CS 224S / LINGUIST 285
Spoken Language Processing

Andrew Maas
Stanford University
Spring 2022

Lecture 17: TTS: Getting good results. Concatenative and Parametric Approaches

Original slides by Dan Jurafsky, Alan Black, & Richard Sproat

Outline for today

- Recipe for building great TTS
- Concatenative waveform synthesis:
  - Diphone Synthesis
  - Unit Selection Synthesis
  - Joining Units
- What to predict in parametric synthesis

Poster session next week!

- **Tuesday May 31 5:30pm – 7:30pm**
  Hewlett Lawn
- Print your poster (9 printed slides works!)
  - We provide poster boards + easels
  - Set up your poster by 5:30
- Present your poster 5:30-7:30
  - 2 mins walk through for audience
  - At least one person attend poster during session

Project milestone feedback

- Overall great progress on ambitious projects!
- Median progress
  - Dataset selected + working with it
  - Baseline methods implemented and tested
  - Many debugging complex baselines / off the shelf models
- Comments in gradescope

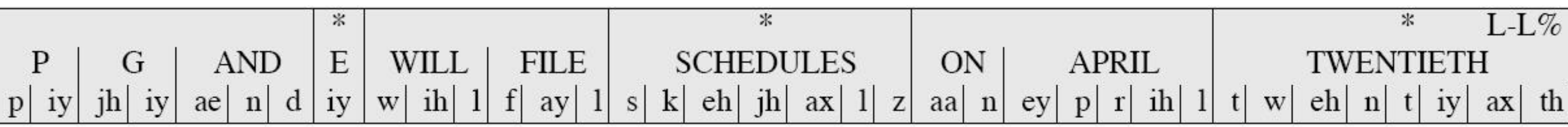
Project milestone feedback

- Get a complete (data prep, train, evaluate) experiment pipeline working with simple models first
- When deciding what to try next:
  - Form a hypothesis about what is broken and how to improve (include in your final paper)
  - Don’t lose track of your high level project goal!
- Include in final reports:
  - Present your project motivation, what you tried, and progress you made in context of overall project goal.
  - It’s okay if various deep learning approaches don’t work as well as you hope.
  - Make sure you describe why you tried what you tried, and control/debugging experiments as neede

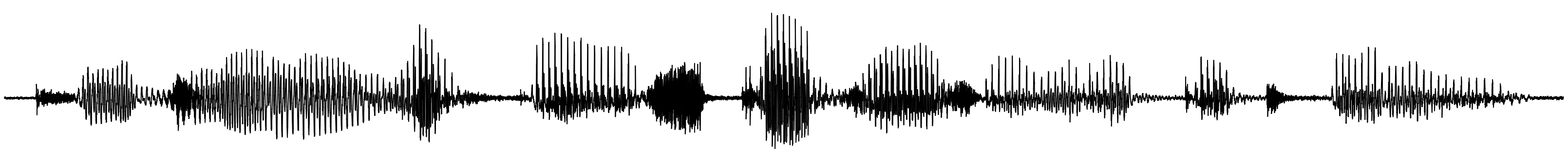
The two stages of TTS

PG&E will file schedules on April 20.

1. Text Analysis: Text into intermediate representation:



2. Waveform Synthesis: From the intermediate representation into waveform



Google audiobook TTS demo

Example state of the art TTS voices for narration. Link

So with modern deep learning TTS, we can get *great* results. What does it take to build such a system?

TTS is getting quite good! What is the recipe for building great TTS?

- Evaluation & measurement
  - Choose criteria (natural, emotive?)
  - Set up *human* evaluation listening tests
- Data collection
  - TTS acoustic quality limited by collected data
  - Require emotional range, expresiveness
- Modeling
  - Deep learning systems work best
  - Concatenative systems easier to build fast
  - Design controllable interface for developer

Evaluation of TTS

- Evaluation of TTS generally requires humans!
  - Listening test paradigm. Listen to example utterances, rate various aspects (naturalness, intelligibility, friendliness, expressiveness, etc.). Scale of 1-5
  - Mean opinion score (MOS). Average of ratings
  - AB Tests (prefer A, prefer B) (preference tests)
- Intelligibility Tests
  - Did the human hear the correct thing? Can test task completion, writing what was said, or simply rating
- Overall Quality Tests
  - A/B preference test vs human narrator is “ceiling”

Evaluation of TTS

- Diagnostic Rhyme Test (DRT)
  - Humans do listening identification choice between two words differing by a single phonetic feature
    - Voicing, nasality, sustenation, sibilation
  - 96 rhyming pairs
  - Veal/feel, meat/beat, vee/bee, zee/thee, etc
    - Subject hears “veal”, chooses either “veal or “feel”
    - Subject also hears “feel”, chooses either “veal” or “feel”
  - % of right answers is intelligibility score.

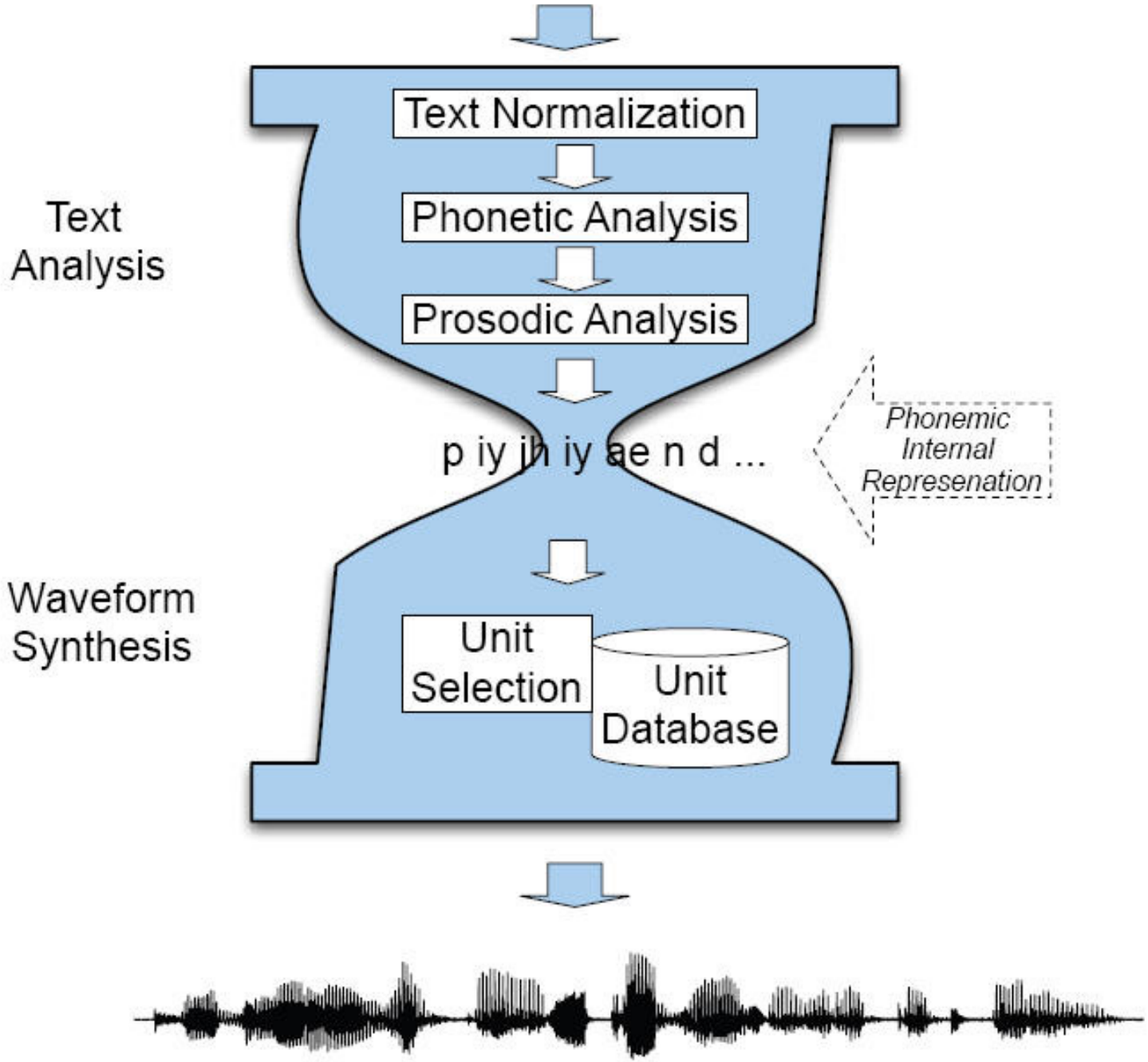
Data collection for TTS

- Great acoustic quality
  - At least 16 kHZ
  - Good microphone
  - Minimal background noise (including page turning and breathing!)
- Emotional and phonetic range to match application
  - System will clone accent of single speaker
  - Must collect emotional speech if TTS needs to produce it
  - Read vs conversational speech is different. Simulate human-human conversations with role play possibly
- Enough data
  - ~10 hours might be enough for read speech (single speaker)
  - Transfer learning enables some data sharing

Recording for Google Assistant

- Great recording conditions, attention to prosody, units for common phrases
- More important for unit selection systems, but data quality can be limiting factor for modern TTS
- https://www.youtube.com/watch?v=qnGNfz7JiZ8

PG&E will file schedules on April 20.



Waveform Synthesis

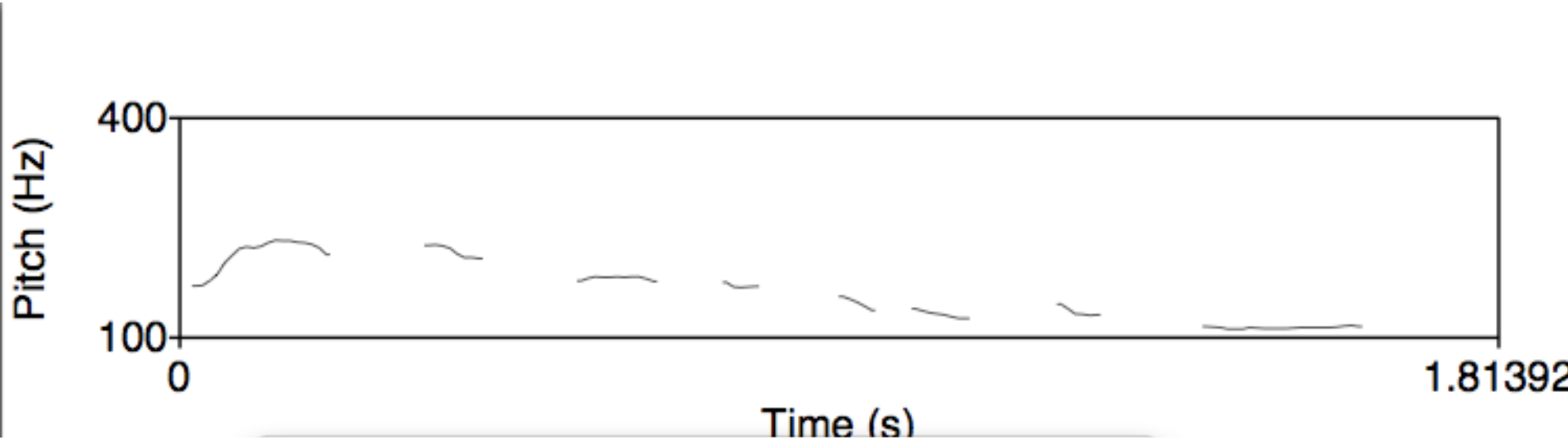
- Given:
  - String of phones
  - Prosody
    - Desired F0 for entire utterance
    - Duration for each phone
    - Stress value for each phone, possibly accent value
- Generate:
  - Waveforms

F0 Generation

- By rule
- By linear regression
- Some constraints
  - By accents and boundaries
  - F0 declines gradually over an utterance ("declination")

Declination

- F0 tends to decline throughout a sentence



F0 Generation by rule

Generate a list of target F0 points for each syllable

For example:

Generate simple H* “hat” accent (fixed speaker-specific F0 values) with 3 pitch points: [110, 140, 100]

Modified by

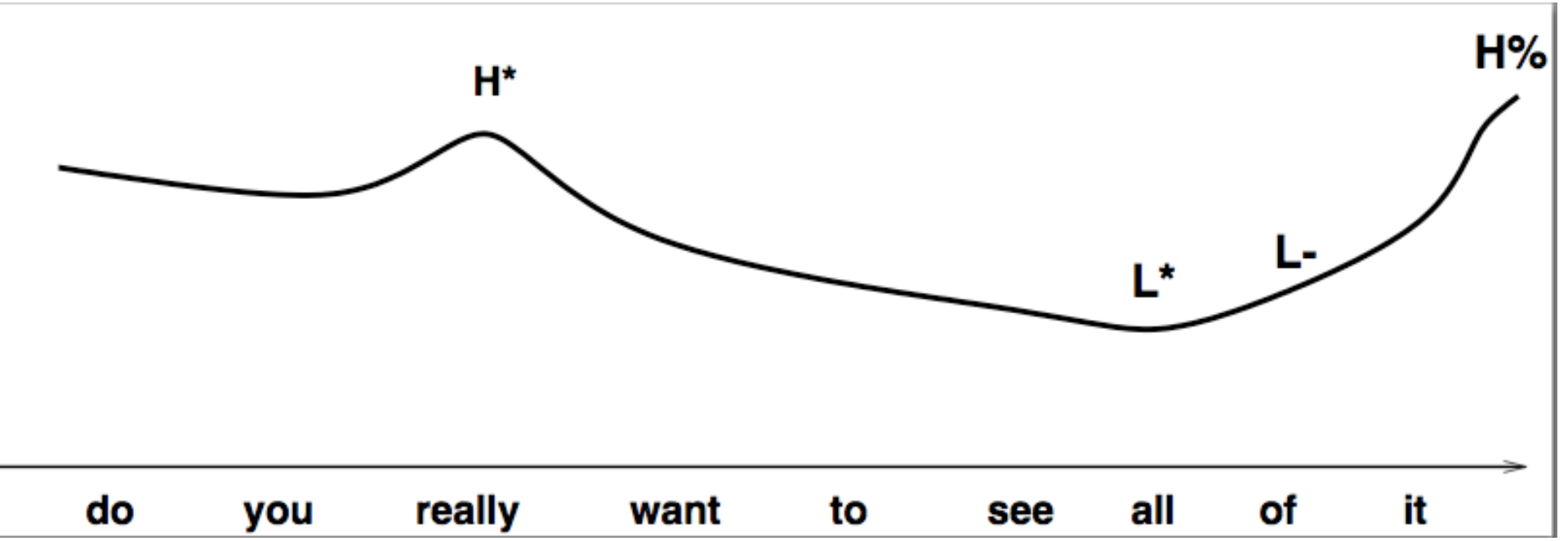
- gender,
- declination,
- end of sentence,
- etc.

F0 generation by regression

- Supervised machine learning
- Predict: value of F0 at 3 places in each syllable
- Predictor features:
  - Accent of current word, next word, previous
  - Boundaries
  - Syllable type, phonetic information
  - Stress information
- Need training sets with pitch accents labeled
- F0 is generally defined relative to pitch range
  - Range between baseline and topline frequency in an utterance

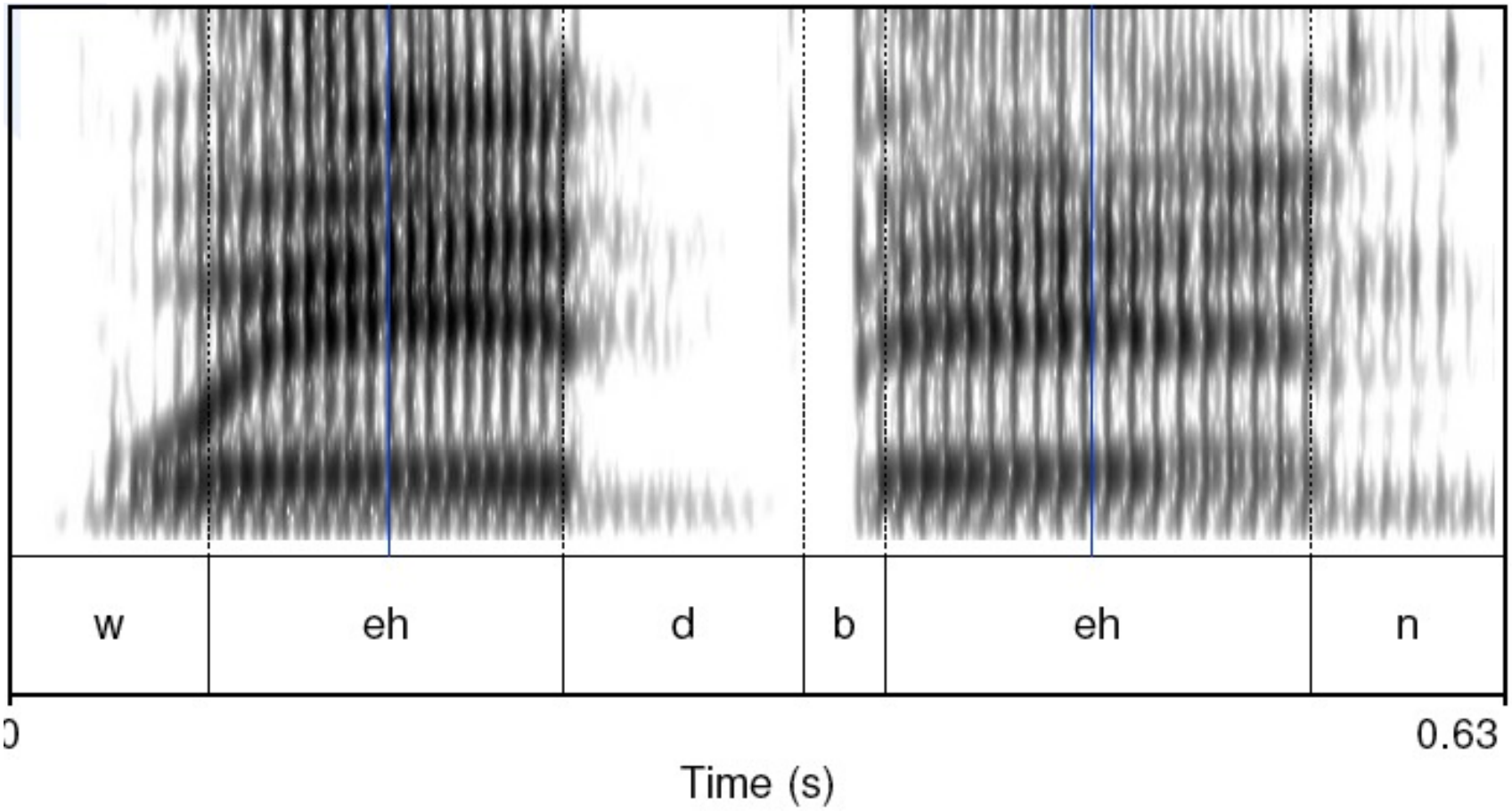
Internal Representation: Input to Waveform Synthesis

				H*										L*		L- H%			
do		you		really				want				to		see		all		of	it
d	uw	y	uw	r	ih	l	iy	w	aa	n	t	t	ax	s	iy	ao	l	ah	v	ih	t
110	110	50	50	75	64	57	82	57	50	72	41	43	47	54	130	76	90	44	62	46	220



Diphones

- Mid-phone is more stable than edge:



Diphone TTS architecture

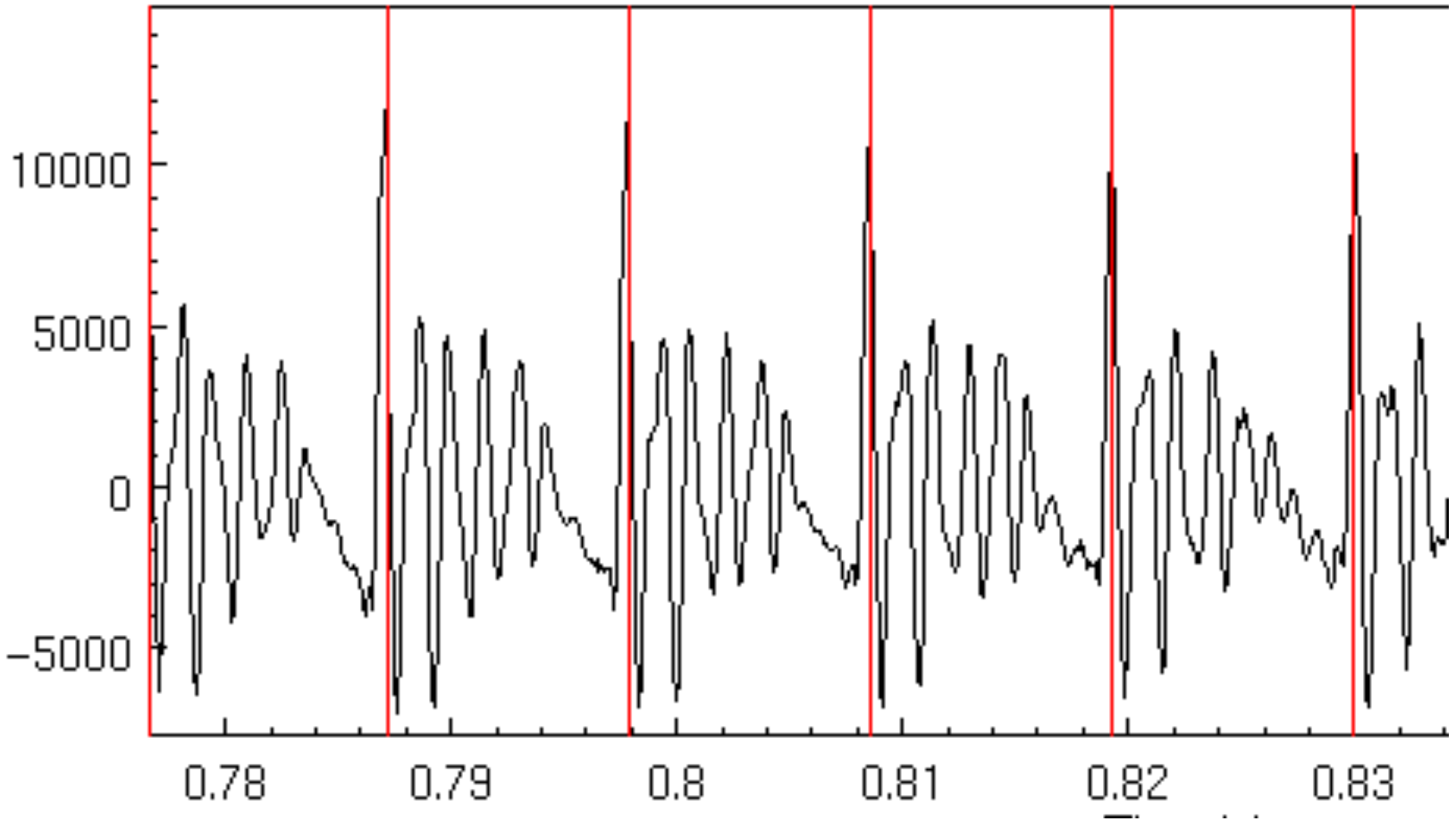
- Training:
  - Choose units (kinds of diphones)
  - Record 1 speaker saying 1 example of each diphone
  - Mark the boundaries of each diphones,
    - cut each diphone out and create a diphone database
- Synthesizing an utterance,
  - grab relevant sequence of diphones from database
  - concatenate the diphones, doing slight signal processing at boundaries
  - **use signal processing to change the prosody (F0, energy, duration) of diphone sequence**

Diphones

- mid-phone is more stable than edge
- Need ~|phones|^2 number of units
  - Some combinations don't exist (hopefully)
  - ATT (Olive et al. 1998) system had 43 phones
    - 1849 possible diphones
    - Phonotactics ([h] only occurs before vowels), don't need to keep diphones across silence
    - Only 1172 actual diphones
  - May include stress, consonant clusters
    - So could have more
  - Lots of phonetic knowledge in design
- Database relatively small (by today's standards)
  - Around 8 megabytes for English (16 KHz 16 bit)

Slide from Richard Sproat

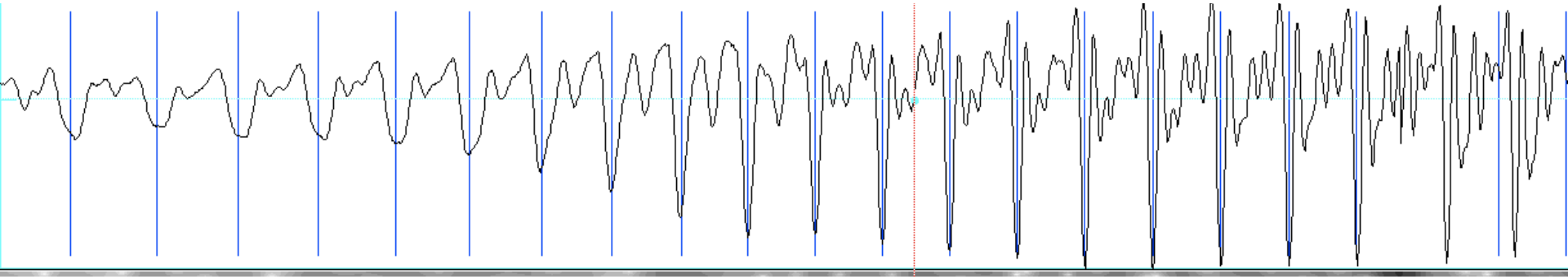
Speech as Short Term signals



Alan Black

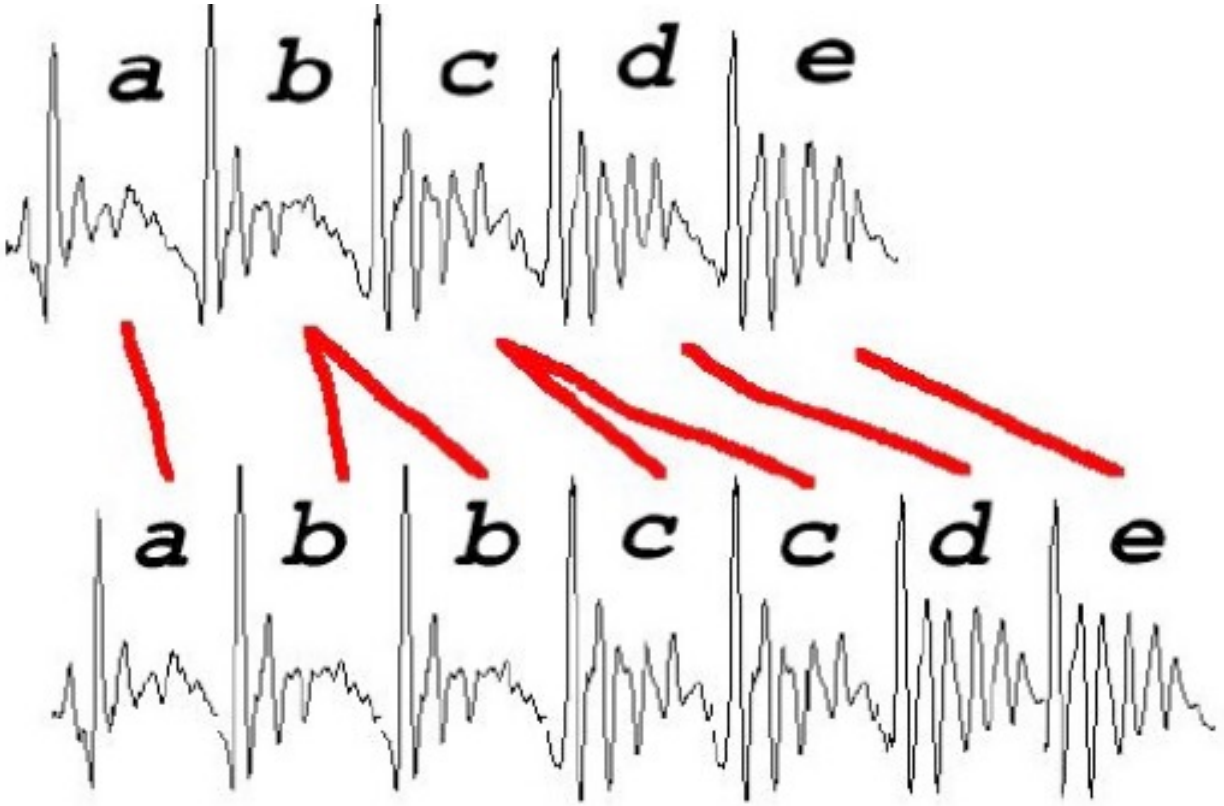
Epoch-labeling

- An example of epoch-labeling useing “SHOW PULSES” in Praat:



Duration modification

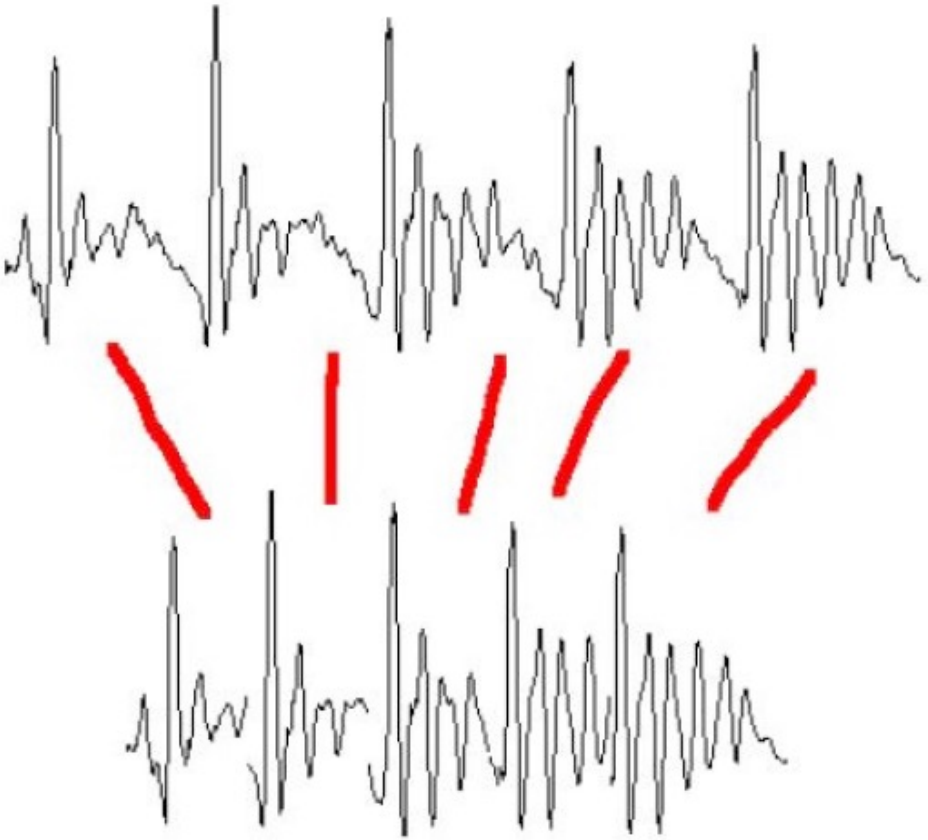
- Duplicate/remove short term signals



Slide from Richard Sproat

Pitch Modification

- Move short-term signals closer together/further apart

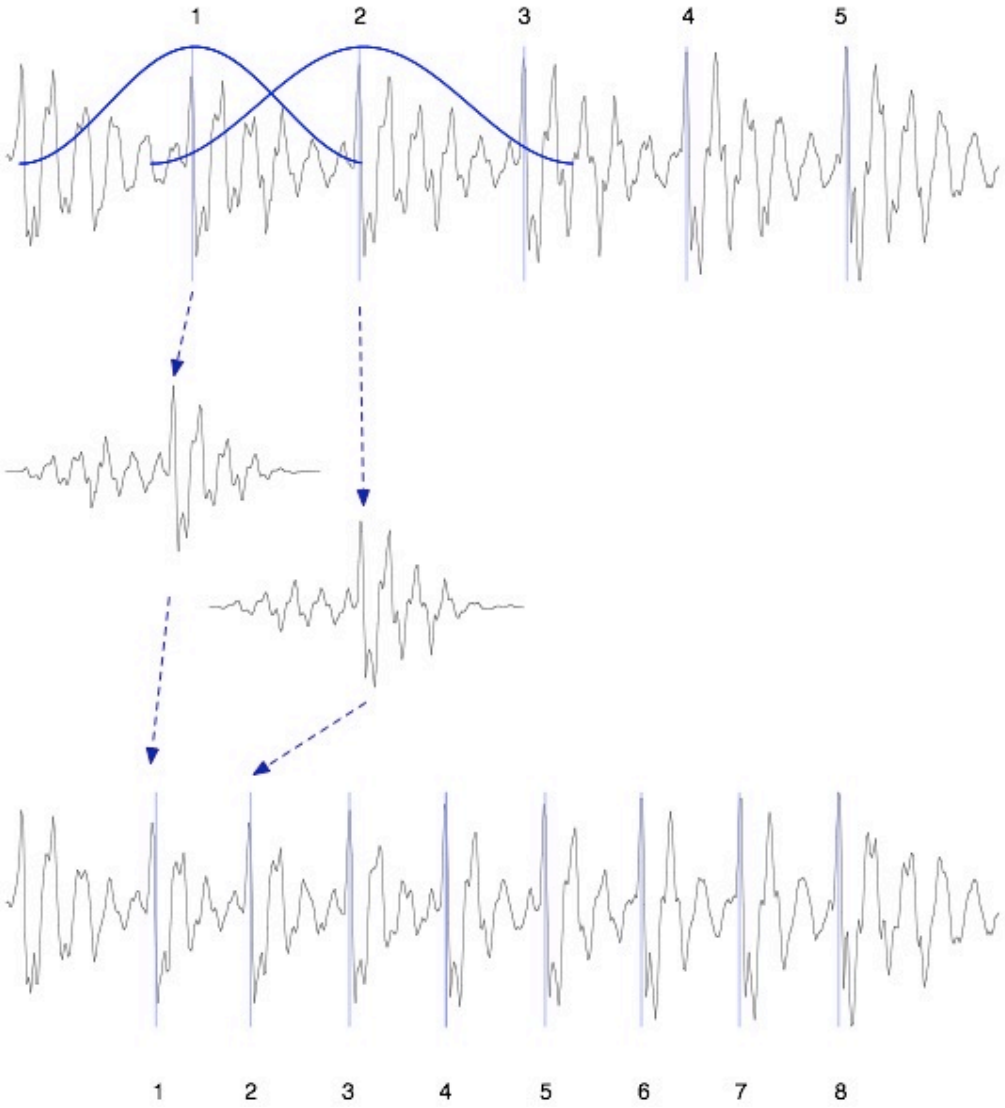


Slide from Richard Sproat

TD-PSOLA ™

- **T**ime-**D**omain (Windowed)
- **P**itch-**s**ynchronous
- **O**verlap-
- -and-**a**dd

- Efficient
- Wide range of possible Hz



Summary: Diphone Synthesis

- Well-understood, mature technology
- Augmentations
  - Stress
  - Onset/coda
  - Demi-syllables
- Problems:
  - Signal processing still necessary for modifying durations
  - Source data is still not natural
  - Units are just not large enough; can't handle word-specific effects, etc

Problems with diphone synthesis

- Signal processing methods like TD-PSOLA leave artifacts, making the speech sound unnatural
- Diphone synthesis only captures local effects
  - But there are many more global effects (syllable structure, stress pattern, word-level effects)

Unit Selection Synthesis

- Generalization of the diphone intuition
  - Larger units
    - From diphones to sentences
  - Many many copies of each unit
    - 10 hours of speech instead of 1500 diphones (a few minutes of speech)
  - Little or no signal processing applied to each unit
    - Unlike diphones

Why Unit Selection Synthesis

- Natural data solves problems with diphones
  - Diphone databases are carefully designed but:
    - Speaker makes errors
    - Speaker doesn't speak intended dialect
    - Require database design to be right
  - If it's automatic
    - Labeled with what the speaker actually said
    - Coarticulation, schwas, flaps are natural

"There's no data like more data"

- Lots of copies of each unit mean you can choose just the right one for the context
- Larger units mean you can capture wider effects

Unit Selection Intuition

- Given a big database
- For each segment (diphone) that we want to synthesize
  - Find the unit in the database that is the best to synthesize this target segment

Unit Selection Intuition

- What does “best” unit mean?
  - **Target cost**: Closest match to the target description, in terms of
    - Phonetic context
    - F0, stress, phrase position
  - **Join cost**: Best join with neighboring units
    - Matching formants + other spectral characteristics
    - Matching energy
    - Matching F0

$$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^{target}(t_i, u_i) + \sum_{i=2}^{n} C^{join}(u_{i-1}, u_i)$$

Targets and Target Costs

- A measure of how well a particular unit in the database matches the internal representation produced by the prior stages
- Features, costs, and weights
- Examples:
  - /ih-t/ from stressed syllable, phrase internal, high F0, content word
  - /n-t/ from unstressed syllable, phrase final, low F0, content word
  - /dh-ax/ from unstressed syllable, phrase initial, high F0, from function word “the”

Slide from Paul Taylor

Target Costs

- Comprised of k subcosts
  - Stress
  - Phrase position
  - F0
  - Phone duration
  - Lexical identity
- Target cost for a unit:

$$C^t(t_i, u_i) = \sum_{k=1}^{p} w_k^t C_k^t(t_i, u_i)$$

Slide from Paul Taylor

Join (Concatenation) Cost

- Measure of smoothness of join
- Measured between two database units (target is irrelevant)
- Features, costs, and weights
- Comprised of k subcosts:
  - Spectral features
  - F0
  - Energy
- Join cost:

$$C^j(u_{i-1}, u_i) = \sum_{k=1}^{p} w_k^j C_k^j(u_{i-1}, u_i)$$

Slide from Paul Taylor

Join costs

- The join cost can be used for more than just part of search
- Can use the join cost for optimal coupling (Isard and Taylor 1991, Conkie 1996), i.e., finding the best place to join the two units.
  - Vary edges within a small amount to find best place for join
  - This allows different joins with different units
  - Thus labeling of database (or diphones) need not be so accurate

Total Costs

- Hunt and Black 1996
- We now have weights (per phone type) for features set between target and database units
- Find best path of units through database that minimize:

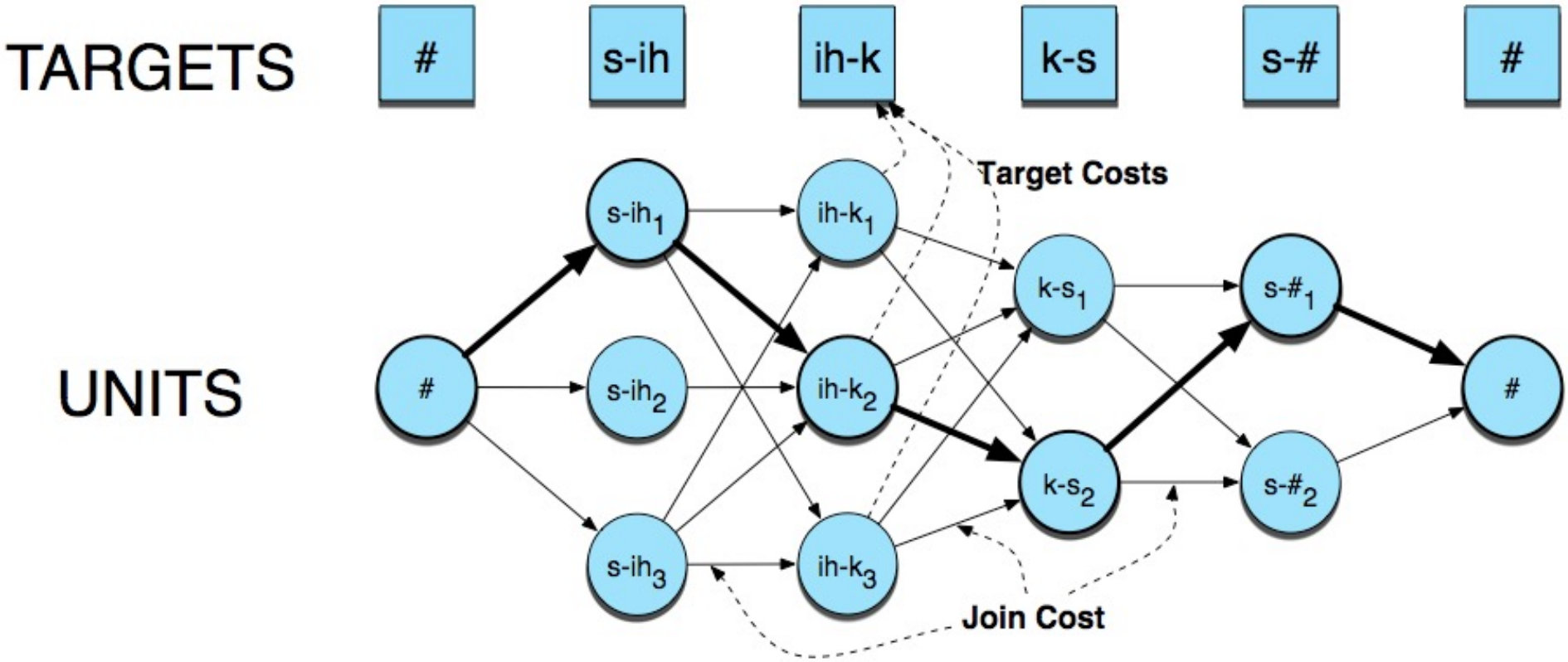
$$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^{target}(t_i, u_i) + \sum_{i=2}^{n} C^{join}(u_{i-1}, u_i)$$

$$\hat{u}_1^n = \underset{u_1,\ldots,u_n}{\operatorname{argmin}} C(t_1^n, u_1^n)$$

- Standard problem solvable with Viterbi search with beam width constraint for pruning

Slide from Paul Taylor

Unit Selection Search



Recap: Joining Units (+F0 + duration)

- For unit selection, just like diphone, need to join the units
  - Pitch-synchronously
- For diphone synthesis, need to modify F0 and duration
  - For unit selection, in principle also need to modify F0 and duration of selection units
  - But in practice, if unit-selection database is big enough (commercial systems)
    - no prosodic modifications (selected targets may already be close to desired prosody)

Unit Selection Summary

- Advantages
  - Quality is far superior to diphones
  - Natural prosody selection sounds better
- Disadvantages:
  - Quality can be very bad in places
    - HCI problem: mix of very good and very bad is quite annoying
  - Synthesis is computationally expensive
  - Can't synthesize everything you want:
    - Unit seleciton (unlike diphone synth) can't move emphasis
    - Unit selection gives good (but possibly incorrect) result

Slide from Richard Sproat

Parametric Synthesis

- Developed by Tokuda and Zen
- Proposed in mid-'90s, popular since 2007ish
- Big idea: Use classifiers/regressors to predict all of F0, duration, *spectral envelope.* Synthesize everything
- Initial work uses the same HMM we used for ASR, but in reverse

Key Questions in Parametric Synthesis

- *What parameters do we predict?* Usually MFCCs for spectrum, log F0, voicing/excitation
- *How do we combine them (vocoding)?* Exact parameterization and combining them well reduces robotic buzzy effects
- *How do we make predictions?* Choice of HMM, machine learning approaches. Less important than the vocoding/combination issues

What does the HMM produce?

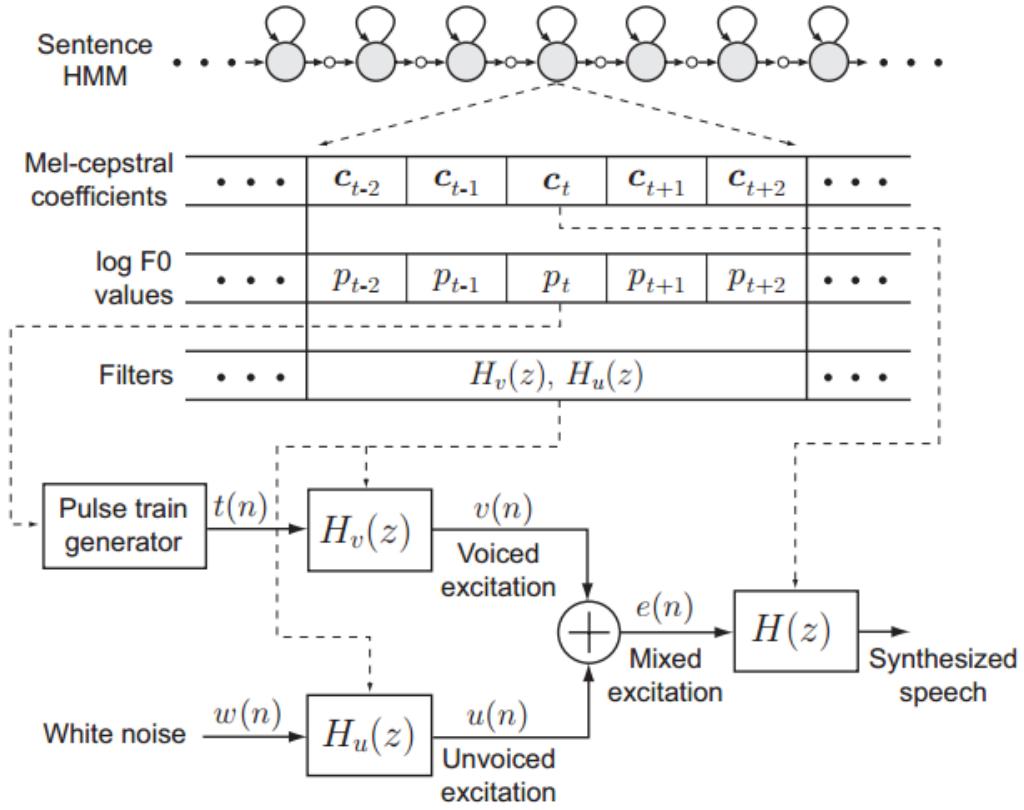
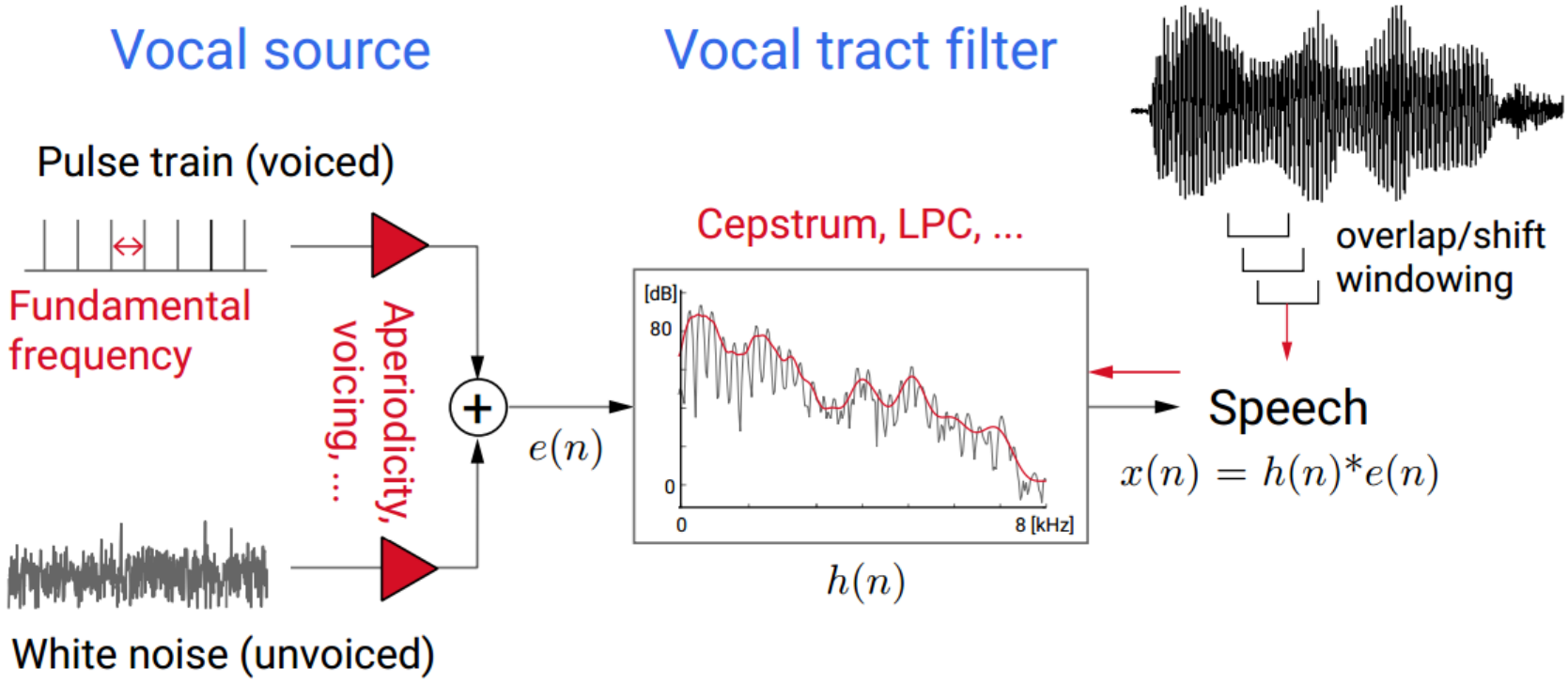


Figure 10: ML-based excitation scheme proposed by Maia et al. for HMM-based speech synthesis: filters $H_v(z)$ and $H_u(z)$ are associated with each state.

(Tokuda, Zen, & Black. 2009)

Synthesis with source-filter model

Piece-wise stationary, source-filter generative model $p(x \mid o)$



Slide from Heiga Zen

HTS Example

- Listen to the “low level” buzzy quality characteristic of most parametric systems
- Listen to clarity/impact of plosives compared to concatenative example
- Parametric

Unit Selection

Comparing vocoder/excitation models

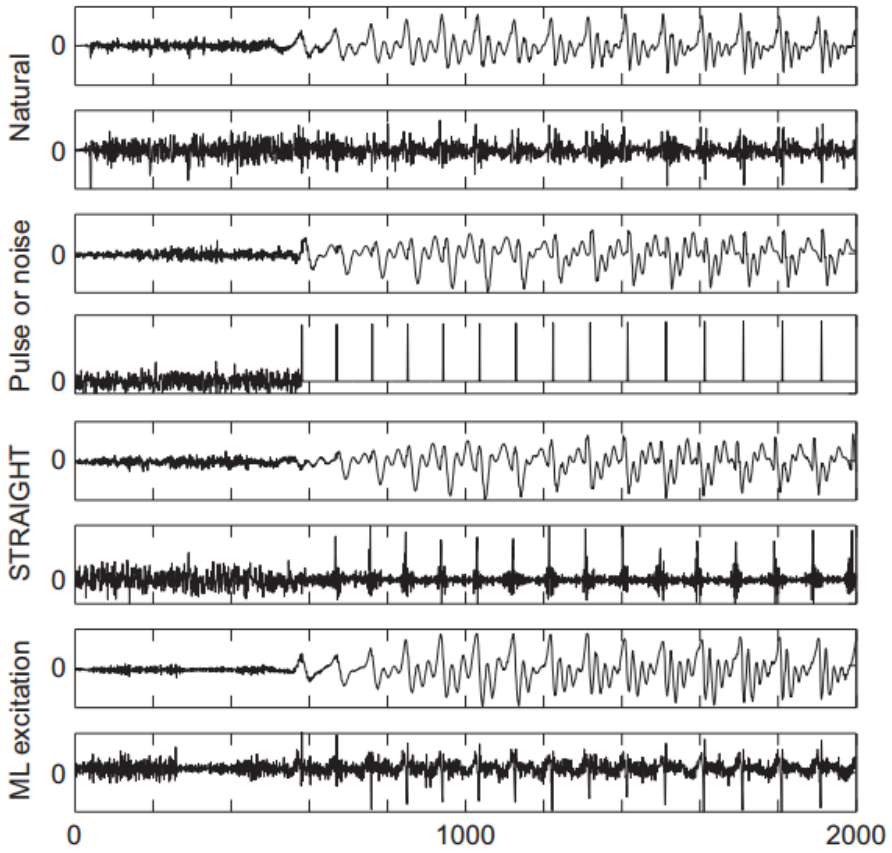


Figure 11: Waveforms from top to bottom: natural speech and its residual, speech and excitation synthesized with simple periodic pulse-train or white-noise excitation, speech and excitation synthesized with STRAIGHT vocoding method, and speech and excitation synthesized with ML excitation method.

(Tokuda, Zen, & Black. 2009)

Conclusions

- Common recipe for any TTS effort to achieve best results (data + evaluation are critical)
- Concatenative systems
  - Easier to get working quickly (no modeling work)
  - Low level signal processing and joins cause artifacts
  - Require large training sets for best coverage
- ”Editing prosody” is critical for human-like TTS
  - Parametric systems expose interfaces to predict/control duration, F0, etc.
  - No natural way to do this in concatenative systems

Slide from Richard Sproat