Backend Design

---- Group 5: Yoyo D., Jiatong S., Nan H., Gary X., Rafa de la TO

We can split our backend development for this period into three stages, including algorithmic refinement, front-back interaction, and scoring specifics.

Algorithmic Refinement

For algorithmic improvement, we defined a new GOP scoring metric, namely Salient GOP. According to the definition of the GOP, the GOP is not in a determined range. So it leaves a deficiency for the continuous phonetic score. Next, a human generally generates speech continuously. So for acoustic modeling, the Chain acoustic model mainly detects the phonetic spikes instead of the whole segment of the phoneme. Given that we need to align all the phoneme to the utterances, decoding alignment is not as promised. The Salient GOP is designed to solve the two problems as follows:

$$sGOP(p|O) = Max_{o\in(\frac{1}{4}L(O),\frac{3}{4}L(O))}|\frac{log(Max_{q\in Q}P(o|q)}{log(P(o|p)}|$$

Since the DNN-HMM model cannot directly compute the segmental level P(O|p), the segmental level posterior probabilities are represented by the mean of the frame-level probabilities. With the fact that the Chain acoustic model is based on phoneme spike (in other words, introduce the “blank” for each bi-phone unit). Hence, the maximum posterior probability from the middle part of the phoneme (¼, ¾) is applied to stand for the P(O|p). This process can also reduce the misalignment. Next, the denominator and the numerator are exchanged with their logarithm forms. Under this transformation, the Formula still contains the previous information but confined into an [0, 1] interval.

The Pronunciation Scoring Experiments

Method-Rater	Pearson-1	Pearson-2	Spearman-1	Spearman-2	MIC-1	MIC-2
Rater 1	\	0.573	\	0.573	\	0.276
GOP	0.425	**0.297**	0.370	**0.315**	0.182	0.143
SGOP	**0.452**	0.287	**0.409**	0.304	**0.212**	**0.173**

We still take the CALL_2K dataset for the evaluation of the Salient GOP. The result is as shown in the table. From the correlation indexes between the two raters, it can be found that the two raters generally have consensus on the whole corpus. The table shows that the SGOP method is more similar to the Rater 1 while the GOP

method is more like Rater 2. There is an about 3% improvement observed from the Salient GOP based on all the indexes comparing to the Rater 1. The GOP only outperforms in 1% on linear measures for the Rater2, but it is still worse than SGOP when considering the MIC. Generally, the Salient GOP performs better than the GOP method.

Front-back Interaction

Our frontend is developed on Django server. We mainly deploy our backend to the frontend as a python package. To use the model, the server firstly initial the backend “Score” object that defines the configuration for the scoring (i.e. lexicon information, phonetic information, and Kaldi workspace that contains models and Kaldi environment. As the server got the request of test utterances, it would call the scoring function. Since our model is trained on Kaldi which has specific definitions for its models (e.g. the HMM structure, the phonetic decision tree for triphones, the i-vector extractor, the TDNN model, and the lattice finite state transducer), we generally encode Kaldi environment as our backend environment. Because of this issue, the Scoring function would do two jobs in parallel, including the post-probability computing and the text converter.

Post-probability computing will first ask the model to create a Kaldi test environment and then specify an extract-post script, including feature extraction and preprocessing (e.g. CMVN). Next, we also arrange a lattice from senome to phoneme because the prediction of the TDNN would be in senome format. At last, we feed the features into the TDNN for inference and pass the predictions to the prepared lattice.

The text converter is rather easy. For now, we do not consider the multi-pronunciation issue for words, so the main task for the converter is to look up the dictionary (i.e. lexicon).

Score Specifics

We present our result to the server in a detailed JSON format including the duration information, the pronunciation information, the fluency information, and the integration information. The duration information includes all the starts and ends of the phonemes, the words, and the whole sentences. So does the pronunciation information. The sentences and words pronunciation is scaled by vowels and consonants to balance the importance. The fluency information is generated via optional silence. The integration information is pronunciation information after quantization.

Shortcomings and Future direction

For this period, we mainly achieved the interaction between the backend and frontend. Besides, we also implemented more scoring metrics (i.e. fluency, integration) to give the speaker more concrete suggestions. However, most of the previously identified shortcomings are not addressed now. So we still have much potential for a better system. Just for recap, we still have the following backend directions.

- Multi-pronunciation
- Fixed Sentences
- More Scoring Dimensions
- History Analysis
- No Further Improvement Instructions

In addition to the algorithmic aspect, we also spot some systematic issues. An essential one of the issues is that our low inference time. For now, we need 1to1 time for scoring, that is if we have 3 seconds utterance, we need to have 3 seconds for model inference. Though, we have already make several implements to address the problem (e.g. adopting Chain model, use less HMM states, minimize the lattice, etc.), it still comes out to be a problem.

To tackle each of the problems, we think about the following future development directions:

- Multi-pronunciation: add multiple paths in the decoding graph to support the different pronunciation or ask the user to state which style they want to learn first.
- Fixed Sentences: add streaming asr to decode and split the speech first and then use the CALL system to score the speech. The method asks us to build a more robust ASR to recognize what users are saying other than current ASR which is adopted as a standard.
- Scoring Dimensions: add more models (e.g. prosody model) to enable more features.
- History Analysis: run a database that stores the history information of the users and create some metrics for a user profile to support historical analysis.
- Future Improvement Instructions: deploy a TTS(text-to-speech) module for speech synthesis that can synthesize examples (we also think about using voice conversion to present the speech examples in the user's own voice)

For the inference efficiency, we found that the most time is consumed in the lattice construction, We are finding ways to take benefit from computing the lattice before the test utterances.