Lecture 14: Text-to-Speech Synthesis

Zhizheng Wu

Agenda

- Recap
- Applications
- Overview of text to speech
- Frontend
- Acoustic model
- Waveform generator
- Tools and readings

Text normalization

- Normalizing text into standard format

- Every NLP task requires text normalization
  - Tokenizing (segmenting) words
  - Normalizing word formats
  - Segmenting sentences

Grapheme to phoneme

- Grapheme: a letter or a group of letters that represent a single phoneme
- Phoneme: the smallest unit of sound that can distinguish one word from another in a particular language

- when a child says the sound /t/ this is a phoneme, but when they write the letter 't' this is a grapheme.

Grapheme t o m a t o

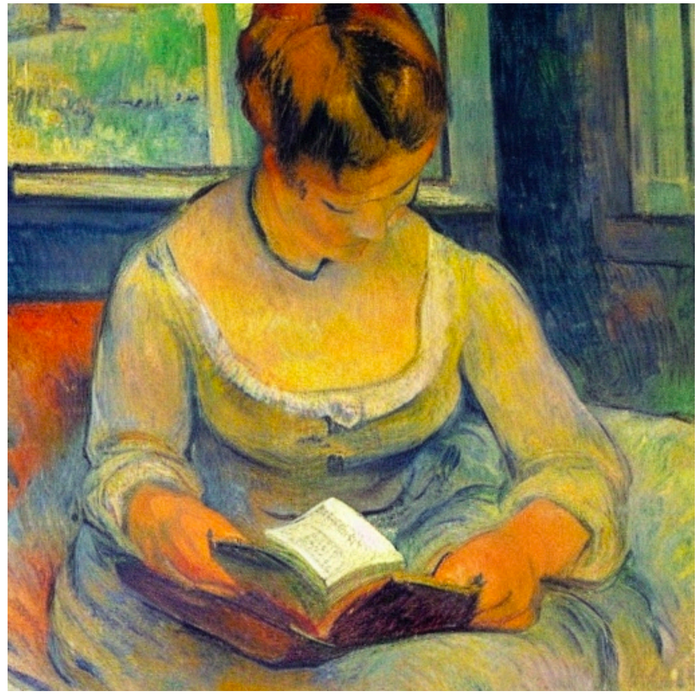
Phoneme /t ə' m eɪ. t oʊ/

Part-of-speech tagging is a disambiguation process

Verb or Noun?

Verb or Noun?

She is **reading** a book about **Reading**



Embedding representations

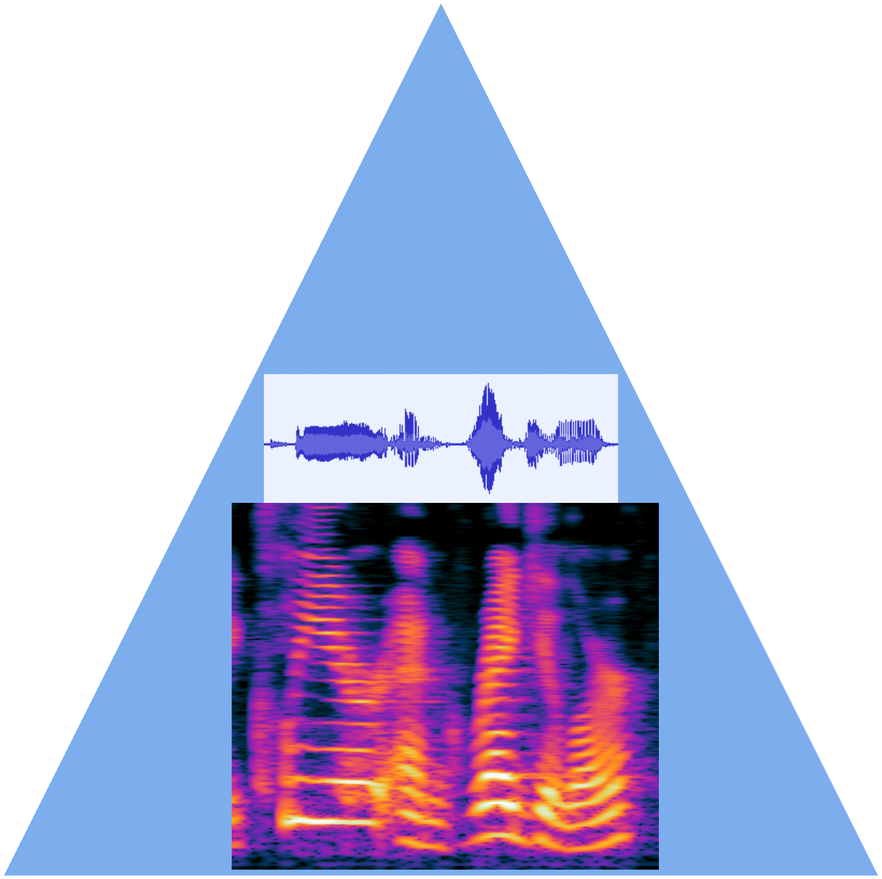
Dense Matrix

1	2	31	2	9	7	34	22	11	5
11	92	4	3	2	2	3	3	2	1
3	9	13	8	21	17	4	2	1	4
8	32	1	2	34	18	7	78	10	7
9	22	3	9	8	71	12	22	17	3
13	21	21	9	2	47	1	81	21	9
21	12	53	12	91	24	81	8	91	2
61	8	33	82	19	87	16	3	1	55
54	4	78	24	18	11	4	2	99	5
13	22	32	42	9	15	9	22	1	21

Sparse Matrix

1	.	3	.	9	.	3	.	.	.
11	.	4	.	.	.	.	.	2	1
.	.	1	.	.	.	4	.	1	.
8	.	.	.	3	1	.	.	.	.
.	.	.	9	.	.	1	.	17	.
13	21	.	9	2	47	1	81	21	9
.	.	.	.	.	.	.	.	.	.
.	.	.	.	19	8	16	.	.	55
54	4	.	.	.	11	.	.	.	.
.	.	2	.	.	.	.	22	.	21

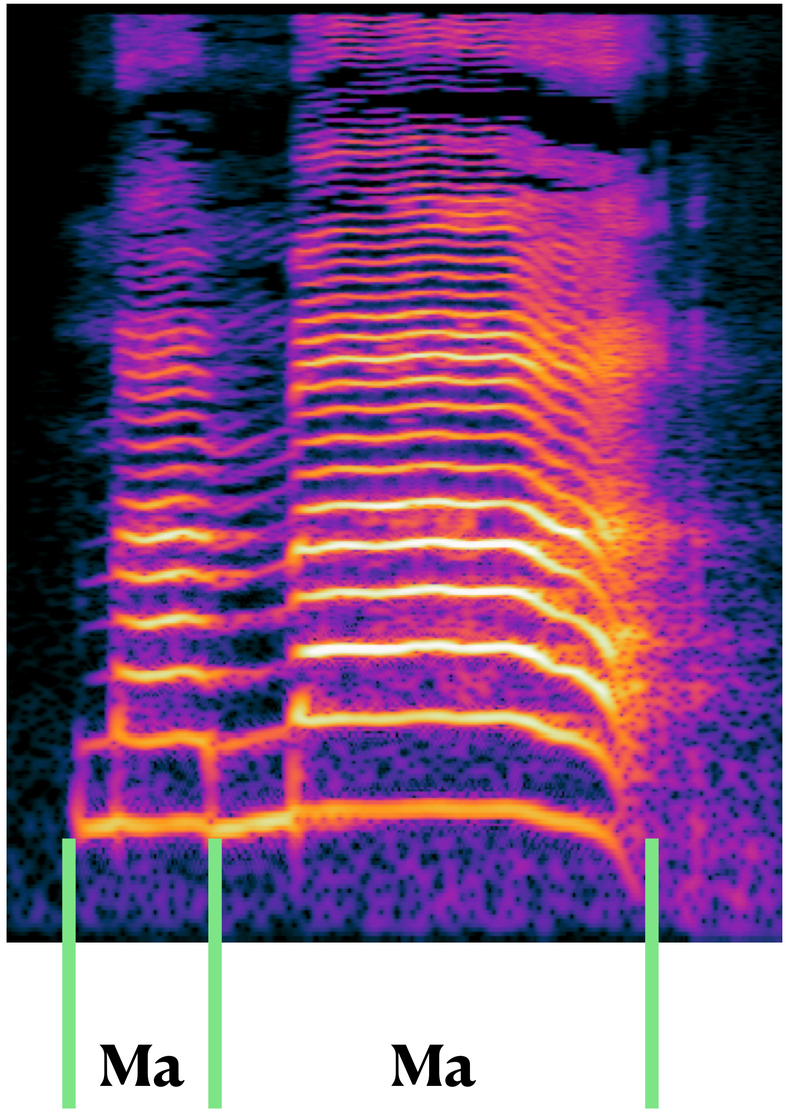
Content

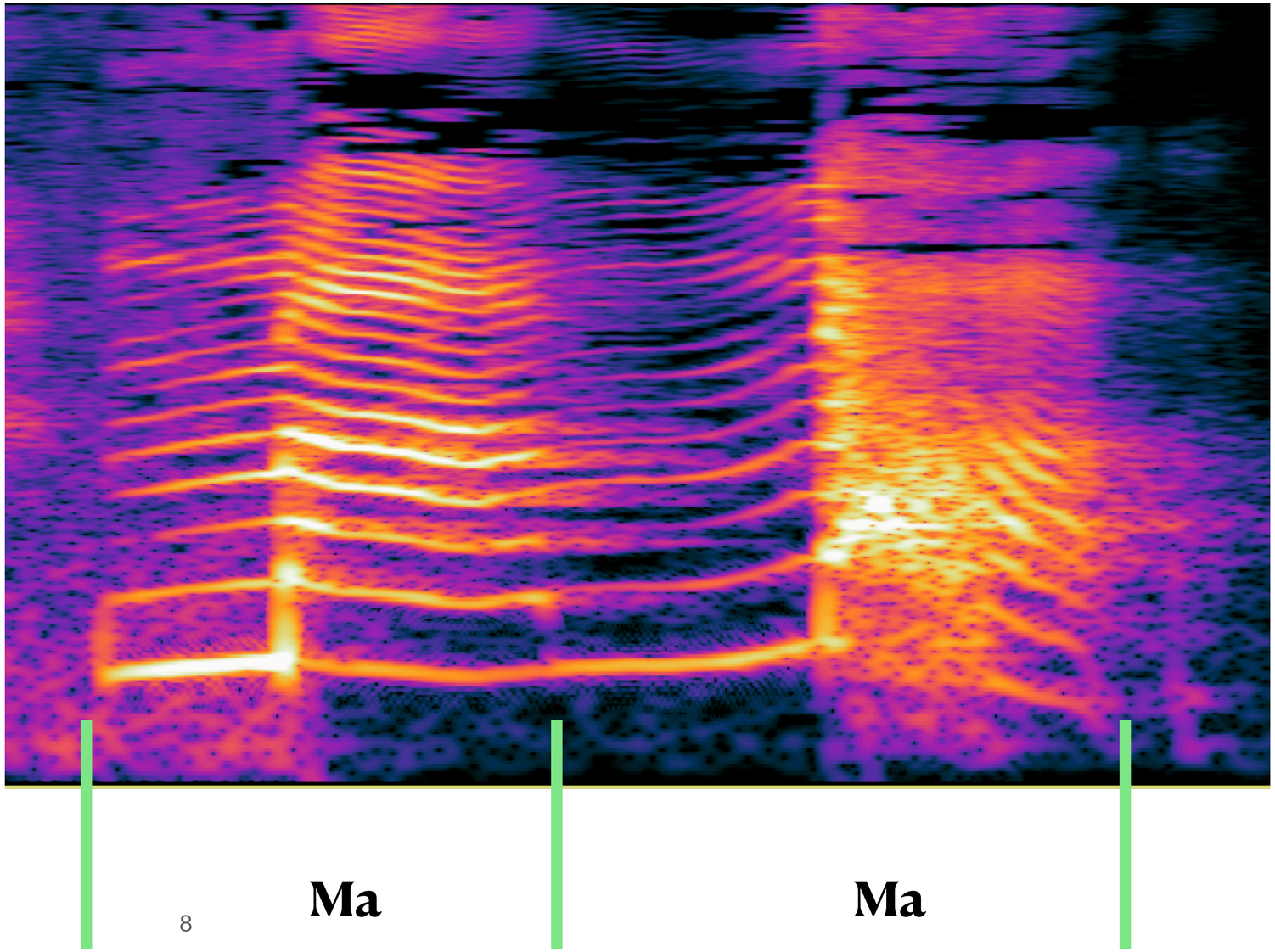


Timbre

Prosody

Speech representation

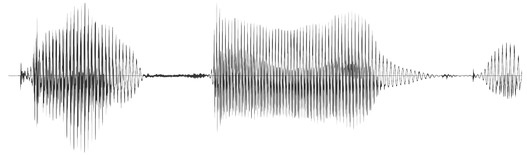




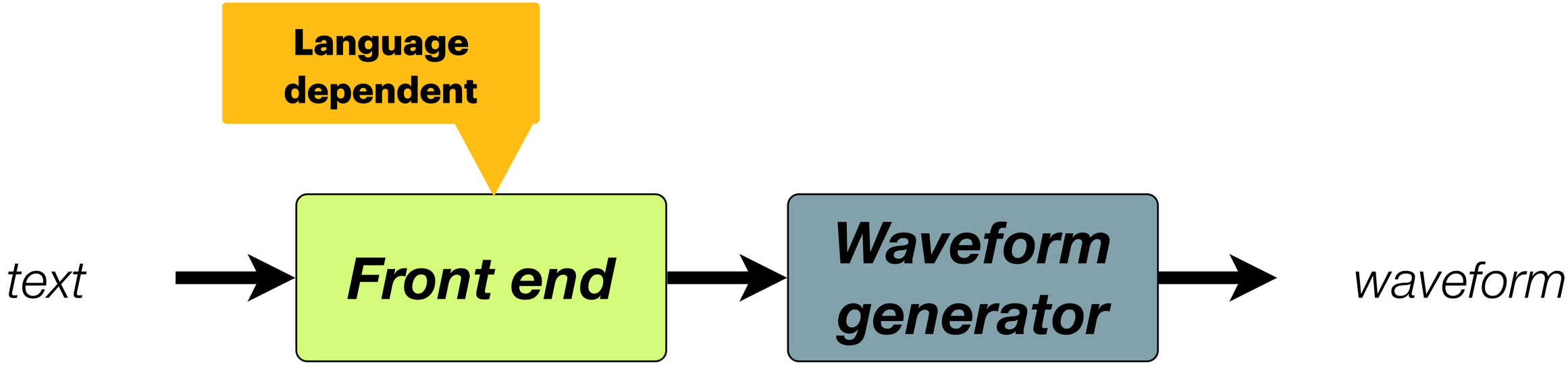
The end-to-end problem we want to solve

text → ***Text-to-Speech*** → *waveform*

`Author of the…`

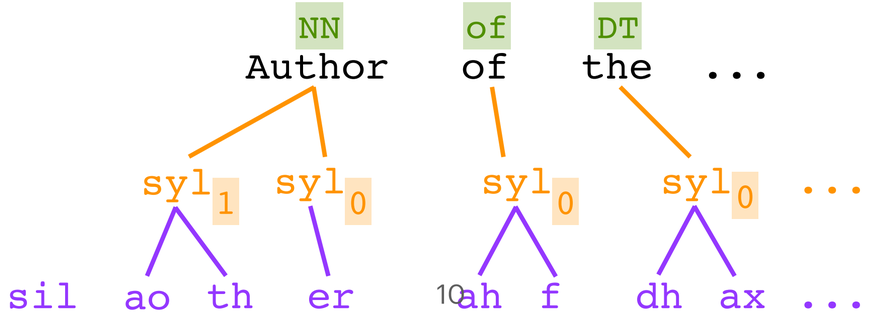


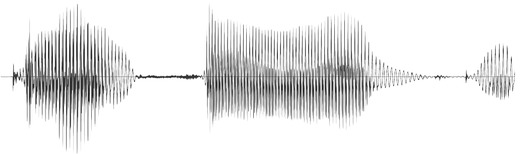
The two-stage pipeline



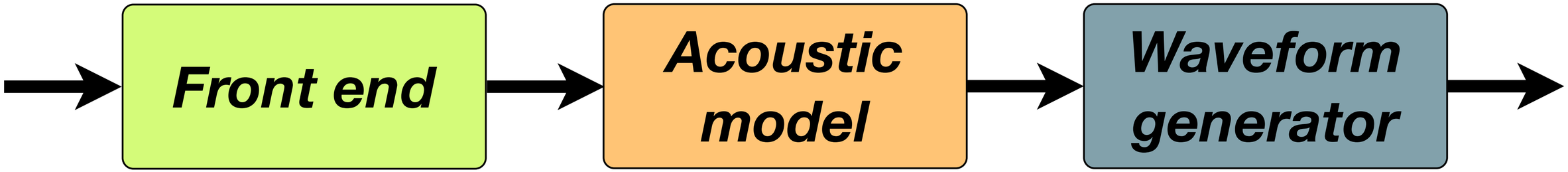
`Author of the…`

linguistic specification





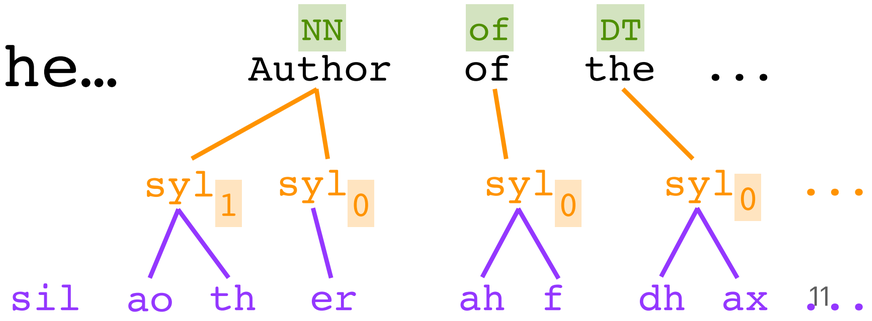
The three-stage pipeline



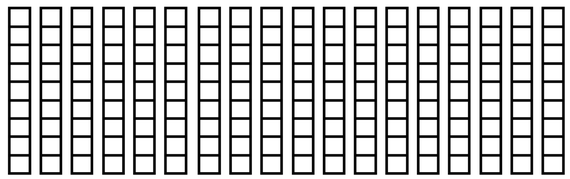
text

Author of the…

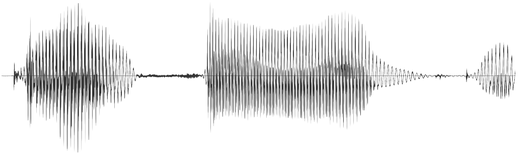
linguistic specification



acoustic features

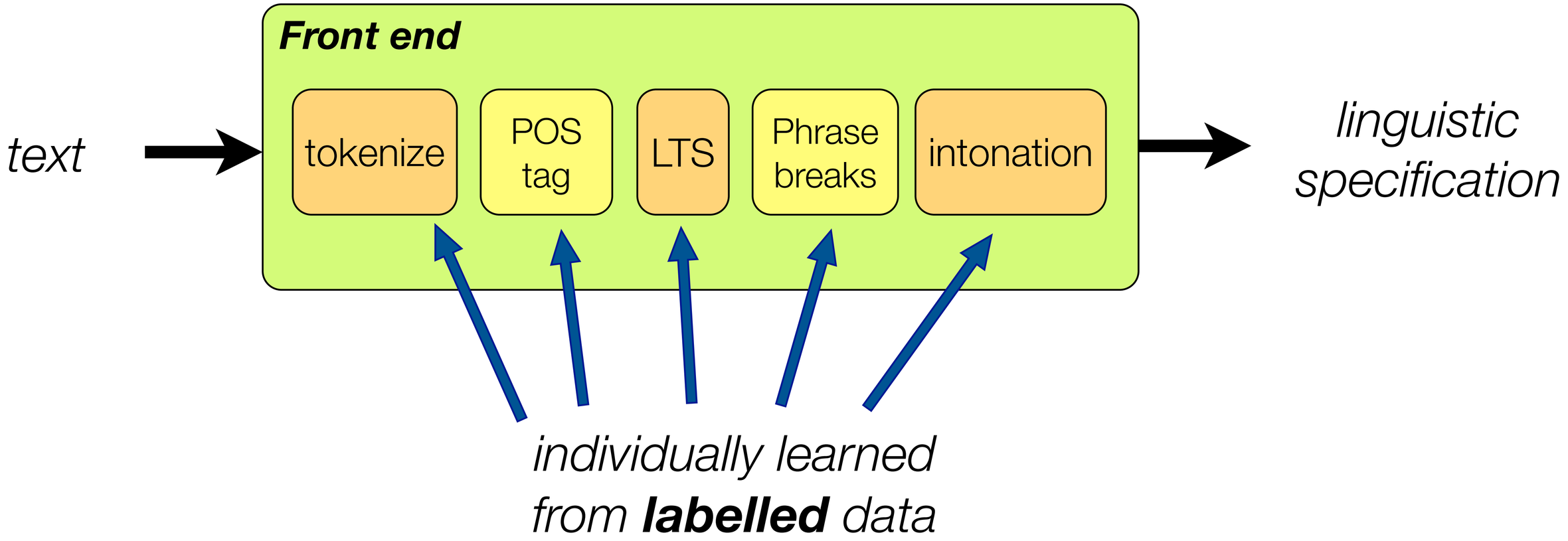


waveform



Front end

Front end


Front end
text
tokenize
POS tag
LTS
Phrase breaks
intonation
linguistic specification
individually learned from **labelled** data

Front end

- Language dependent: Each language has its unique characteristics

Hello world

世界你好

Front end

- Handle text normalization
  - $123 -> one hundred and twenty three dollars

Front end

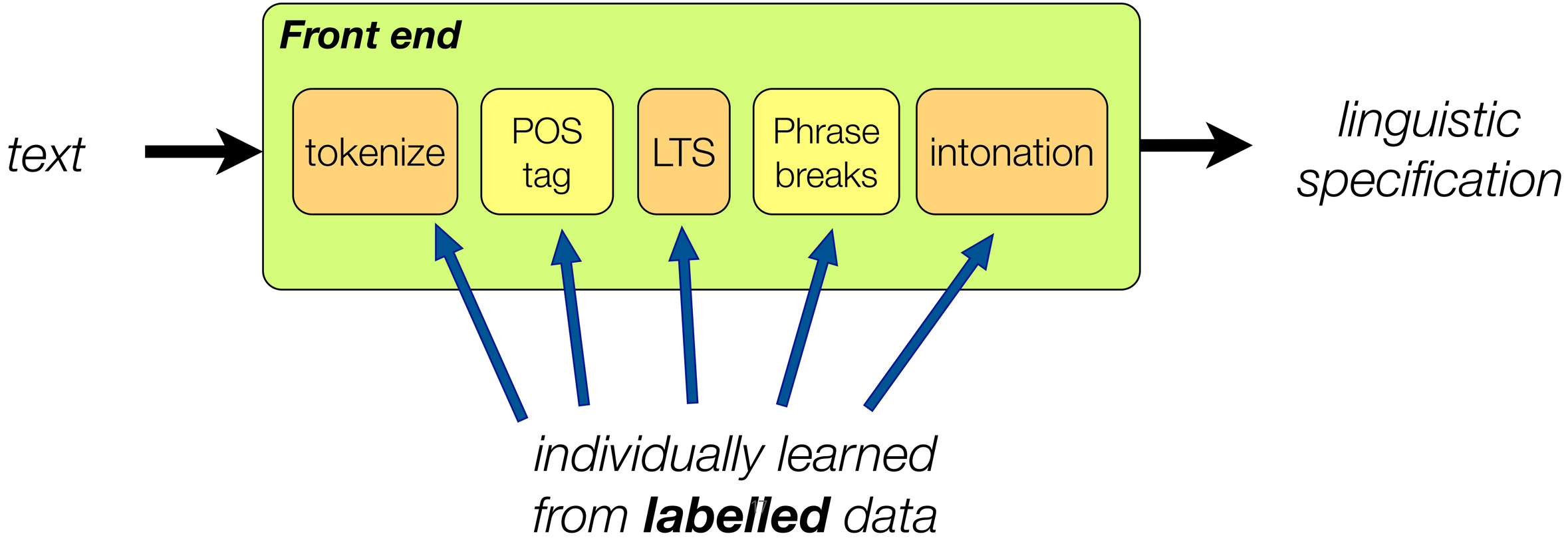
- Handle pronunciation of words in different context

  - Read
  - record

  - 奇偶 vs 奇怪

Classic front end

- A chain of **processes**
- Each process is performed by a **model**
- These models are independently trained in a **supervised** fashion on annotated data

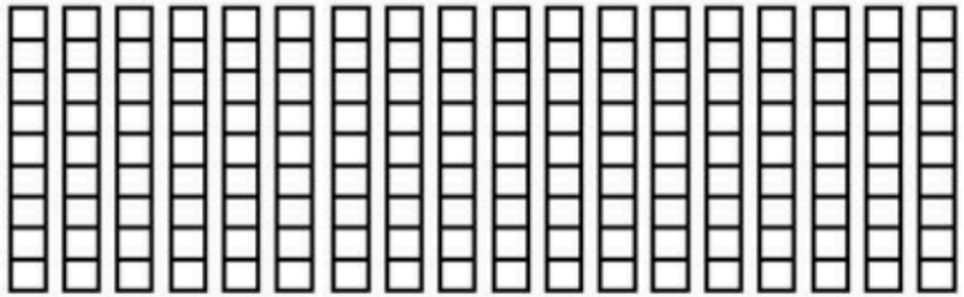


Neural front end

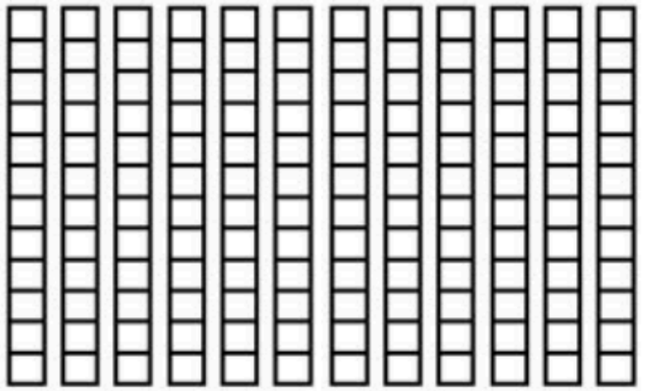
- Learn by a neural network

Linguistic features vs acoustic features

output sequence
(acoustic features)



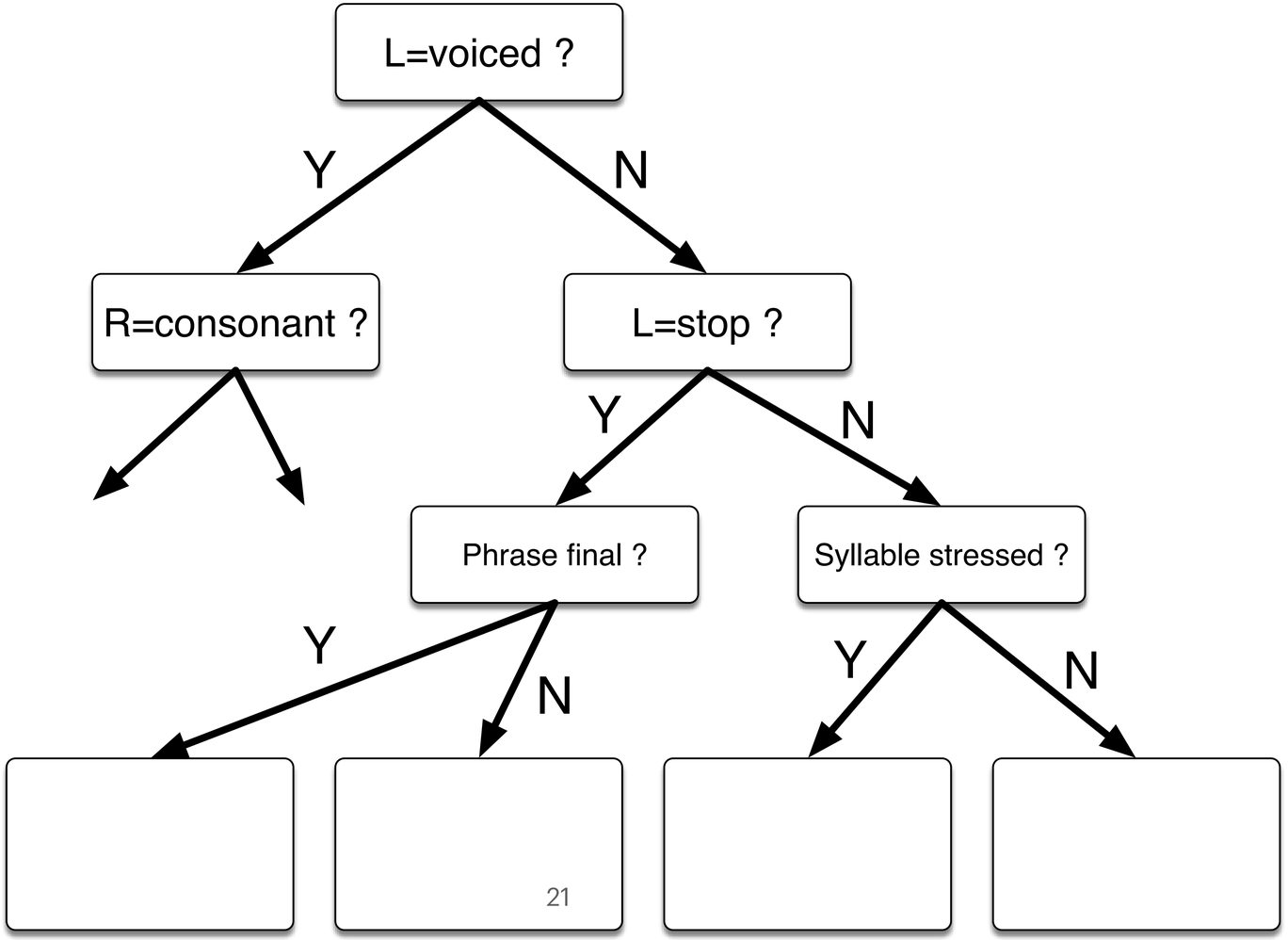
input sequence
(linguistic features)



Acoustic model

Acoustic model - Decision tree

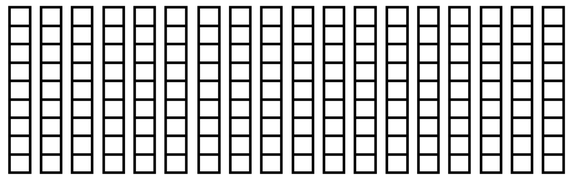
- Decision tree to group HMM states, which model acoustic feature distribution

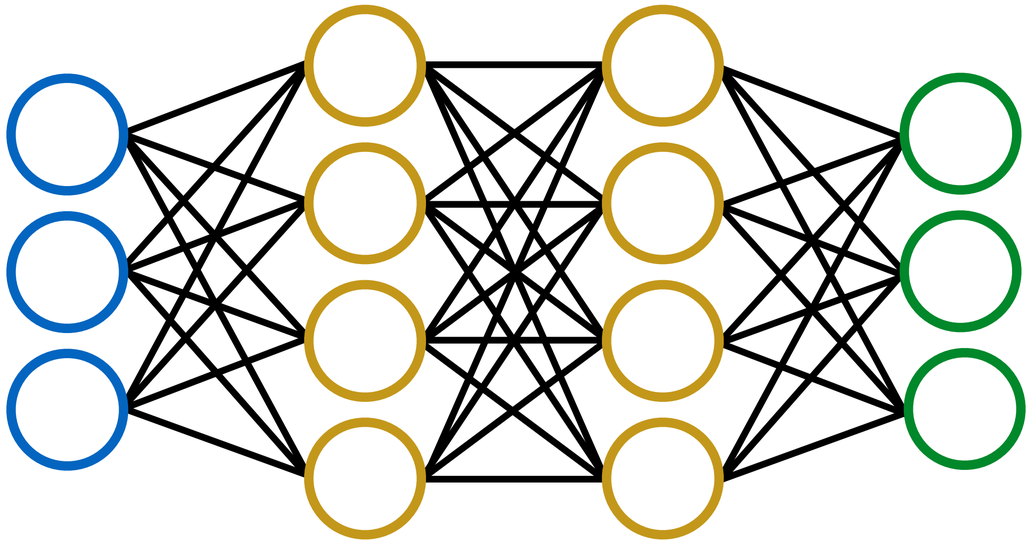


Acoustic model: DNN

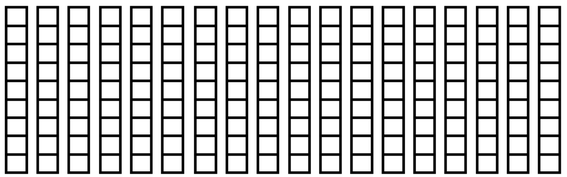
- Feedforward neural network

input
Linguistic features





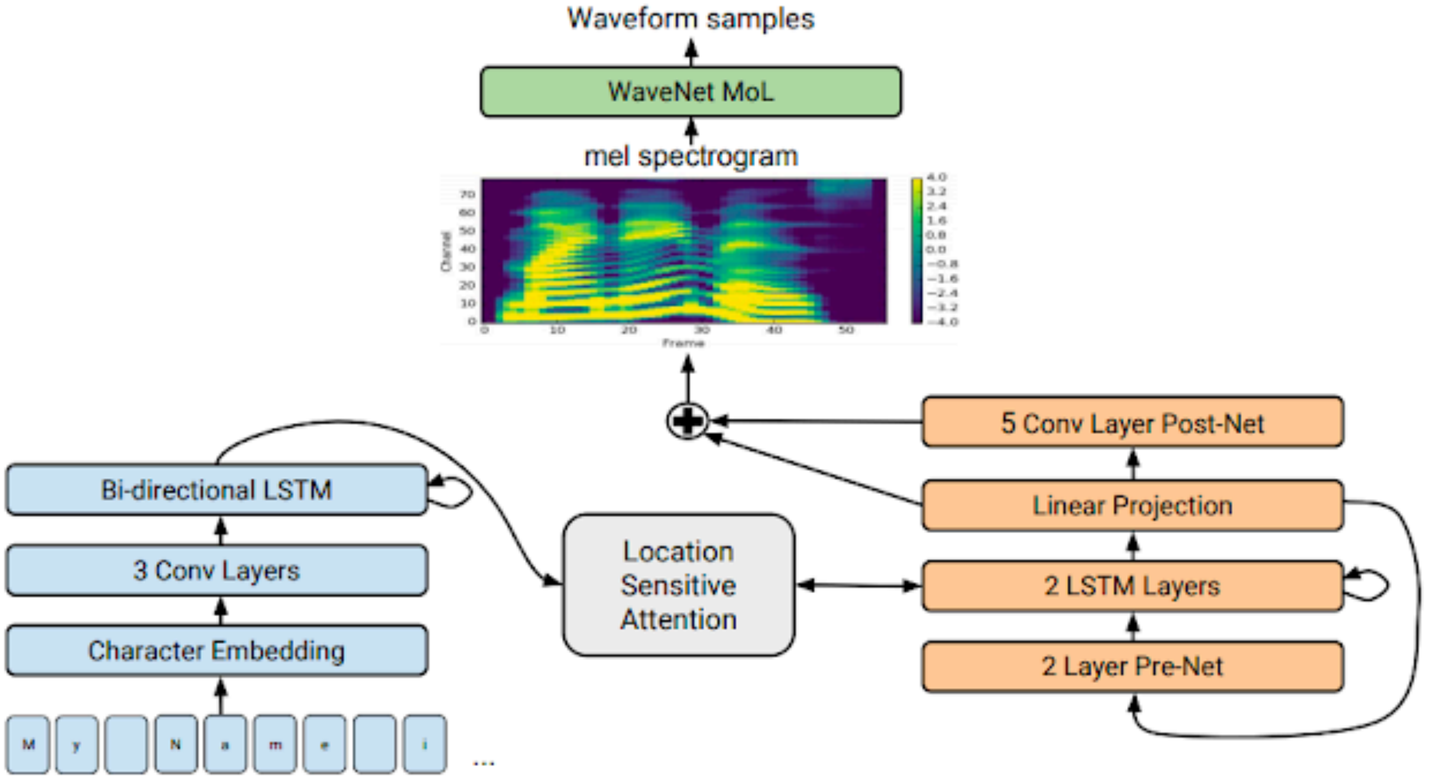
output
acoustic features



Watts, Oliver, Gustav Eje Henter, Thomas Merritt, Zhizheng Wu, and Simon King. "From HMMs to DNNs: where do the improvements come from?." In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5505-5509. IEEE, 2016.

Acoustic model - RNN based

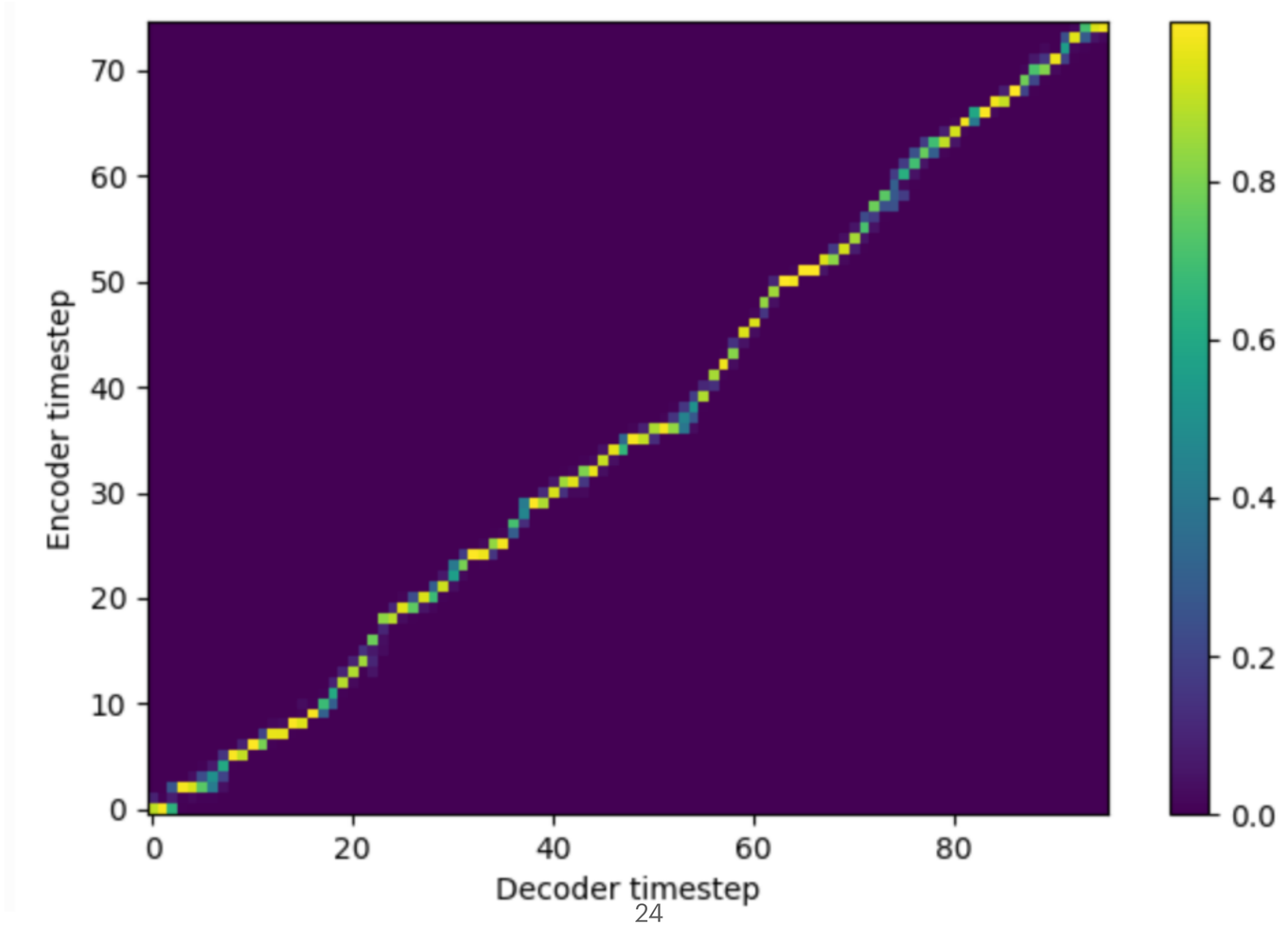
- Tacotron2: A sequence-to-sequence model based on Recurrent Neural Networks



Shen, Jonathan, et al. "Natural tts synthesis by conditioning wavenet on mel spectrogram predictions." *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018.

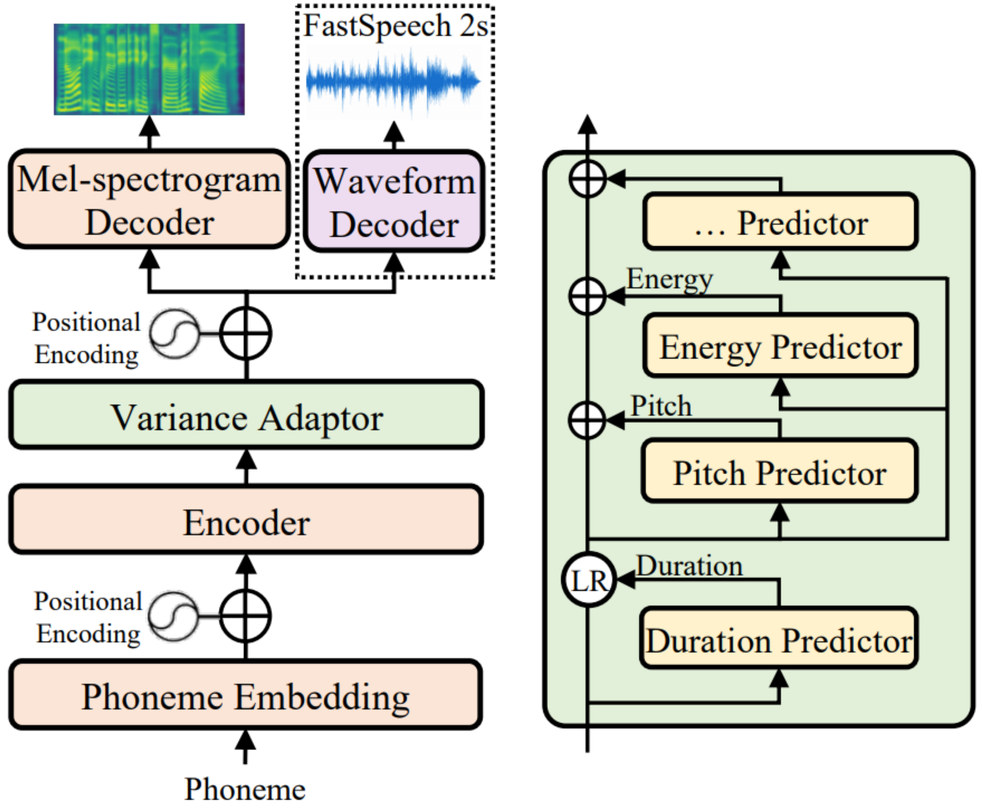
Acoustic model - RNN based

- Attention



Acoustic model - Transformer based

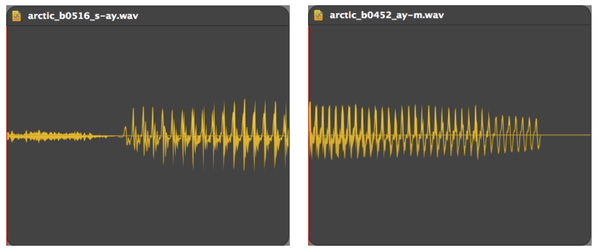
- FastSpeech2: parallel generation and not depending on the location attention



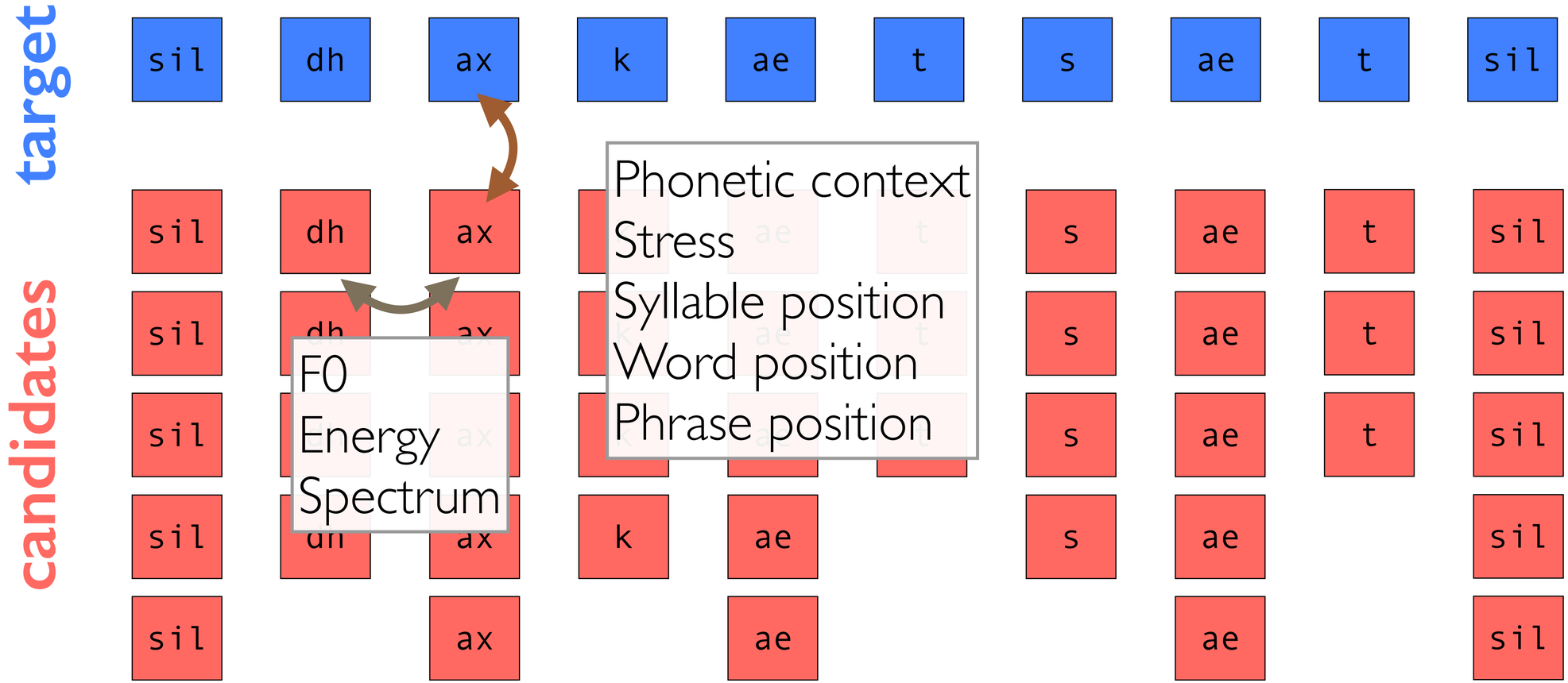
Ren, Yi, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. "Fastspeech 2: Fast and high-quality end-to-end text to speech." *arXiv preprint arXiv:2006.04558* (2020).

Waveform generator

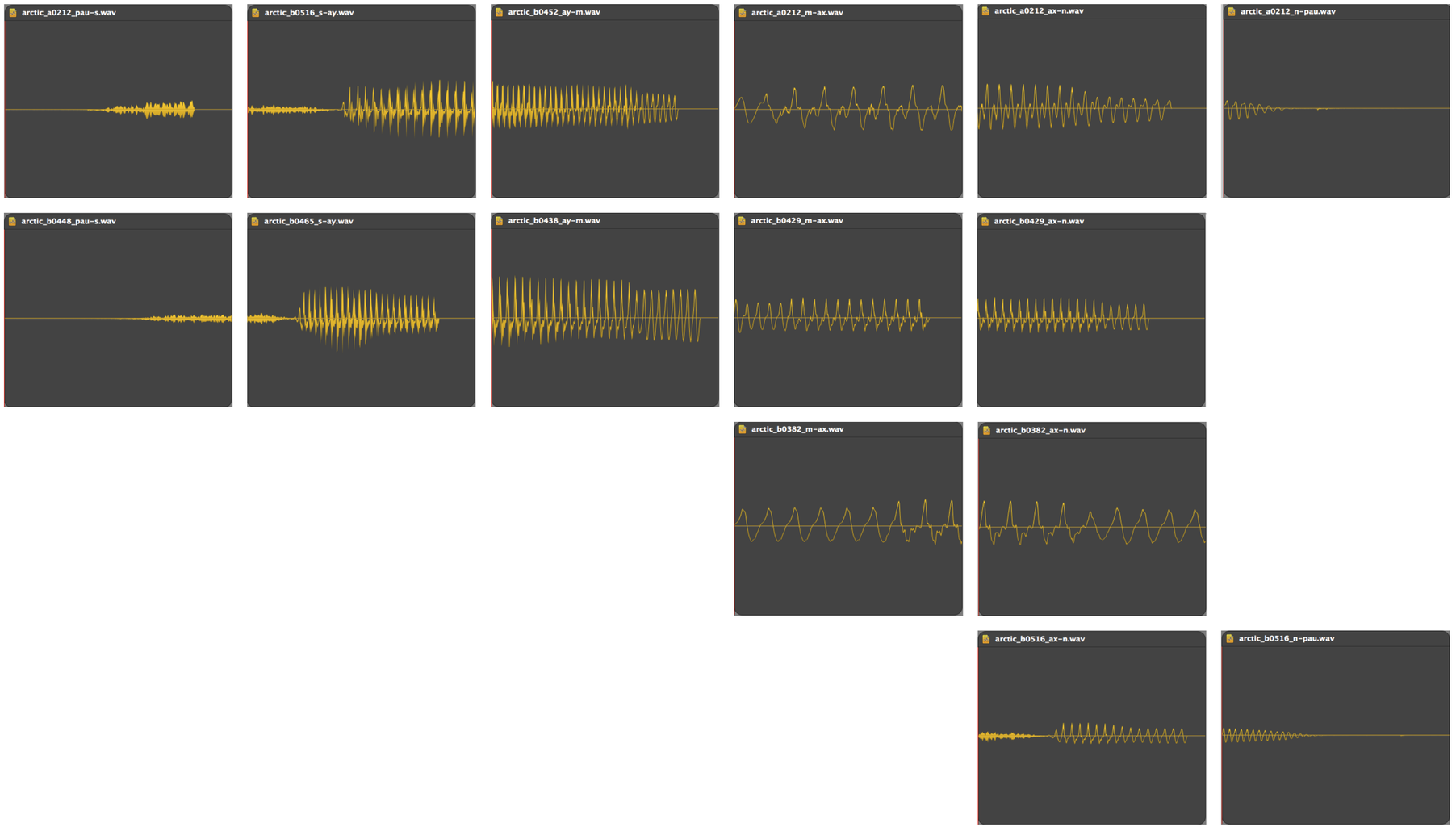
Waveform generator: Waveform concatenation



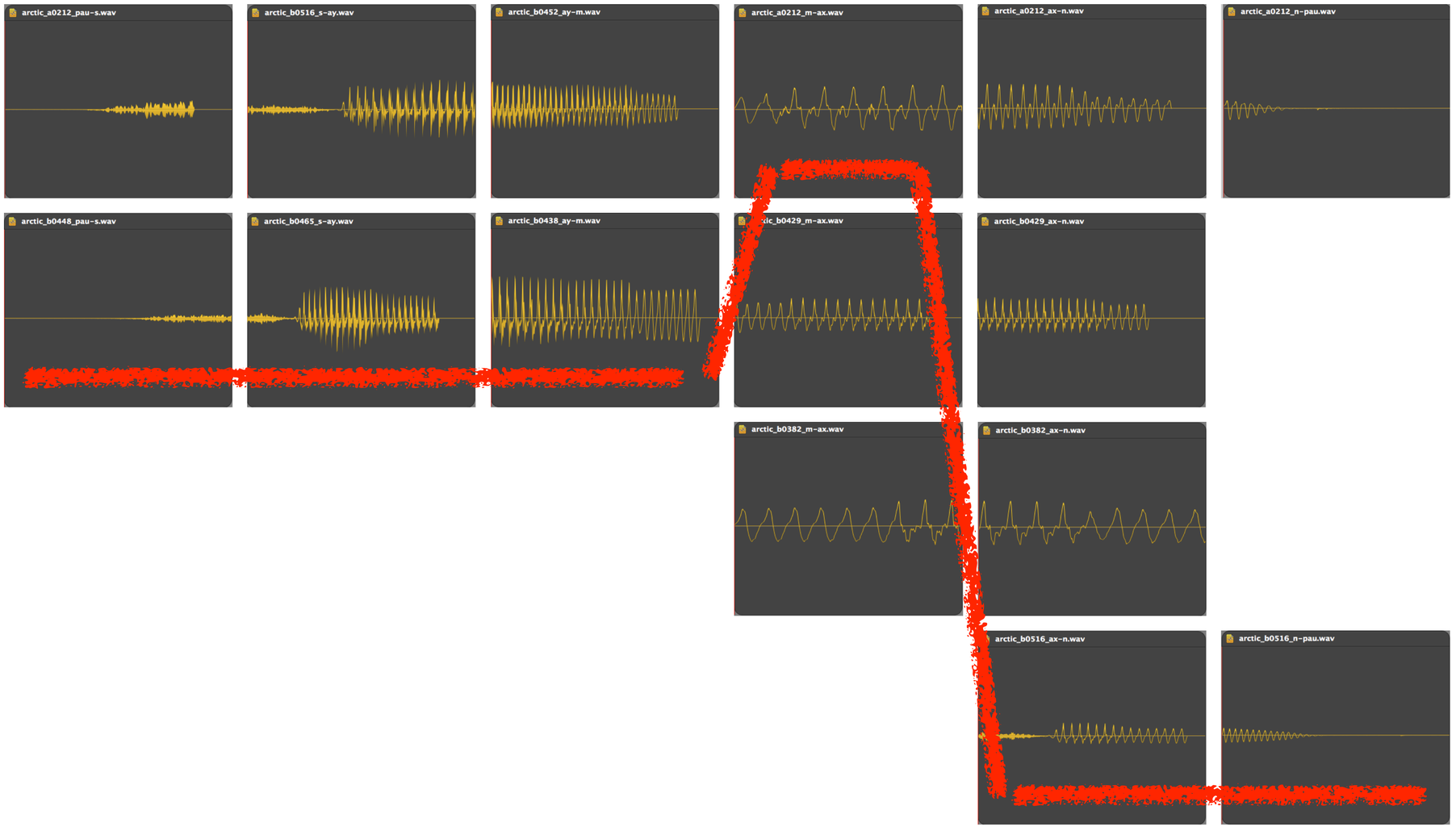
Classical unit selection (drawn here with phone units) - target and join costs



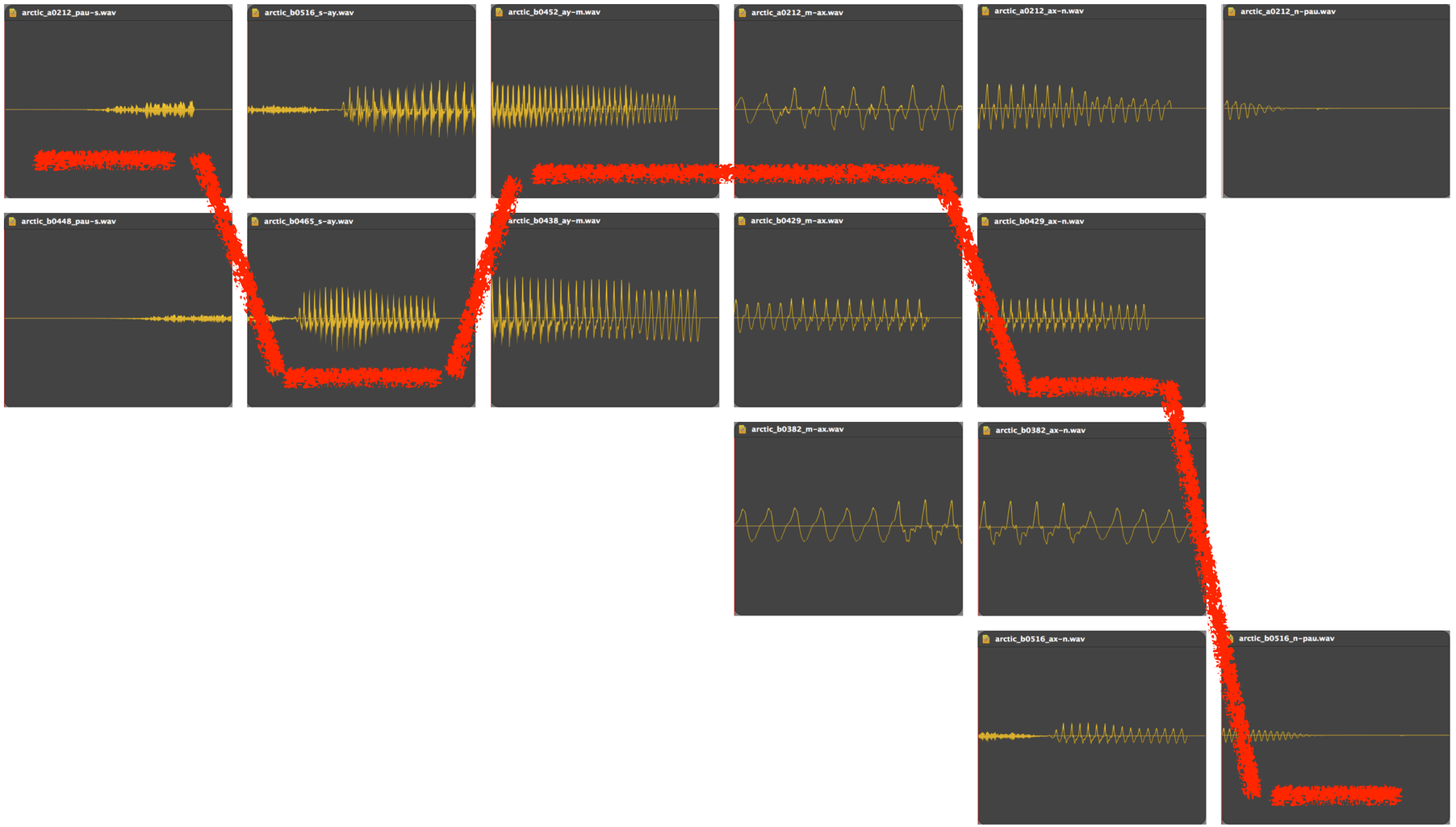
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
arctic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

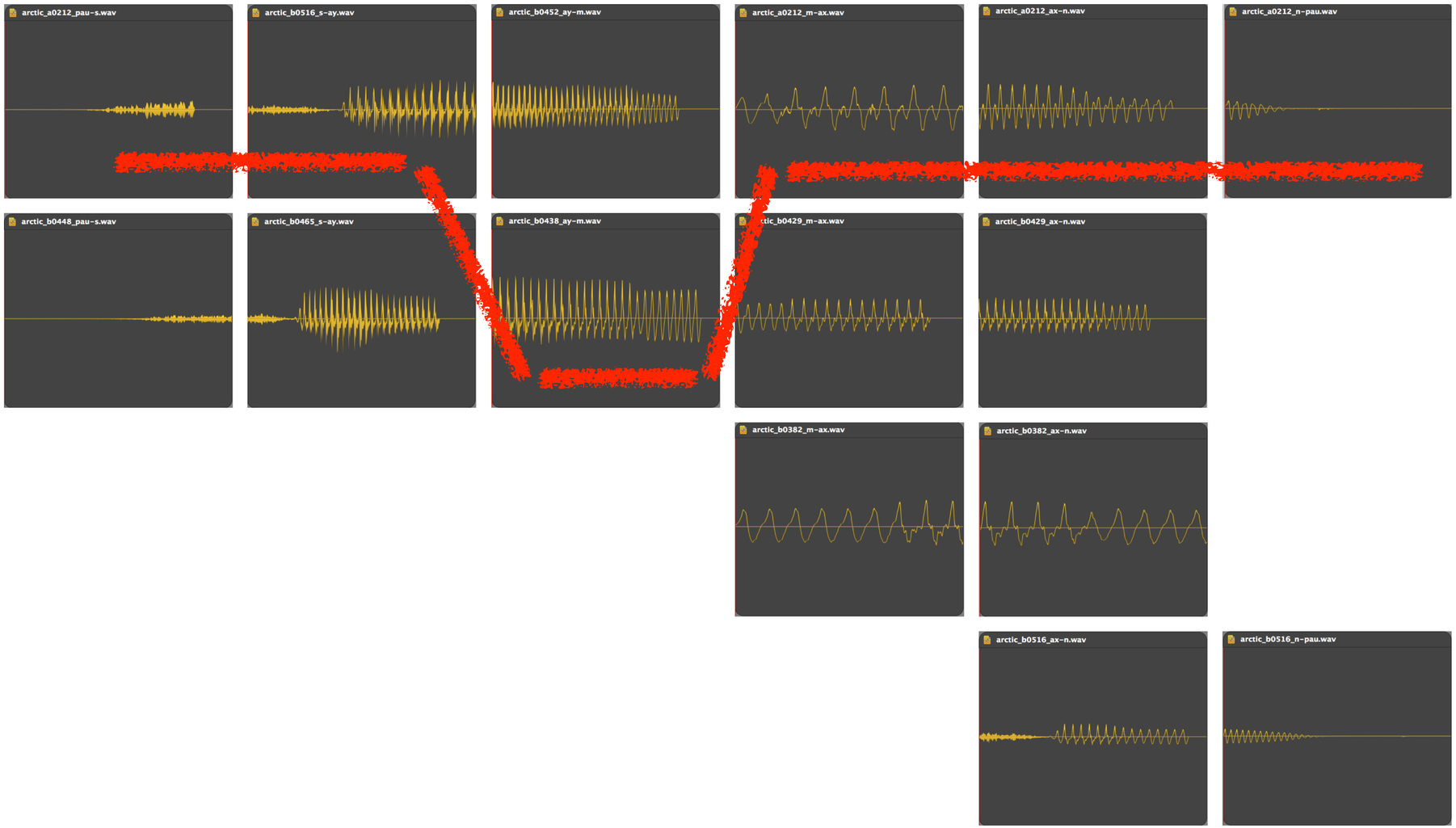
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
arctic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

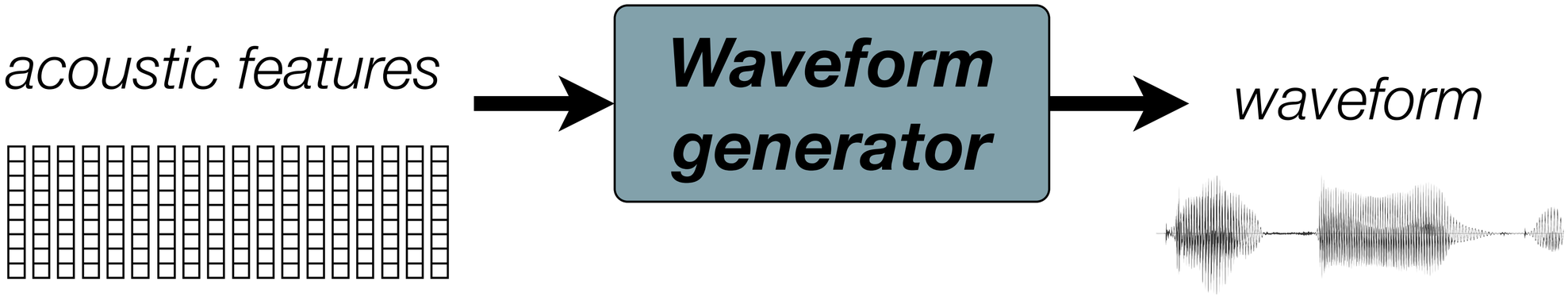
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
arctic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

“Simon”

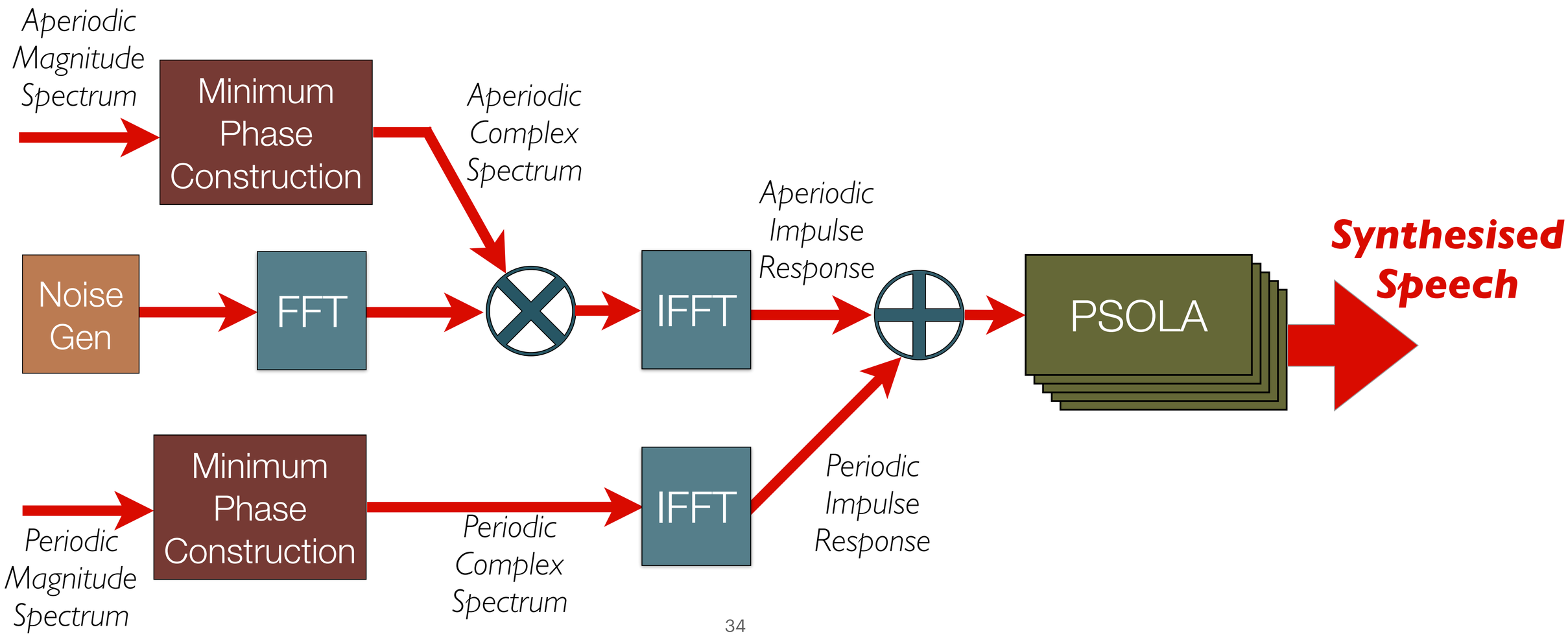

arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
arctic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

Waveform generator: Vocoder



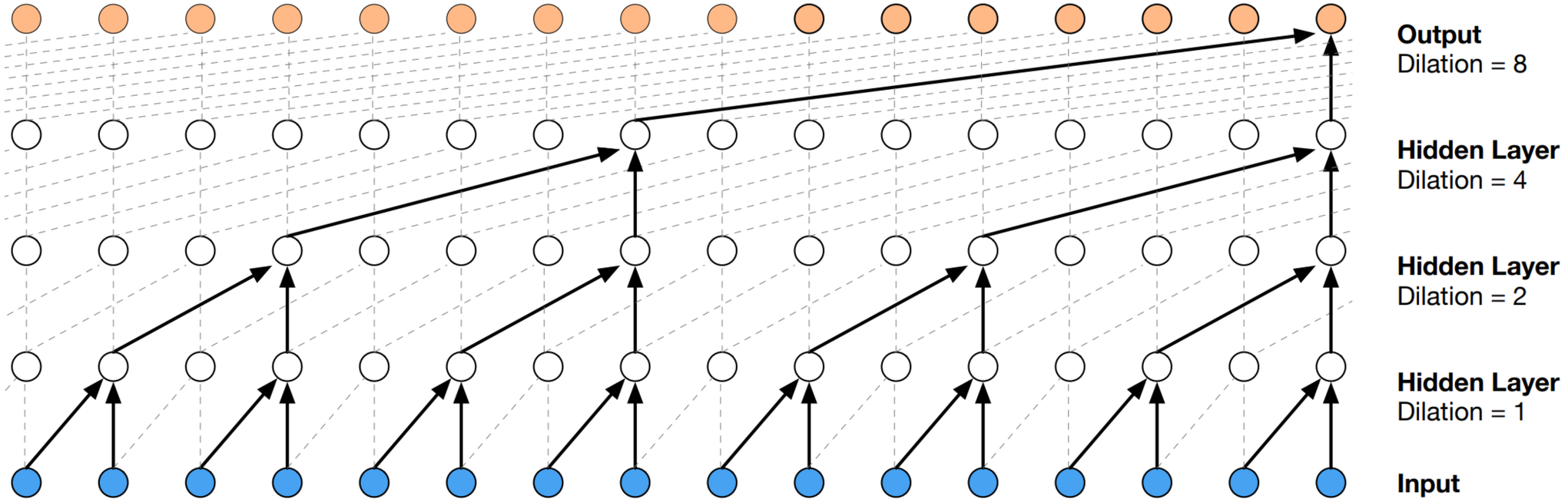
Vocoder - Signal processing based

- WORLD vocoder



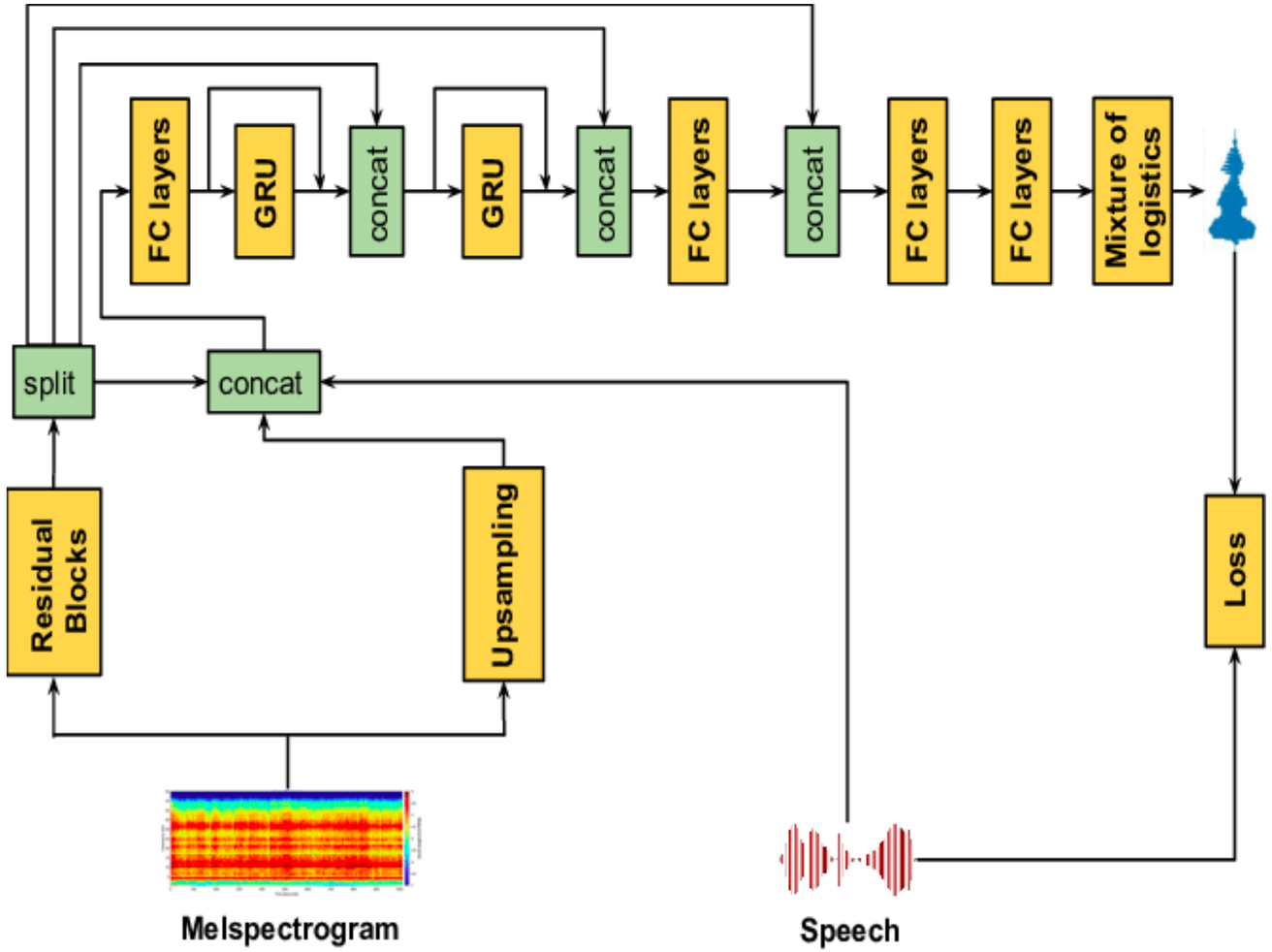
Vocoder: Autoregressive

- WaveNet: autoregressive model with dilated causal convolution



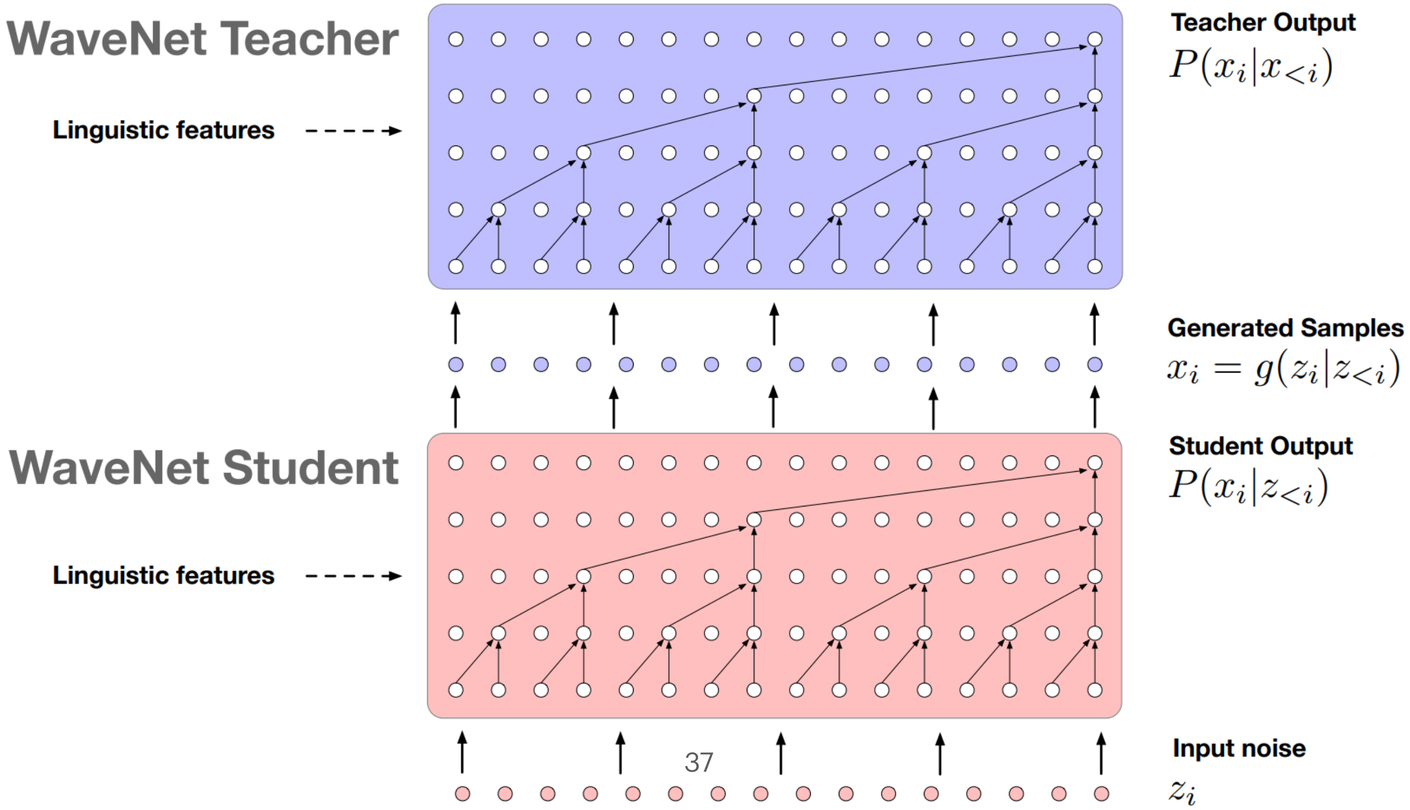
Vocoder: Autoregressive

- WaveRNN: autoregressive model with RNN



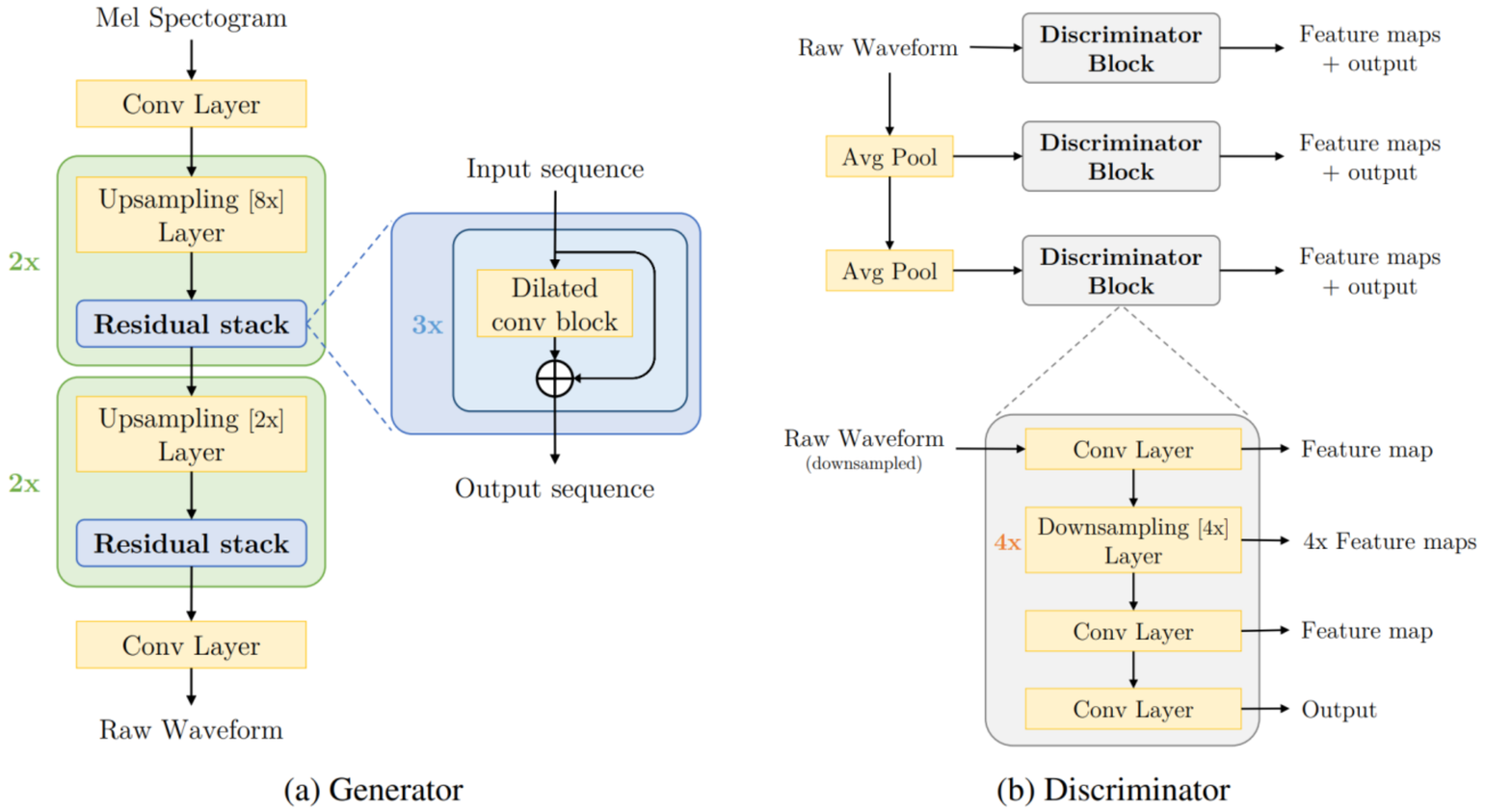
Vocoder: Flow based

- AF (autoregressive flow) and IAF (inverse autoregressive flow)
  - Parallel inference of IAF student
  - Parallel training of AF teacher



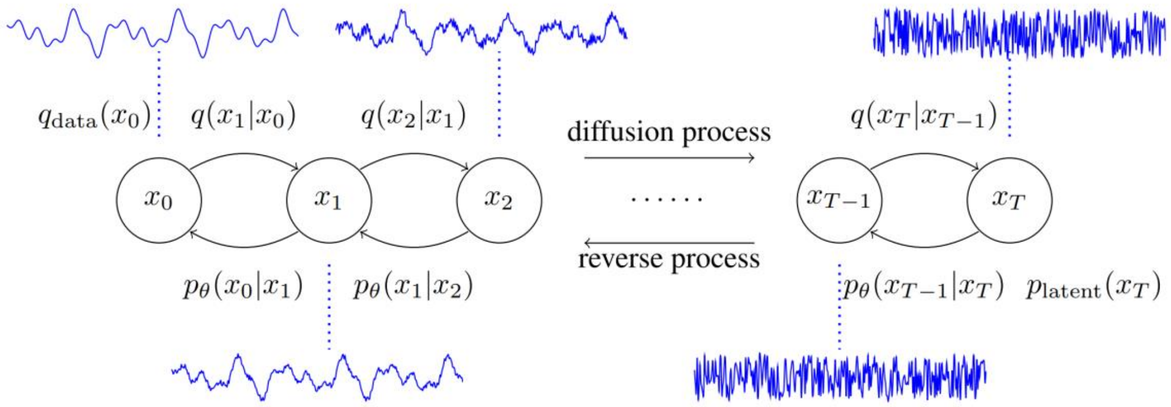
Vocoder: GAN based

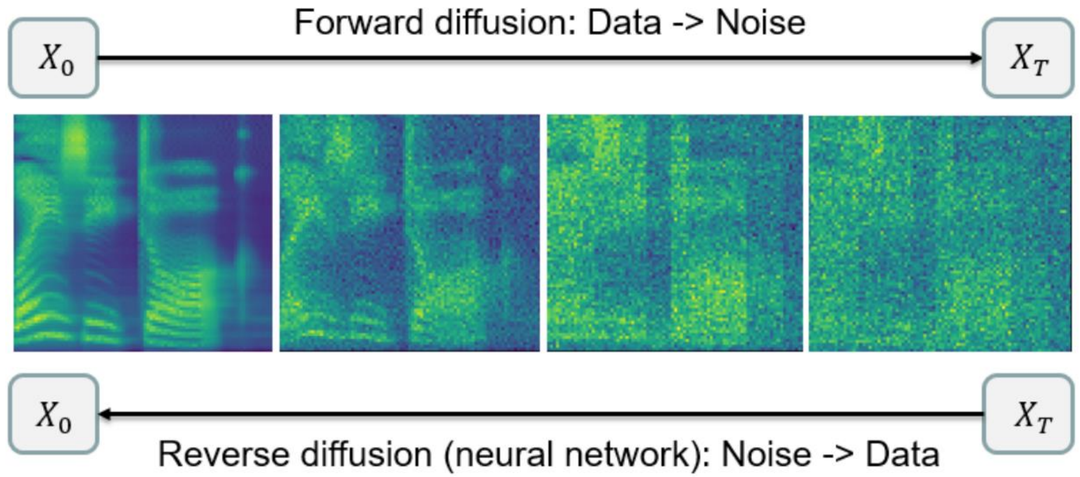
- MelGAN: Generator + Discriminator



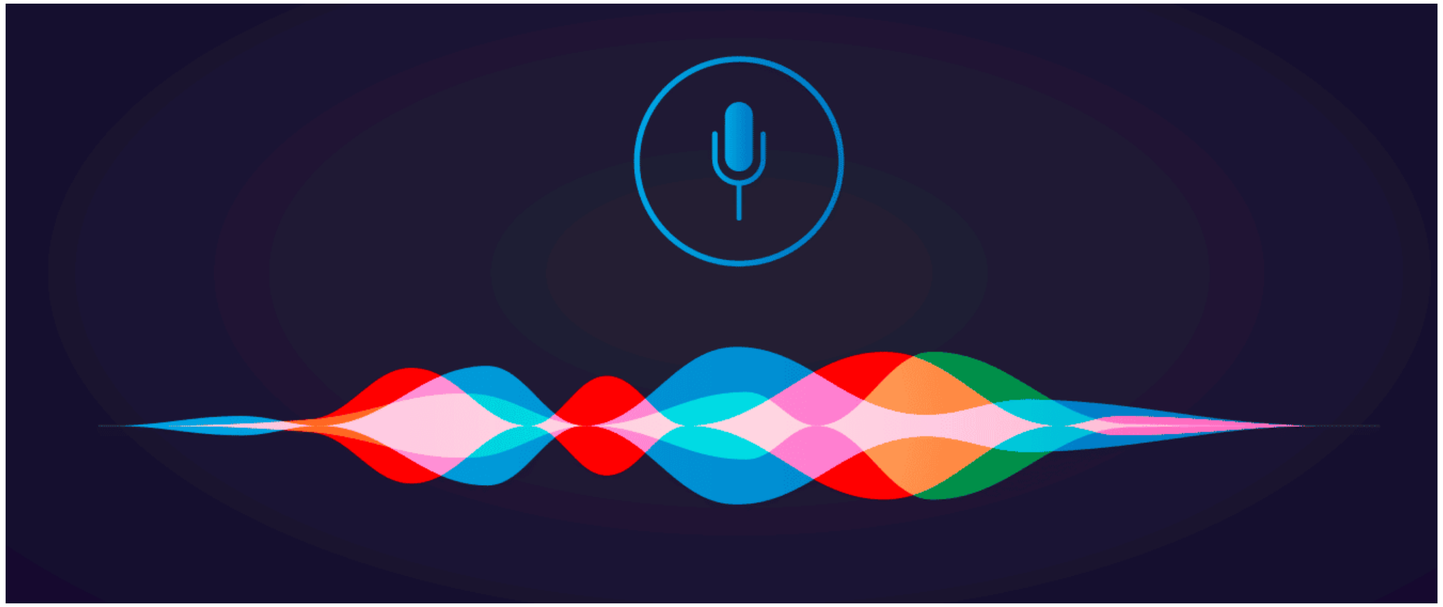
Vocoder: Diffusion based

- Diffusion probabilistic model
  - Forward process: diffusion
  - Reverse process: denoising

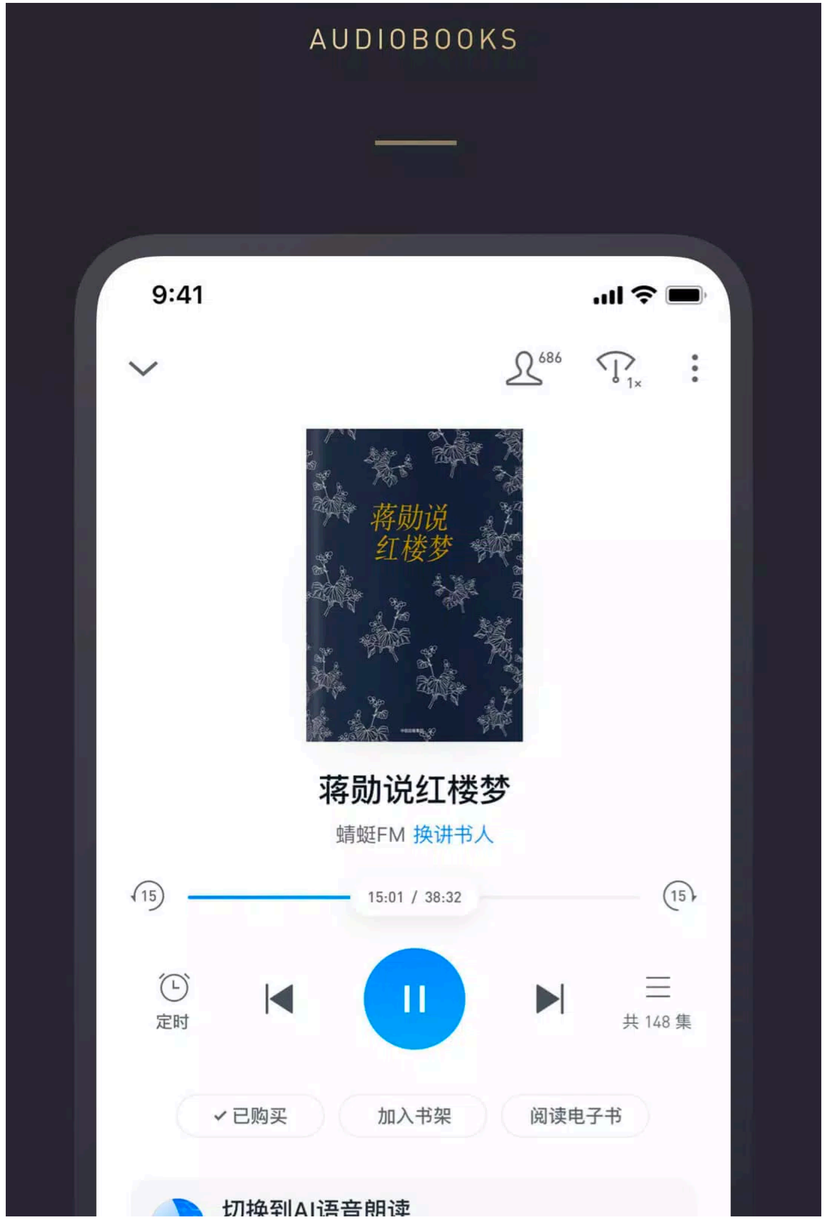




Applications



Applications

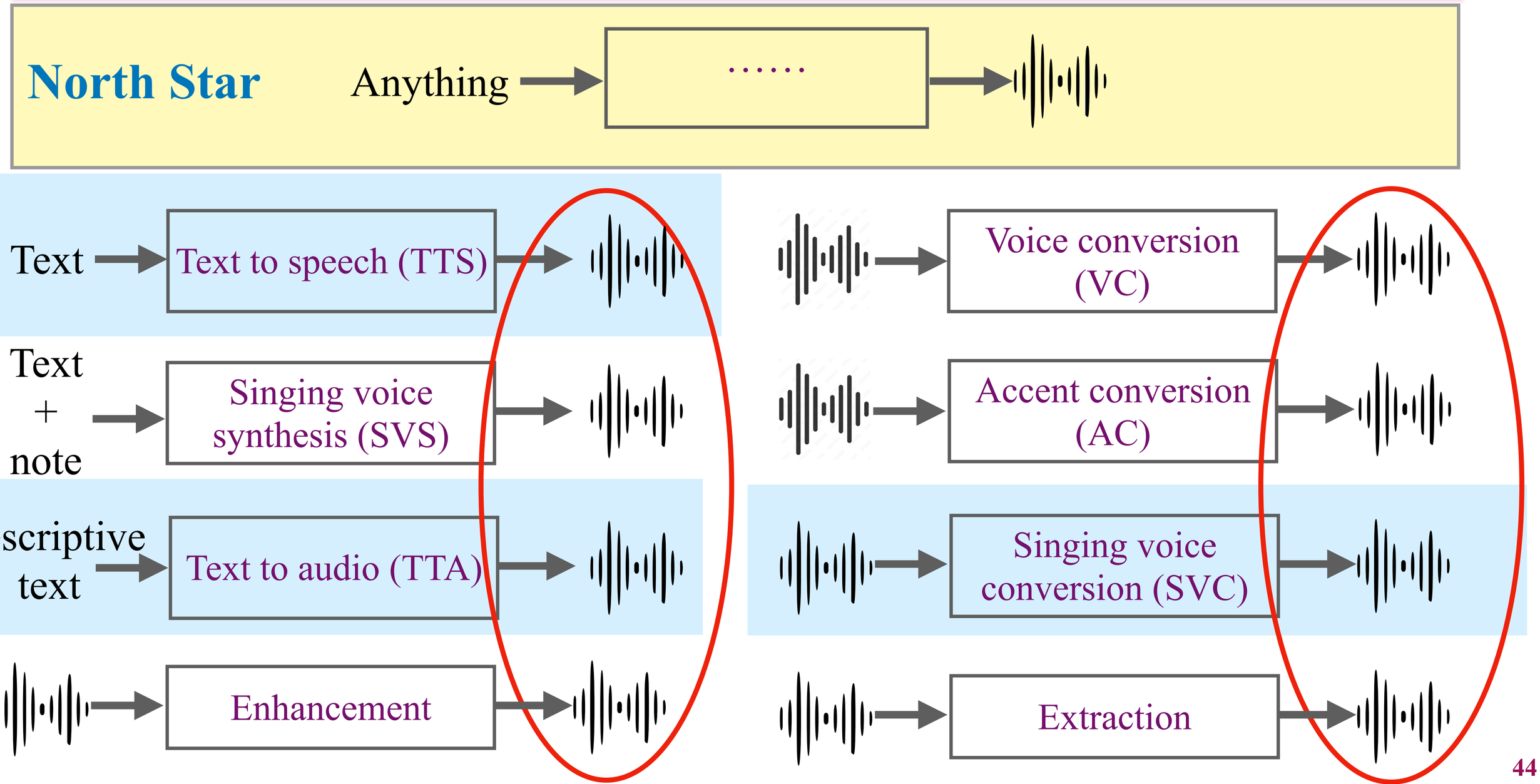


Applications

Tools

- TTS open-source
  - https://github.com/open-mmlab/Amphion
  - https://github.com/coqui-ai/TTS
  - https://github.com/espnet/espnet

Amphion: Generating audio, music and speech



Vocoder

- Objective comparison with existing toolkits. Eval on dev-test set of LibriTTS

System	PESQ(↑)	F0RMSE(↓)	Sampling Rate	Data	Steps	GitHub
Amphion (HiFiGAN)	3.55	161.2	24khz	LibriTTS LJSpeech VCTK	1.5M	https://github.com/open-mmlab/Amphion
HiFiGAN	3.41	229.35	22.05khz	LibriTTS LJSpeech VCTK	2.5M	https://github.com/jik876/hifi-gan
ESPNet	3.55	199.17	24khz	LibriTTS	2.5M	https://github.com/kan-bayashi/ParallelWaveGAN

TTS: Objective and subjective results

- Training data: LJSpeech

System	CER(↓)	WER(↓)	FAD(↓)	Speaker Similarity ↑	MOS
Amphion (VITS)	0.06	0.10	0.84	0.97	3.61 ± 0.1
Coqui/TTS (VITS)	0.06	0.12	0.54	0.98	3.69 ± 0.1
SpeechBrain （Fastspeech2）	0.06	0.11	1.71	0.94	3.54 ± 0.11
Tortoise-tts (Diffusion-based model)	0.05	0.09	1.90	0.55	3.61 ± 0.11
ESPNet (VITS)	0.07	0.11	1.28	0.99	3.57 ± 0.11

TTS: FastSpeech2

- Recipe
  - https://github.com/open-mmlab/Amphion/tree/main/egs/tts/FastSpeech2

FastSpeech2 Recipe

In this recipe, we will show how to train FastSpeech2 using Amphion's infrastructure. FastSpeech2 is a non-autoregressive TTS architecture that utilizes feed-forward Transformer blocks.

There are four stages in total:

1. Data preparation
2. Features extraction
3. Training
4. Inference

> **NOTE:** You need to run every command of this recipe in the `Amphion` root path:

```
cd Amphion
```

1. Data Preparation

Dataset Download

You can use the commonly used TTS dataset to train TTS model, e.g., LJSpeech, VCTK, LibriTTS, etc. We strongly recommend you use LJSpeech to train TTS model for the first time. How to download dataset is detailed here.

TTS: VITS

- Recipe
  - https://github.com/open-mmlab/Amphion/tree/main/egs/tts/VITS

VITS Recipe

In this recipe, we will show how to train VITS using Amphion's infrastructure. VITS is an end-to-end TTS architecture that utilizes conditional variational autoencoder with adversarial learning.

There are four stages in total:

1. Data preparation
2. Features extraction
3. Training
4. Inference

> **NOTE:** You need to run every command of this recipe in the `Amphion` root path:

```
cd Amphion
```

1. Data Preparation

Dataset Download

You can use the commonly used TTS dataset to train TTS model, e.g., LJSpeech, VCTK, Hi-Fi TTS, LibriTTS, etc. We strongly recommend using LJSpeech to train single-speaker TTS model for the first time. While for training multi-speaker TTS model for the first time, we would recommend using Hi-Fi TTS. The process of downloading dataset has been detailed here.

Readings

- Interspeech 2022 TTS tutorial
  - https://github.com/tts-tutorial/interspeech2022/blob/main/INTERSPEECH_Tutorial_TTS.pdf
- Text-to-Speech Synthesis
  - https://www.cambridge.org/core/books/texttospeech-synthesis/D2C567CEF939C7D15B2F1232992C7836