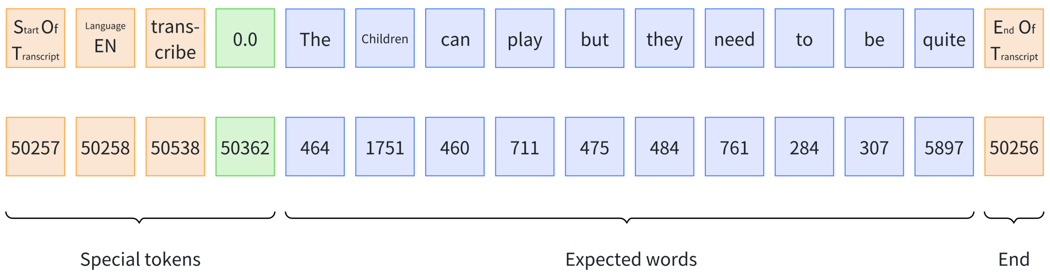

Figure 1, tokenization, whisper using 4 special tokens, follows expected text, end with EOT token.

1.2 Model & loss function.

Our model is identical to HuggingFace's `WhisperForConditionalGeneration`. It takes audio input and a list of tokens as conditions, generating logits of shape `1 x vocab_size` for each token. This results in an overall logits matrix of shape `vocab_size x n_token`. For example, if a child says "I will go home" but the expected text is "I will go mall," feeding the expected tokens into the model yields logits of shape `vocab_size x n_token`. Because the first three words match correctly, we see high confidence at their respective token indices. However, for the fourth word, the logits show higher confidence at the index for "home" rather than "mall."

To adapt the model for overall correctness checking, we design a custom loss function. First, we extract the logits corresponding to the expected text (`1 x n_token`) from the model's conditional output (`vocab_size x n_token`). We then apply a binary cross-entropy (BCE) loss, treating it as a binary classification problem. This approach fine-tunes the ASR training process to focus on correctness.

During the original ASR training, a cross-entropy (CE) loss is applied to the logits (`vocab_size x n_token`) because the correct tokens are known. In our task, however, we only know whether the entire sentence is correct, not the specific tokens. Therefore, we aggregate the logits of the expected text (`1 x n_token`) into a single `1 × 1` value and apply BCE loss. This ensures the model is optimized to detect correct or incorrect pronunciations at the sentence level.

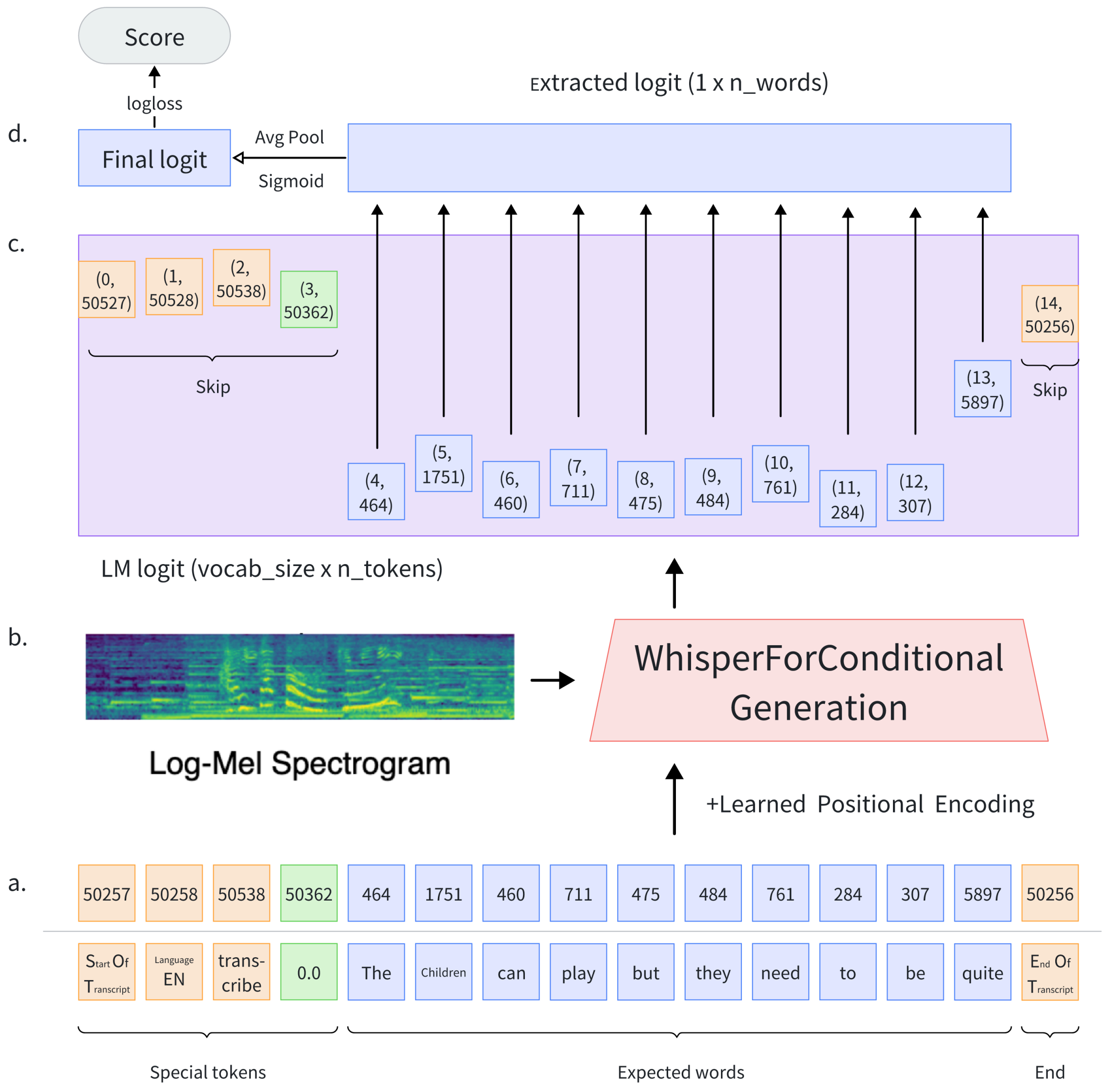

Figure 2. Model and Loss Function Design. (a) Input tokens: The tokens provided to the model. (b) Modeling procedure: The log-mel spectrogram and tokens (with positional encoding) are fed into the WhisperForConditionalGeneration model. (c) Logit extraction: For each non-special token, retrieve the logit at (index, token_id) and store it in an extracted logit tensor. This step yields the logit for each token. (d) Logit aggregation: Apply mean pooling to obtain a single final logit, then use a binary cross-entropy (BCE) loss with the binary label (score).

1.3 Prediction & localization

As mentioned above, we extract the logits of the `expected_text` and aggregate them for our overall prediction. If we wish to localize the incorrect position, we first examine the logits (1 × *n_token*) and look for the first low value, which indicates a potentially mispronounced word. Since each audio clip is quite short, this word-level localization is sufficient. Figure 3 presents four examples illustrating accurate error localization. Figures 3(a) and 3(b) involve sentence repetition, while Figures 3(c) and 3(d) focus on nonword repetition. In the first audio sample (*jvnyso.wav*), the child mispronounces "children" and omits "but." We observe low scores for both "children" and "but," resulting in low-value regions near the start of the clip. In the second audio (*jvrgdy.wav*), the child changes the phrase "...friends during the movie" to "... friends when he watch the movie." For nonword repetition, the model similarly pinpoints mispronunciations. In the first example (*aaajvs.wav*), the child pronounces the word correctly at first but then shifts to an incorrect form after "toovh." In another example (*aadyvd.wav*), the model detects a wrong phoneme in the middle, where "kahybeymookdowb" becomes "kahymeymookdowb." These cases demonstrate that our approach can accurately localize the words where the pronunciation deviates from the expected text.

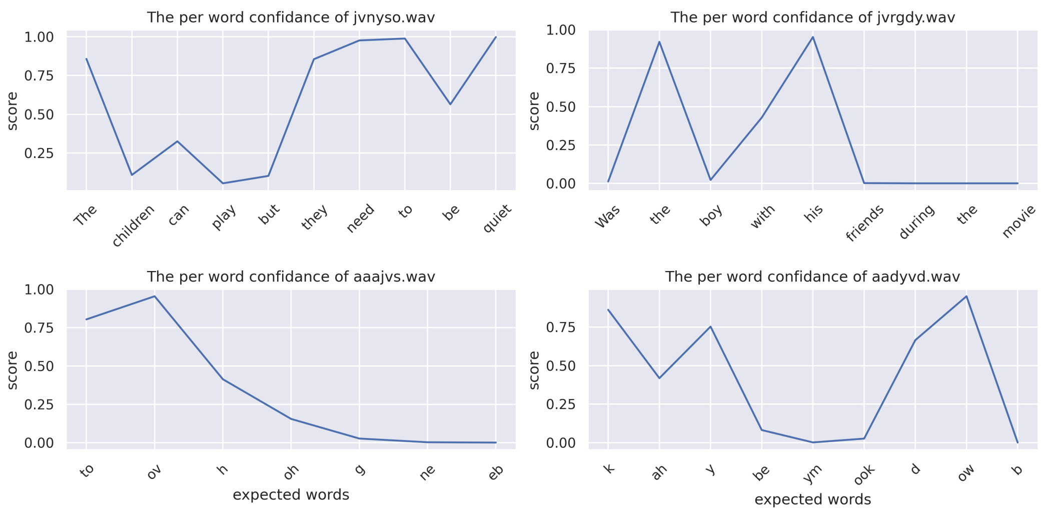

Figure 3. Visualization of per-token scores. The top-left and top-right panels show two cases of sentence repetition with errors in different positions. The bottom-left and bottom-right panels depict nonword cases.

To further locate the timestamp, we can use package whisperX, figure 4 shows a result with WhisperX's word segmentation and timestamp tagged.

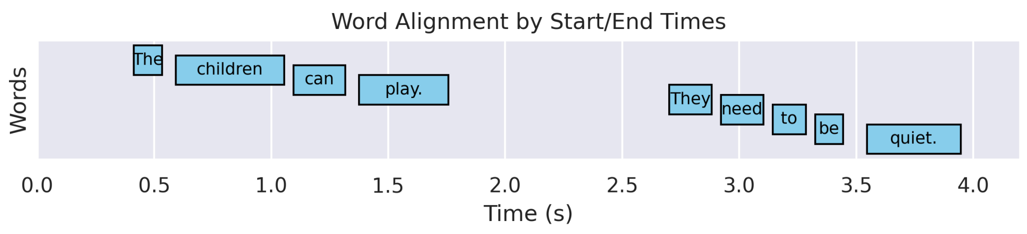

Figure 4. the timestamp segmentation of jvnyso.wav.