RIVUSVOX EDITOR: Near-live zero-shot adaptive voice editing

EXECUTIVE SUMMARY

Team: Francesca Balestrieri, Zack Bezemek

GitHub repository: https://github.com/bezemekz/RivusVoxEditor

Background and project overview

The main goal behind our project was to be able to edit speech on the fly, by removing key words and replacing them with other words with a generated voice as similar to the speaker's voice as possible, all while having no prior knowledge about the speaker.

Our objectives: We had the following two main objectives for our project:

- to improve the inference time and the speech editing mechanics of one the current state-of-the-art speech editing models (FluentSpeech);
- using our improved model, to be able to edit speech during a live streaming with a maximum fixed delay of possibly 10s.

Stakeholders: Potential stakeholders include anyone involved in live streaming or live television (due to the live streaming speech editing app that we have developed), and video and sound editors for what concerns just the speech editing model.

Modelling approach

Speech editing model: We took as our base model the FluentSpeech model developed by Ziyue Jiang, Qian Yang, Jialong Zuo, Zhenhui Ye, Rongjie Huang, Yi Ren, and Zhou Zhao, which is a automatic speech editing architecture using a context-aware diffusion models to iteratively refine the edited mel-spectrograms conditional on context features (see https://github.com/Zain-Jiang/Speech-Editing-Toolkit and https://arxiv.org/abs/2305.13612); for our base model, we ignore the stutter prediction components.

We made the following modifications to the base model:

- The base model uses the Montreal Forced Aligner (MFA) for the alignment of the mel spectrogram to the transcript. We found that the alignment timestamps of MFA are often not very precise, and we decided to use instead a modified version of WhisperX (with wav2vec2-xlsr-53-espeak-cv-ft as the aligner) because of the better alignment precision. This entailed restructuring and modifying parts of the base model to accommodate the change of aligner.

- The base model implementation of how the regions of the original audio to be edited and how short silences are handled had some important issues, which resulted, for example, in initial/final phonemes of the edited regions being cut off if certain (fairly common) conditions occurred. We instead implemented a modified version to make the handling of the regions more stable and precise.

- We improved the precision and the reliability of the computation of the duration of the mel spectrogram and of other markers for the edited transcript by introducing a more flexible hyperparameter mask_loc_buffer.

Fine-tuning to individual speakers: If the speech editor is used by a specific person, it makes sense to fine-tune some of the components of the model to that specific person. Since the base model FluentSpeech was trained on the LibriTTS corpus, which mainly includes American speakers, we decided to pick a British female speaker with a quite distinctive voice as our specific speaker (the Narrator from Baldur's Gate 3, voiced by amazing Amelia Tyler). We found a 4 hours video of background-noise-free Narrator's lines on YouTube. After downloading the audio and heavily preprocessing it to cut it into clean segments, for each chunk we then used Whisper to get a first tentative transcript and then manually cleaned the dataset, before finally putting it into a form that could be passed to our model. We then run two different types of fine-tuning: a naive one, and a more refined one. For the more refined type of fine-tuning, following the paradigm outlined in the recent seminal paper AdaSpeech (https://arxiv.org/abs/2103.00993) we only fine-tuned on conditional input certain specific layers of our model involving the speaker embedding and speaker style. We run fine-tuning on a local machine with a GTX1080 gpu for 17,750 (naive)/ 53,250 (AdaSpeech-style) steps. Compared to the base model losses, most of the losses went significantly down with the AdaSpeech-style fine-tuning.

Live streaming speech editing app: For the live streaming app, we used a TKinter GUI and threading to manage various parallel activities (input stream; transcription and inference; output to GUI; video handling). The user is able to then select the preferred modality of audio and video input (e.g. microphone/cam, YouTube urls, or files saved on the local machine) and edit the dictionary of words to be edited. In the GUI, the spectrogram, the transcript, the video frame (if available) of the current (possibly edited) segment of audio played is then shown for the user's convenience.

Key performance indicators

In line with our project objectives, using ablation techniques we evaluated our model based on the speed of the inference time relative to the base model, and on the various loss functions used by the base model. We also evaluated the generated edited speech based on the more subjective metric of "sounding better/closer to the original speaker".

Results

Our model performs inference much faster than the base model (usually in under a second). The near-live speed allowed us to create the live streaming speech editing app, which simulates live streaming and manages to keep up with the stream after inference with a maximum delay of usually 5-6 seconds.

The modifications that we have made in the way that phoneme durations and silences are handled in the generation of the mel spectrogram corresponding to the edited audio yielded much more robust and reliable results compared to the original model, where there was a common problem of entire phonemes being ignored when generating the edited audio, resulting in missing syllables/sound in the output edited audio.

The AdaSpeech-style fine-tuning to a specific individual speaker also yield significant decreases in all of the various loss functions used by the base model for training. Moreover, from a subjective perspective, the resulting generated voice sounds incredibly like the ground truth voice; however, we have encountered some worsening in phonemes enunciation (but not necessarily in alignment or prosody). We speculate that this is due to the profound difference between phonemes pronounced with an American accent and phonemes pronounced with a British accent. In the base model, both the encoder and decoder are trained on mainly American speakers, while due to lack of time and difficulty of accessing the original vocoder's weights we could only fine-tuned the encoder to our chosen British speaker.

Future work

- Properly clean the code (there are lots of vestigial remains from the FluentSpeech code that we don't actually use).
- Fine-tune the phonemes decoder (instead of just the encoder) as well, in order to improve the quality of the generated edited speech when tailoring our model to an individual speaker, in order to solve some phoneme enunciation problems encountered.
- Integrate better the various models used (remove redundancies).
- Improve the stability and expand the range of functionalities of the live-streaming speech editing app.

- Include a loss which attempts to improve the smoothness of the mel spectrogram boundary (like fluenteditor paper)
- Use whisperx as ground truth phoneme alignment during training. This would implicitly also allow for different language if we also use it as our phoneme tokenizer.
- Consistency models to speed up spectrogram denoiser
- Adaspeech style conditional layer norms for speaker ids