Overview

As of April 2024 we've experienced a remarkable progress in ML for speech synthesis. Modern TTS models take a lot of inspiration from NLP field in particular by employing transformers as in VALL-E[1], using masked language modelling for learning self-supervised representations as in HuBERT[2] or WavLM[3] (importantly WavLM jointly learns masked speech prediction and denoising, that makes this model robust to various acoustic conditions). Another important line of work uses learnable residual vector-quantization for building of neural codecs that efficiently compress speech, music and audio signal in general. SoundStream[4] is one of such models that beats state of the art codecs on very low bitrates, 3kbps.

Training any TTS model requires:

- Segmented and aligned **dataset**, pairs of texts and corresponding audio clips. Covering multiple speakers, speaking styles and acoustic conditions. $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N}$ Modern TTS models are trained on hundreds even thousands of hours. For research purposes it's sufficient to train model on tens of hours, to get reasonable model performance. One of the classic datasets for English TTS is LJSpeech[5] , this is a single speaker dataset with total length of approximately 24 hours.
- **Model architectures** that leverage inductive biases and observations in the domain of speech, audio, seq-to-seq modelling. Speech contains a lot of redundancy that makes compression possible. Ideas that allow to bridge the gap between highly compressed text representation and audio waveform will be helpful in building robust and resource efficient TTS models. As showcased in the most recent models FACodec[6] and speech trident survey[7]
- **Compute**. TTS models take significant time to converge, modern models are trained on multi-node multi-gpu clusters and take weeks or even months to converge. But again as with datasets it's sufficient to have access even to a single-node multi-gpu instance with modern GPUs like RTX 4090 or RTX3090 to effectively experiment in the field of TTS. I'm training my models on 6 RTX3090 GPU rig, and this configuration is sufficient for majority of Speech processing model I'm working on. You are lucky if you can train models on DGX instances with V100s, take most out of them!

Data Preparation

TTS model the process of reading text out loud, consequently to avoid ambiguities input texts have to be properly preprocessed. Preprocessing pipeline are language dependent and on a high level view consist of:

- *text normalization* includes short form expansion, conversion of numbers to words. All this can be achieved in deterministic way with help of WFST(weighted finite state transducer) models. They are implemented for multiple languages in NeMo text processing toolkit[8] You can find a well detailed introduction to WFSTs in Jurafsky's book Speech and Language Processing[9]

  Example: ```

  ```
  St.Patrick's day is a good reason to treat yourself with that $30 bottle of wine!  -> Saint Patricks day is a good reason to treat yourself with that thirty dollar bottle of wine!
  ```

- *phonemization* common next step in text preprocessing is to convert written text to phonemes, all because phonemes are more closely mapped to underlying acoustics. Today's universal approach is to do IPA phonemization with help of espeak-ng. Phonemizer[10] is good python library for this task, it supports multiple backends and allows to run phonemization for over 100+ of languages. This step is particularly important for languages with complicated spelling as English, but might be of less necessity for Spanish that is more strait forward in spelling. Overall this makes task of TTS easier, and makes possible convergence with less amount of data and compute.
  Example:

```
Saint Patricks day is a good reason to treat yourself with that thirty dollar bottle of
wine! -> s'eɪnt p'atɹɪks d'eɪ ɪz ɐ g'ʊd ɹ'iːzən tə tɹ'iːt jɔːs'ɛlf wɪð ðat θ'ɜːti d'ɒlə b
'ɒtəl ɒv w'aɪn
```

Alignments

Another important aspect of data preparation is verification of the correspondence between written and spoken text. Often times TTS datasets are automatically segmented from web crawled long-form recordings such as audio books and youtube videos. It's common to see datapoints with some of the missing or redundant words in text. Excessive regions without speech at the beginning or end of the utterance also might significantly deteriorate performance of the TTS model.

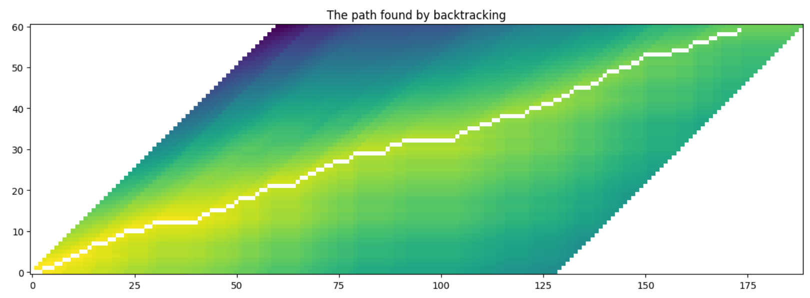
Question

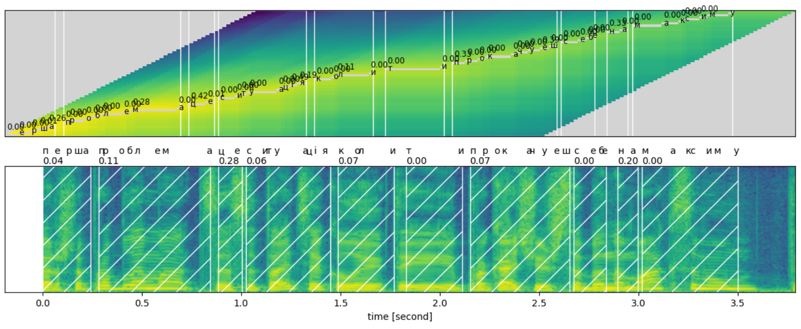
Can you think of other scenarios where alignments might be useful and for what applications?

It's a common practice to inspect the dataset and filter out problematic datapoints. Team at NVIDIA proposed a toolbox for construction and analysis of ASR datasets, that also can be used for TTS purposes[11]. That approach allows to align unsegmented long-form recordings with the corresponding text, with consequential chunking and analysis of segmented data. CTC-based pretrained model is used to extract alignments as described in CTC-segmentation paper[12], poorly aligned regions are discarded. Final filtering is done by measuring WER and removing clips with low correspondence between text and audio.

0:00 / 0:03

```
word                    score    start    end
перша                   0.04         0     240
проблема                0.11       280     840
це                      0.28       880    1000
ситуація                0.06      1020    1440
коли                    0.07      1480    1760
ти                      0.00      1820    2100
прокачуеш               0.07      2140    2640
себе                    0.00      2660    2820
на                      0.20      2880    2980
максиму                 0.00      3000    3480
```





Tacotron2

Back in 2017 tacotron2[13]was a SOTA model, achieving human level quality with `MOS=4.5`. Now tacotron is a classic example of sequence-to-sequence modelling in pre-transformers era.
Bellow I'll expand a bit more on what tacotron is, how the model is trained, and where are its merits and limitations.
Tacotron is a text to spectrogam model that consists of 3 major building blocks: characters(phoneme) encoder, attention block; auto-regressive decoder.

Encoder is represented as a stack of convolutional layers that take phone embeddings and output contextualized phones, that effect is attributed to convolutional nature of the encoder additional bi-directional LSTM allows to model long-term contexts and influences across all the characters of the sequence.

Attention block is the most important and hard to learn part of the model. Authors use location-sensitive attention that is described in attention based models for ASR[14] Intuitively attention is represented as an *alignment matrix* between text and spectrogram frames. Where length of text is N characters and length of the spectrogram is M frames.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}, A \in \mathbb{R}^{N \times M}$$

Attention selects relevant characters that are mapped to the corresponding spectrogram frame. We do this by accounting previous alignment a_{i-1} too, so that model attends to element of the sequence that in close proximity to previous position. This encourages model to monotonically move forwards.

ⓘ **Question**

Can you think of alternatives on how text tokens can be mapped to their corresponding acoustic representations?

Decoder We use teacher forcing (ground truth previous frame is used for conditioning) and attention context from the encoder to predict next spectrogram frame. During inference teach-forcing is not possible so previously predicted spectrogram frames are used for the conditioning. Autoregressive nature of decoding is slow and grows with the length of the spectrogram.

Loss function. The model is trained using frame-wise MSE loss. Remember teacher forcing mentioned above? We need it exactly for the purpose of being able to compute MSE, since otherwise model will be less constrained in terms of modelling durations for each of the phonemes in the utterance. Our goal is to match number of frames in ground truth and predicted spectrograms as well as to have all the frames exactly aligned.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N} \|x^{(i)} - \hat{x}^{(i)}\|^2, x^{(i)} \in \mathbb{R}^F$$

ⓘ **Question**

What metrics can you think of for measuring quality of a TTS system?

Ecosystem

These days when deep learning for speech is getting mature, multiple libraries are available for training TTS models, the most prominent of them are:

- NVIDIA's NeMo[15]
- ESPnet[16] from CMU
- s3prl[17] - speech self supervised learning toolkit

These toolkits are good for training production quality models, though for research purposes I suggest to focus on standalone easily hackable repos with implementations of your interest so that you can iterate quicker while testing new ideas or learning new concept. I suggest exploring NeMo tutorials[18] on a jupyter notebooks[19] covering various TTS models including classic examples and modern approaches.

---

1. VALL-E↩
2. HuBERT↩
3. WavLM↩
4. SoundStream↩
5. LJSpeech↩
6. FACodec↩
7. SpeechTrident↩
8. NeMo text processing↩
9. Speech and Language Processing↩
10. phonemizer↩
11. construction and analysis of speech datasets↩
12. CTC-Segmentation of Large Corpora for German End-to-end Speech Recognition↩
13. Tacotron2 ↩
14. Attention-Based Models for Speech Recognition↩
15. NeMo↩
16. ESPnet↩
17. s3prl↩
18. NeMo TTS tutorials↩
19. NeMo TTS jupyter notebooks↩