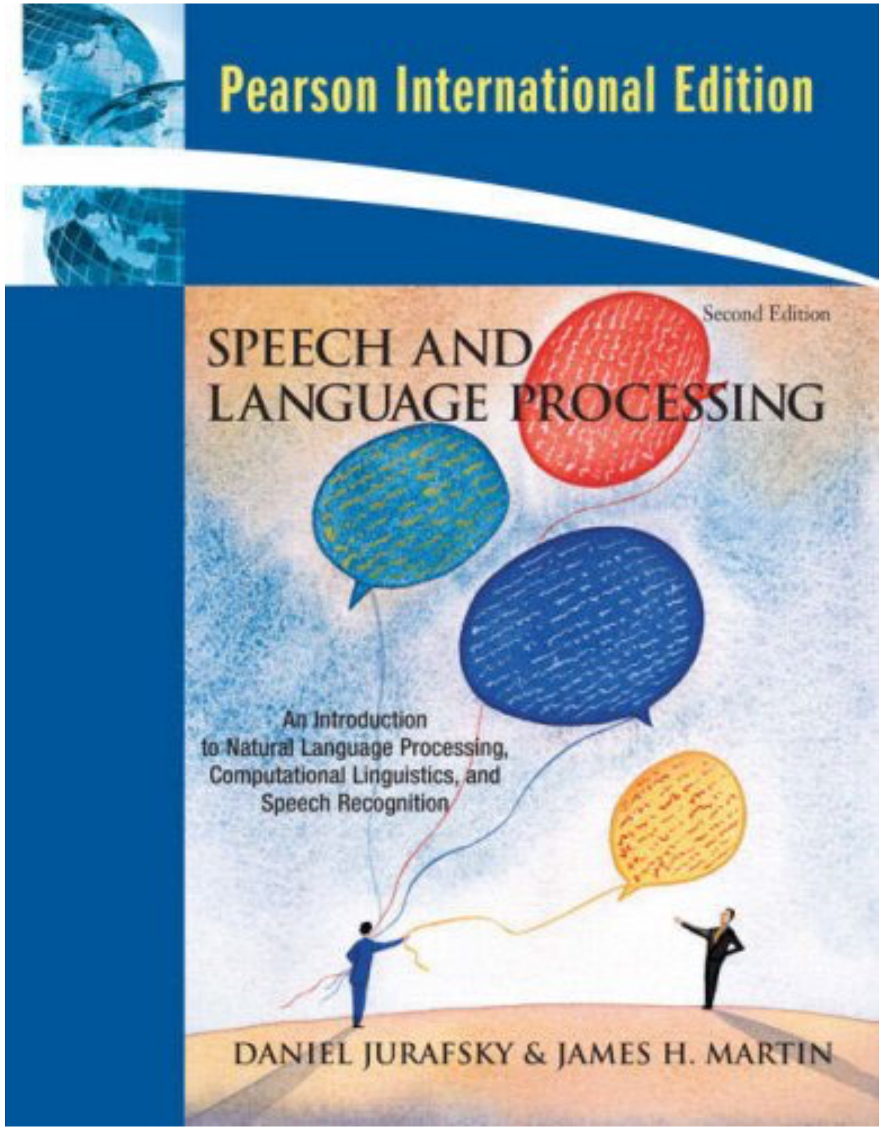
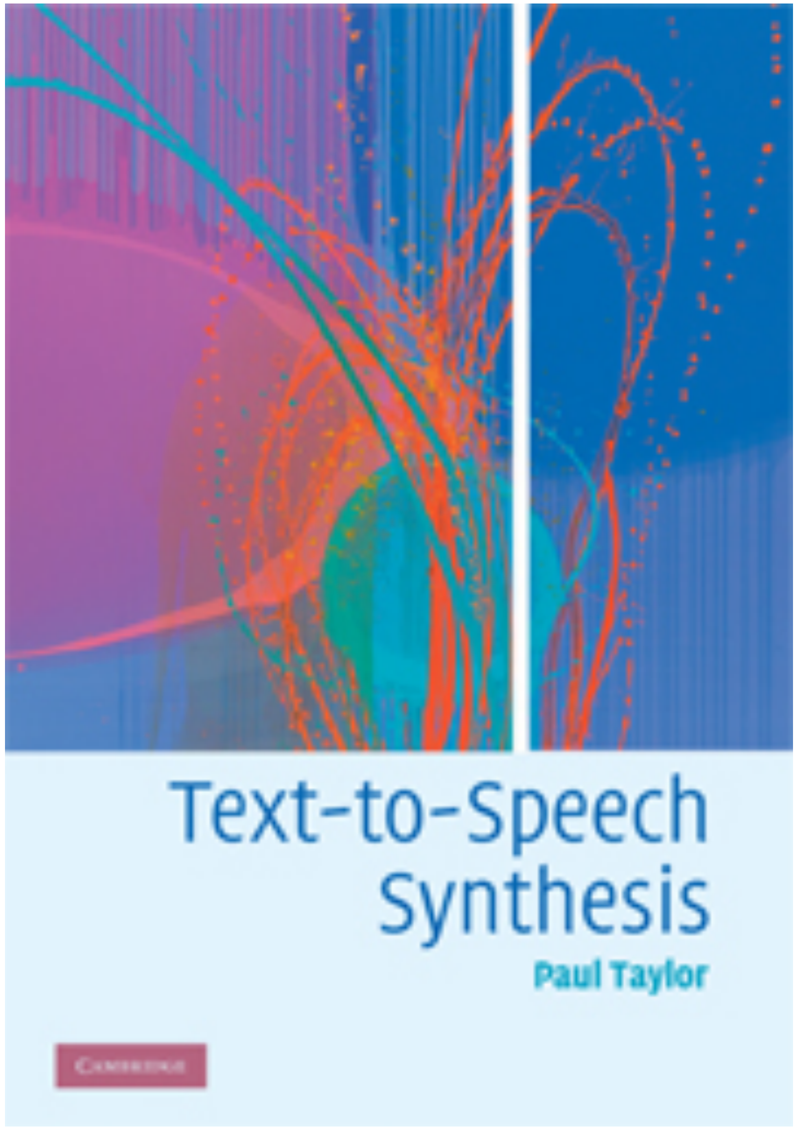
Text Processing for Speech Synthesis

Simon King

Centre for Speech Technology Research

University of Edinburgh

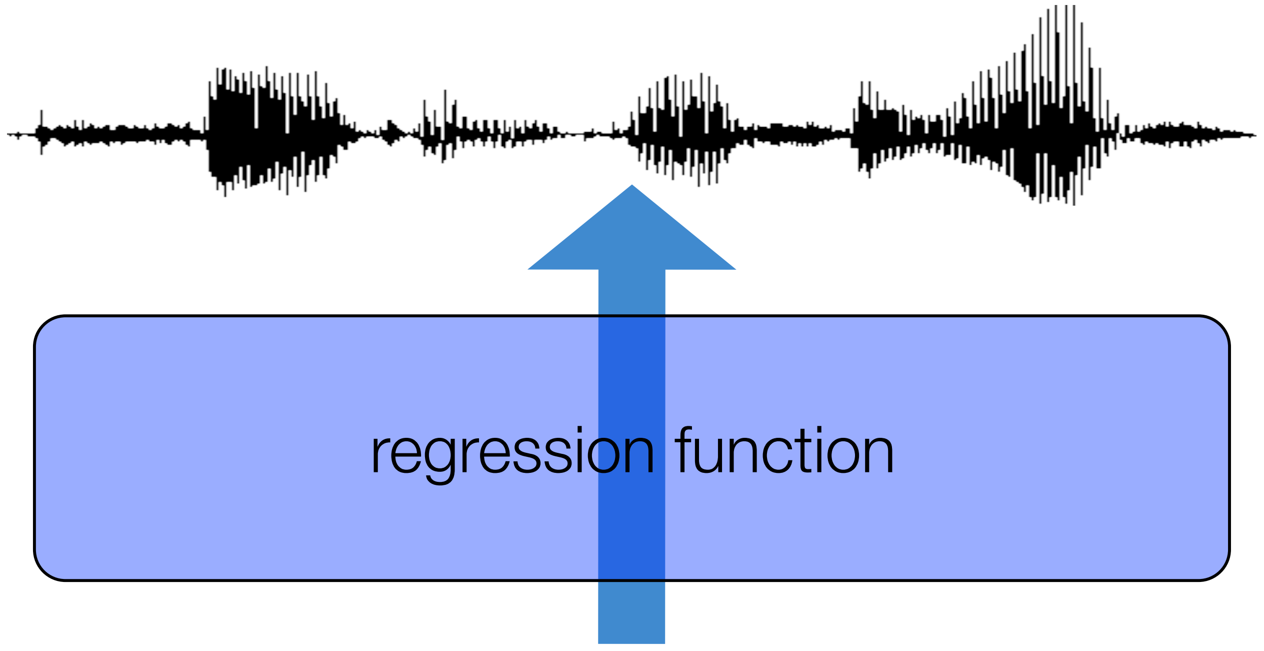

Pearson International Edition
Second Edition
SPEECH AND LANGUAGE PROCESSING
An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition
DANIEL JURAFSKY & JAMES H. MARTIN


Text-to-Speech Synthesis
Paul Taylor
Cambridge

What are you going to learn?

- What does a typical **front-end** do?
  - what will its output be? how will that be used for waveform generation?
- Starting from raw text
  - tokenisation: splitting text into smaller units
  - normalisation: converting everything to words
- Word processing
  - Morphological analysis (not in detail)
  - Part-of-speech tagging
- Pronunciation
  - Dictionary & letter-to-sound
  - ***Classification And Regression Trees (CARTs)***
- Prosody
- Techniques for avoiding most of the above (!)

Looking ahead: speech synthesis as a regression problem

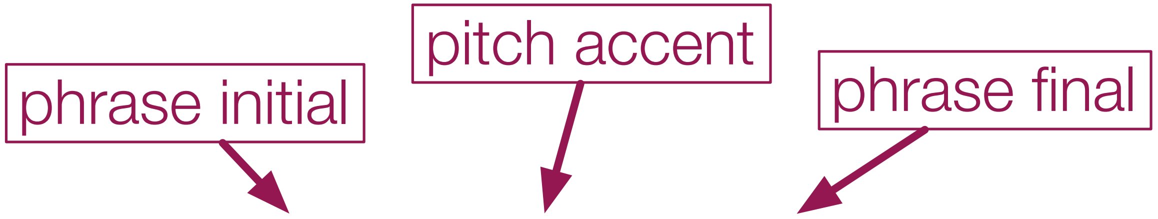


Input text to be converted to speech?

Text Processing for Speech Synthesis

Introduction

Two-stage pipeline
1. From text to linguistic specification



sil dh ax k ae t s ae t sil

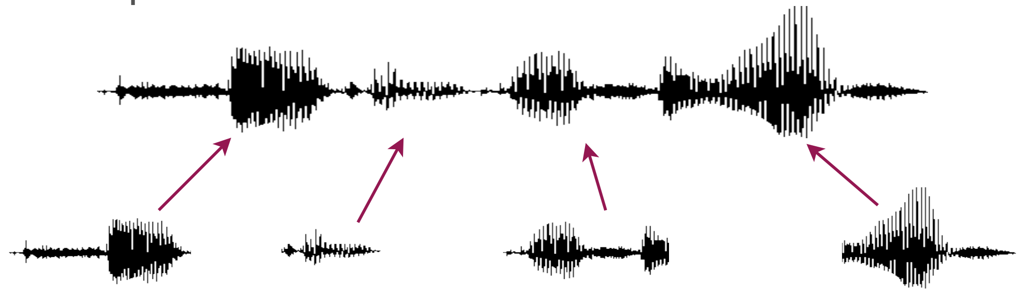
"the cat sat"

DET NN VB

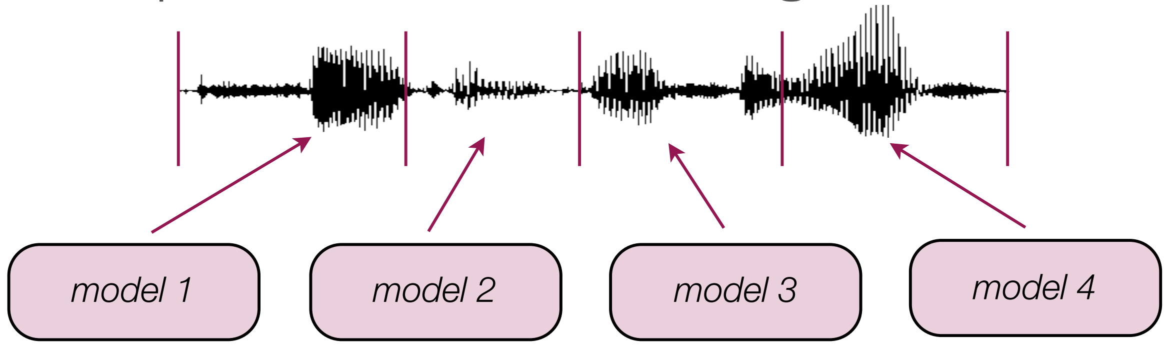
((the cat) sat)

Two-stage pipeline
2. From linguistic specification to waveform

- **Concatenation** builds up the utterance from units of recorded speech:



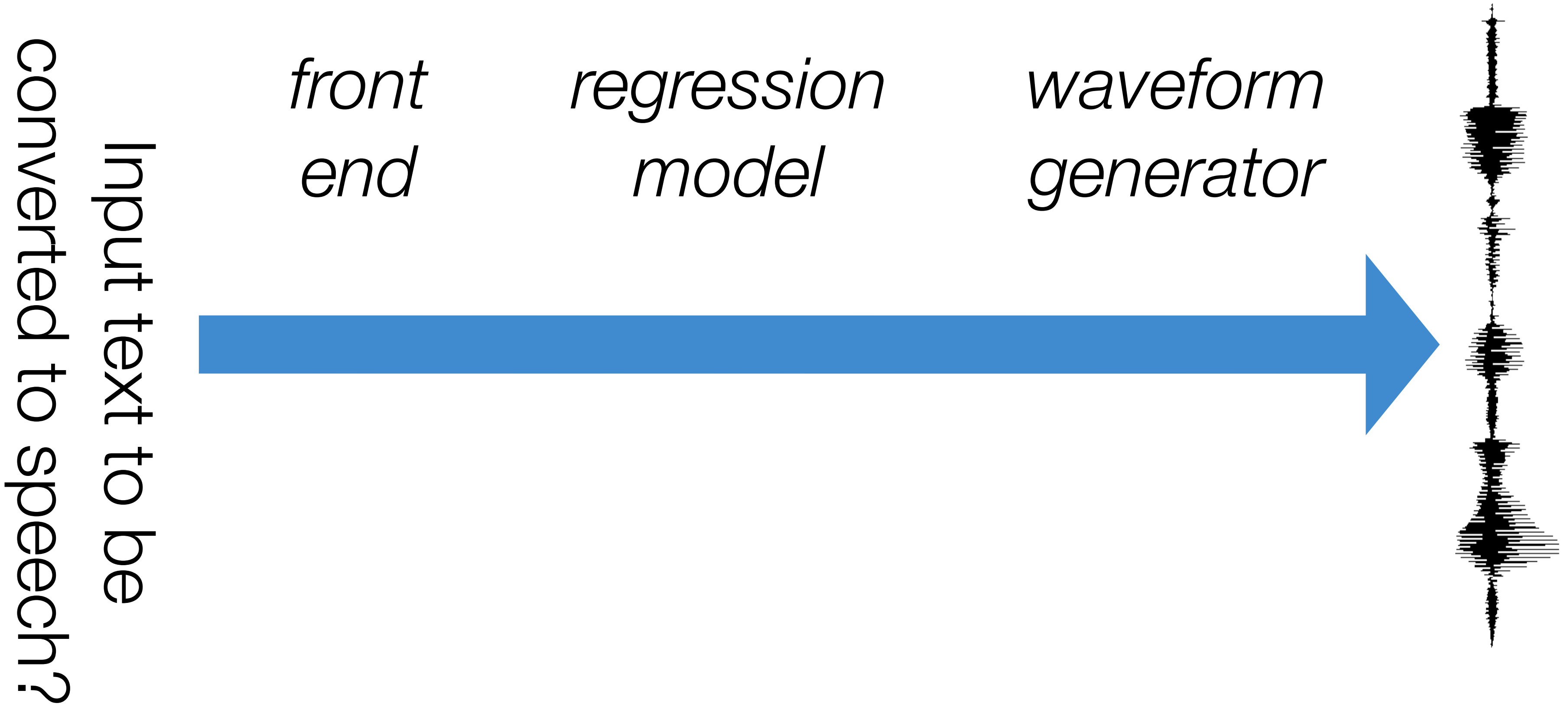
- **Generation** uses a sequence of models to generate the speech:



In fact, **three**-stage pipeline

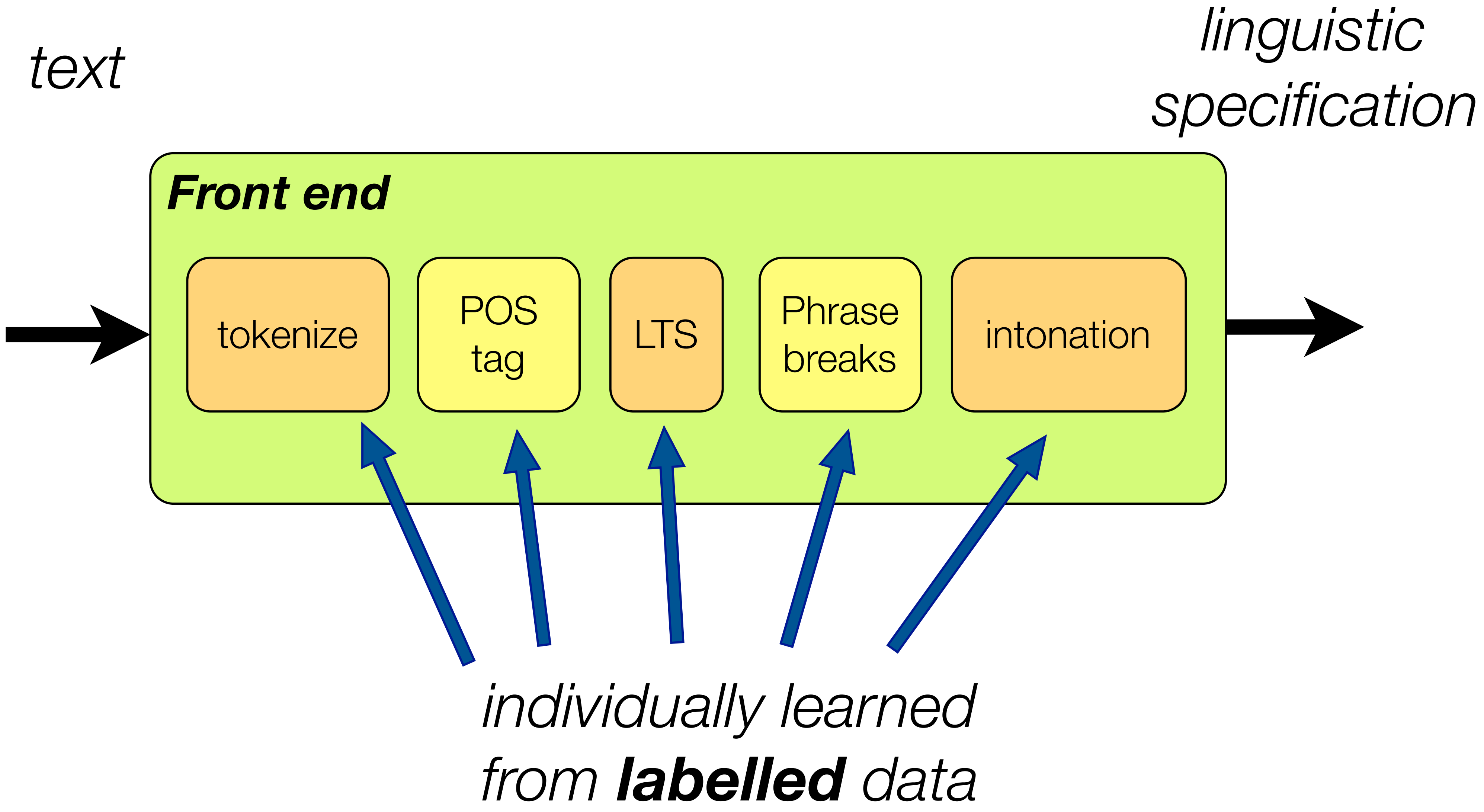
1. text to linguistic specification
2. linguistic specification to acoustic specification
3. acoustic specification to waveform

How much of the regression does each stage in the full TTS pipeline do?



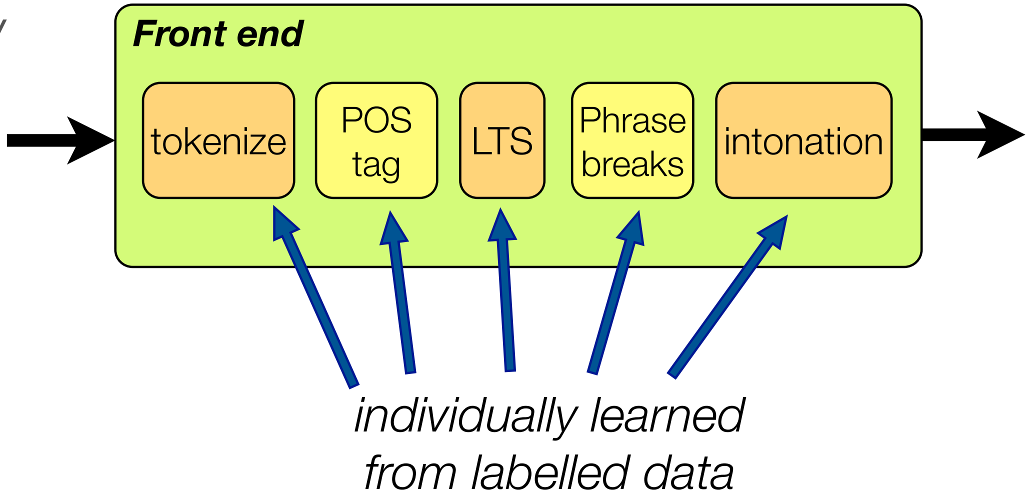
? IFF ASF Hybrid SPSS ?

Front end: typical pipeline

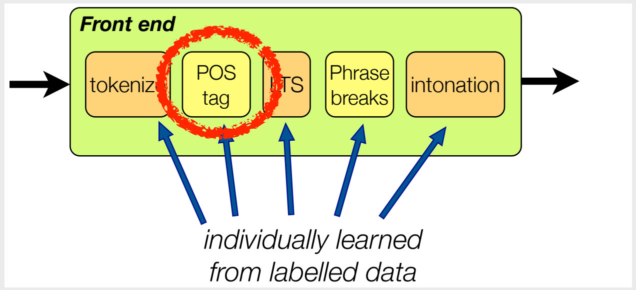


Front end: typical pipeline

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



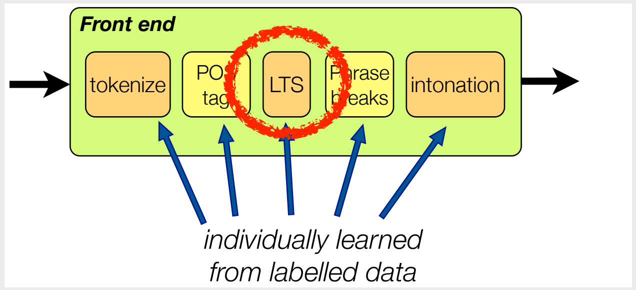
Example process I



- Part-of-speech tagger
- Accuracy is very high
- <u>But</u>
  - trained on **annotated** text data
  - **categories** are designed for text, not speech

NN	Director
IN	of
DT	the
NP	McCormick
NP	Public
NPS	Affairs
NP	Institute
IN	at
NP	U-Mass
NP	Boston,
NP	Doctor
NP	Ed
NP	Beard,
VBZ	says
DT	the
NN	push
IN	for
VBP	do
PP	it
PP	yourself
NN	lawmaking

Example process 2



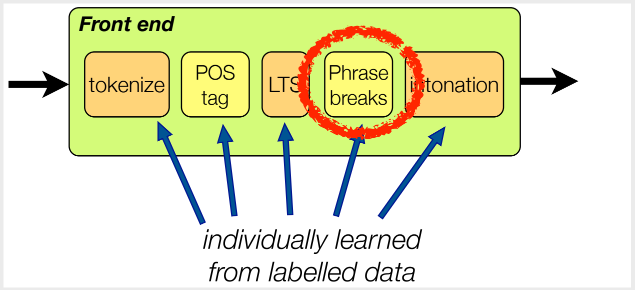
- Pronunciation model
  - dictionary look-up, *plus*
  - letter-to-sound model
- <u>But</u>
  - need deep **knowledge** of the language to design the phoneme set
  - human **expert** must write dictionary

This sequence is the annotated training data for our letter-to-sound predictor

```
ADVOCATING   AE1 D V AH0 K EY2 T IH0
ADVOCATION   AE2 D V AH0 K EY1 SH AH
ADWEEK   AE1 D W IY0 K
ADWELL   AH0 D W EH1 L
ADY   EY1 D IY0
ADZ   AE1 D Z
AE   EY1
AEGEAN   IH0 JH IY1 AH0 N
AEGIS   IY1 JH AH0 S
AEGON   EY1 G AA0 N
AELTUS   AE1 L T AH0 S
AENEAS   AE1 N IY0 AH0 S
AENEID   AH0 N IY1 IH0 D
AEQUITRON   EY1 K W AH0 T R AA0 N
AER   EH1 R
AERIAL   EH1 R IY0 AH0 L
AERIALS   EH1 R IY0 AH0 L Z
AERIE   EH1 R IY0
AERIEN   EH1 R IY0 AH0 N
AERIENS   EH1 R IY0 AH0 N Z
AERITALIA   EH2 R IH0 T AE1 L Y AH0
AERO   EH1 R OW0
```

A	-
E	EH1
R	R
I	IY0
A	AH0
L	L
S	Z

Example process 3



- Phrase-break prediction
  - binary classifier us
    sequence as input
- But
  - trained on **annota**
    spoken data
  - therefore very **sm**
    training set

This sequence is the annotated training data for our phrase break predictor

DT	NB
CD	NB
CD	NB
NN	NB
JJ	NB
NN	B

Break !

Text Processing for Speech Synthesis

A typical front end

A typical front-end, from Festival

- Text processing
  - Tokenisation; rules (e.g. for dates and numbers)
  - Part of speech tagging
  - Phrase break prediction
- Pronunciation
  - Lexicon
  - Letter-to-sound rules or decision tree (CART) trained on data
- Duration prediction
  - CART trained on data
- Intonation
  - TOBI accents predicted & realised using CART models

Text Processing for Speech Synthesis

“Text decoding” (Taylor, chapter 5)

Tokenisation

- The input to a TTS system can be any text, for example:

In 1871, Stanley famously said “Dr. Livingston, I presume”

- Punctuation is generally preserved, so this might be tokenised as:

(In) (1871) (,) (Stanley) (famously) (said) (“) (Dr.) (Livingston) (,) (I) (presume) (”)

- In some systems, the punctuation is stored as a feature of the preceding or following token
- Now we need to deal with each token, converting everything to **words**
  - **what is a word?** something we can list in a pronunciation dictionary

Abbreviations

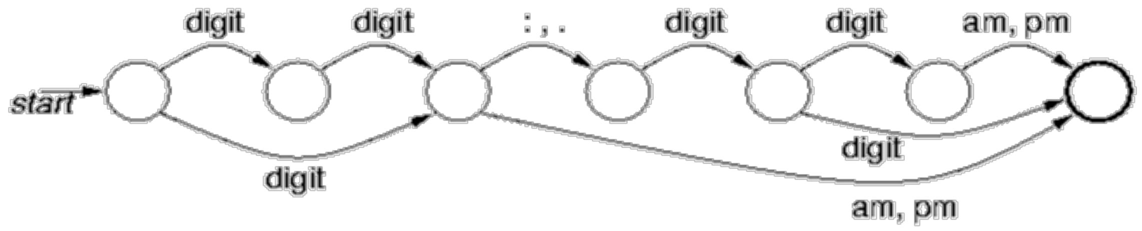
- In text, abbreviations are often used, but conventionally they are read out fully:
- Even simple abbreviations can be ambiguous, e.g.:
  - Dr. Livingston vs. Livingston Dr.
  - St. James vs. James St.
  - V can be a roman numeral or Volts
  - 100m could be “100 million” or “100 metres” or “100 miles”, …

Numbers

- The interpretation of numbers is context sensitive
    - 2.16pm
    - 15:22
    - 2.1
    - 20/11/05
    - The 2nd
    - $100bn
    - 99p
    - 0131 651 3174

Detect, classify, expand

- **General approach to dealing with all non-standard words (NSWs)**
- Detection
    - hand-crafted rules (e.g.,regular expressions), statistical classifiers trained on data
- Classification
    - decision trees
- Expansion
    - lookup tables, rules, transducers



From words to sounds

- Once we have a sequence of fully spelled-out words, we start working towards a sequence of phonemes
- Morphology (optional - not very helpful for English)
- Part-of-speech (POS) tagging
- The lexicon
- Post-lexical rules
- Letter-to-sound (LTS) rules
    - can be done with a Classification and Regression Tree (CART)
    - which is a very widely applicable technique (e.g, clustering HMM states)

Part-of-Speech (POS)

- Some words have multiple possible POS categories
- We must **disambiguate** the POS:
    - without POS information, pronunciation might be ambiguous e.g. “lives”
    - POS will also be used to predict the prosody later on
- POS tagging is the process of determining a single POS tag for each word in the input; the method can be
    - deterministic, or
    - probabilistic

Penn treebank POS tag set

- CC Coordinating conjunction
- CD Cardinal number
- DT Determiner
- EX Existential there
- FW Foreign word
- IN Preposition or subordinating conjunction
- JJ Adjective
- JJR Adjective, comparative
- JJS Adjective, superlative
- LS List item marker
- MD Modal
- NN Noun, singular or mass
- NNS Noun, plural
- NNP Proper noun, singular
- NNPS Proper noun, plural
- PDT Predeterminer
- POS Possessive ending
- PRP Personal pronoun
- PRP$ Possessive pronoun
- RB Adverb
- RBR Adverb, comparative
- RBS Adverb, superlative
- RP Particle
- SYM Symbol
- TO to
- UH Interjection
- VB Verb, base form
- VBD Verb, past tense
- VBG Verb, gerund or present participle
- VBN Verb, past participle
- VBP Verb, non-3rd person singular present
- VBZ Verb, 3rd person singular present
- WDT Wh-determiner
- WP Wh-pronoun
- WP$ Possessive wh-pronoun
- WRB Wh-adverb

plus 9 tags for punctuation

Probabilistic POS tagging

- One of the simplest and most popular methods is to train models on labelled data (i.e., already tagged, by hand), combining
  - HMMs (Hidden Markov Models):
    - where the observations are words and the models are the POS classes (This will make more sense after the speech recognition part of the course)
  - N-grams
- The latest state-of-the-art taggers are extremely accurate. Festival's tagger is now somewhat dated, but performs well enough

Text Processing for Speech Synthesis

Pronunciation

The lexicon

- The lexicon entries have three parts:
  - Head word
  - POS
  - Pronunciation (in terms of phonemes)
- The POS is sometimes necessary to distinguish homographs, e.g.:

head	POS	phonemes
lives	NNS	l ai v z
lives	VBZ	l I v z

Syllables and lexical stress

- The lexicon will usually also mark syllable structure and lexical stress
    - present n (((p r eh z) 1) ((ax n t) 0))
    - present v (((p r iy z) 0) ((eh n t) 1))
- In Festival, there are three steps to find the pronunciation of a word:
    - Look up in main lexicon
    - If not found, look up in addenda (e.g. domain specific additional lexicon)
    - If not found, use letter-to-sound model
- The main lexicon is large and ordered to allow fast searching, the addenda contains a small number of words added by hand, and the letter-to-sound model will deal with the rest

Letter-to-sound

- If lexical lookup fails, we fall back on letter-to-sound rules
- Example:
    - The letter c can be realised as /k/, /ch/, /s/, /sh/, /ts/ or /ɛ/ [deleted]
    - We might write rules like:
        - If the “c” is word-initial and followed by “i” then map to /s/
        - If the “c” is word-initial and followed by “h” then map to /ch/
        - If ...
- This approach works well for Spanish, but performs very poorly for English
- In general, we want an automatic method for constructing these “rules”
    - The most popular form of model: a **classification tree**

Post-lexical rules

- The lexicon and letter-to-sound rules arrive at a pronunciation for each word *as it would be spoken in isolation*, known as the “**citation form**”
- Now we need to apply cross word and phrasal effects such as:
  - Vowel reduction
  - Phrase-final devoicing
  - r-insertion
- Since these effects are small in number, hand written rules work OK
- Festival has a mixture of
  - hard-wired rules (compiled into the C++ code), and
  - voice specific rules (implemented in Scheme which can be changed at run-time)

Text Processing for Speech Synthesis

Classification and Regression Trees

(we’ll consider the case of classification only here)

Video

http://www.speech.zone/classification-and-regression-trees-cart

CART – classification and regression trees

- These are decision trees for predicting the value of either a
  - Categorical variable (classification tree)
  - Continuous variable (regression tree)
- We'll consider only the categorical case, but the principles are the same for continuous variables
- The nodes in the tree are questions about features which describe the environment
- The tree is learned automatically from data
- Trees are human readable and editable (mostly)
- Concise and fast
- Automatically select predictors that are useful, ignores those that are not

Learning from data: the two main stages

- It’s very important to make a clear distinction between:

- **Learning the model from data (“training”)**
    - we obtain some labelled training data
    - we choose some form of model (e.g., classification tree)
    - we fit the model to the training data (e.g., grow the tree)

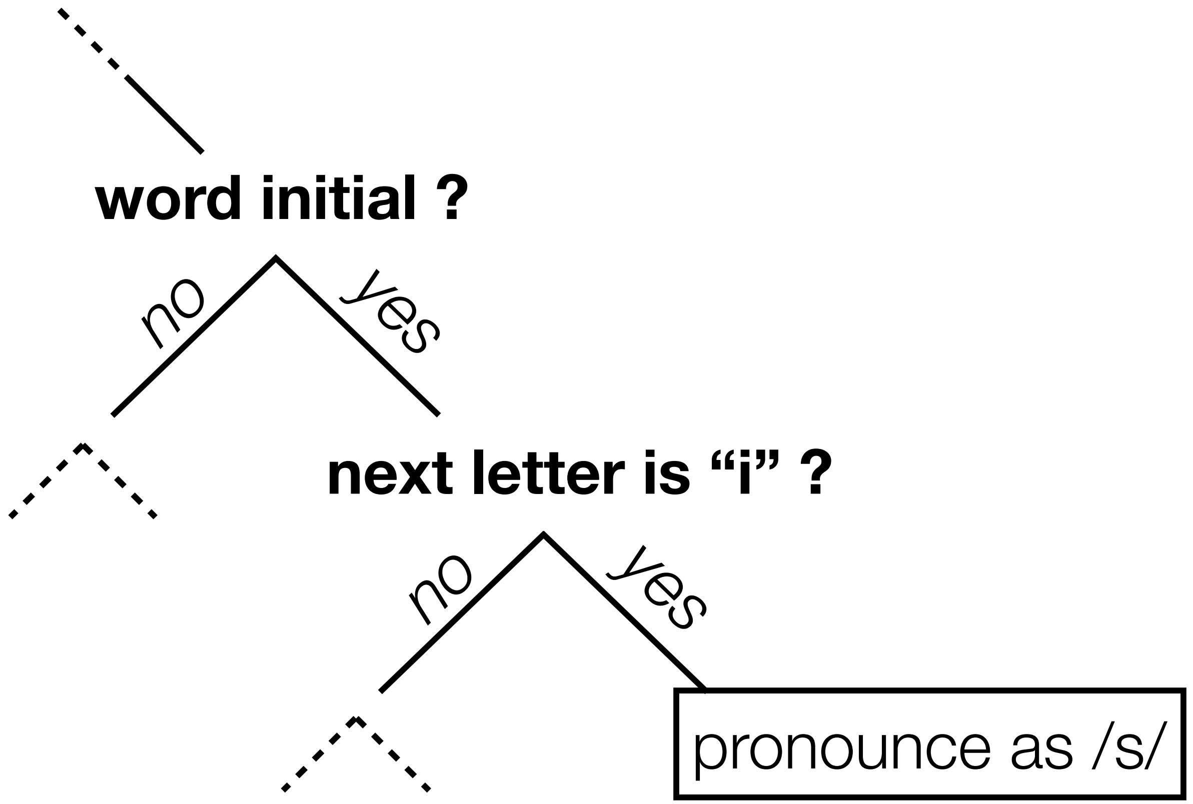
- **Using the model to make classifications, predictions, etc. (“testing”)**
    - we have some unlabelled test data
    - we use the model to label the test data

Predictors and predictees

- Predictors: things whose value we know (think independent variables)
- Can be just about anything
    - Continuously valued
    - Discrete (categorical)
- Predictee: the thing whose value you want to predict (think dependant variable)
- Letter-to-sound “rules” can be written as a classification tree
    - The predictors used for letter to sound rules might include: the surrounding context letters, position in the word, word boundary information

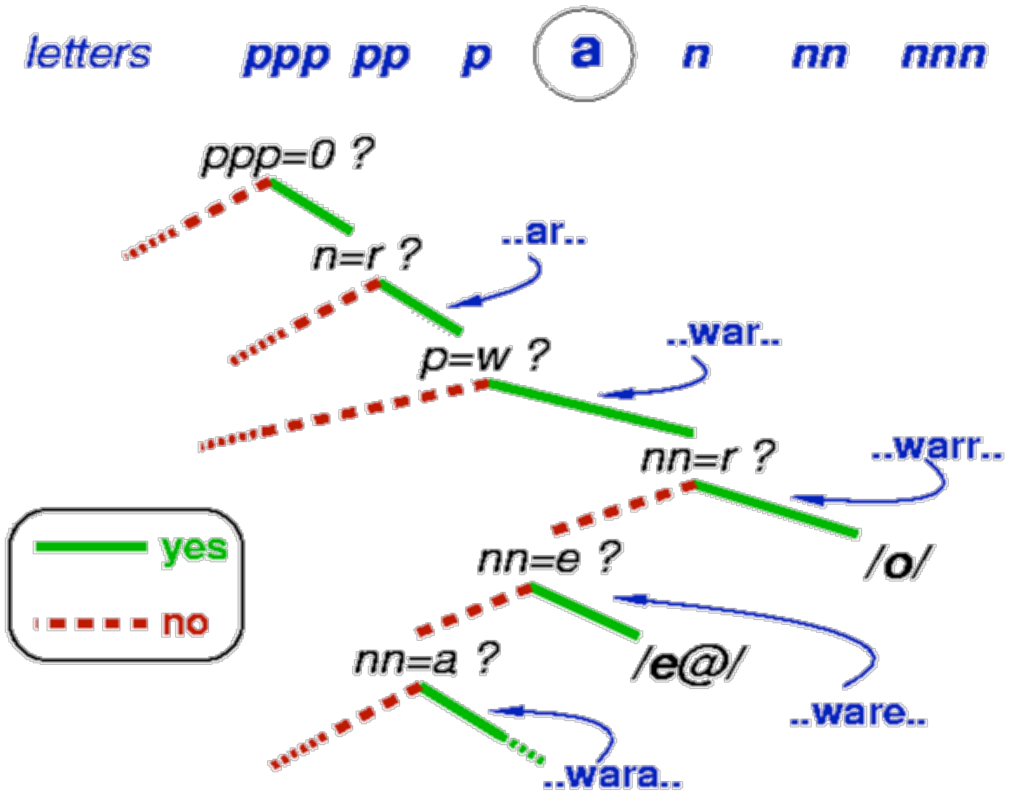
Classification trees are equivalent to ordered rules

- Here's a fragment of a tree - we've already decided the letter is "c" :



Part of Festival's LTS tree

- Here is a fragment of the LTS tree from Festival: letter "a" for British English



Learning a CART from data: prerequisites

- Before learning a CART model, we need to specify:
    - The predictors (sometimes called features)
    - The predictee
    - All the possible questions we can ask about the predictors

- The list of possible questions can be determined automatically (e.g., ask whether a categorical predictor is equal to each possible value it can take)
- The training algorithm will choose which questions to use, and where to put then in the decision tree

Questions

- For discrete predictors, question are simply of the form:
    - Is value of predictor equal to v ?
    - Is value of predictor in the set {u,v,w}?
- The number of possible questions of the first type is much smaller than for the second type
- For continuous predictors, questions are simply of the form:
    - Is the value of predictor greater than v
- To reduce the space of possible questions, can try only a fixed number of v values (e.g. 10). This is in effect **quantising** the continuous variable and then treating it as discrete

Learning a CART from data: algorithm

- At the start, all data is placed in a single set at the root node of the tree
- A question is placed in the tree, and the data is split according to it: the data is partitioned into two subsets, which descend the branches of the tree. This procedure is then recursively applied
- At each iteration, we need to decide:
    - Which question to put into the tree next?
        - need to measure how well each question splits the data, i.e., how coherent the resulting subsets are (e.g., measure variance for continuous data or entropy for discrete data)
    - Whether to stop growing the tree?
        - some stopping criterion is required, such as a minimum number of data points in a subset)

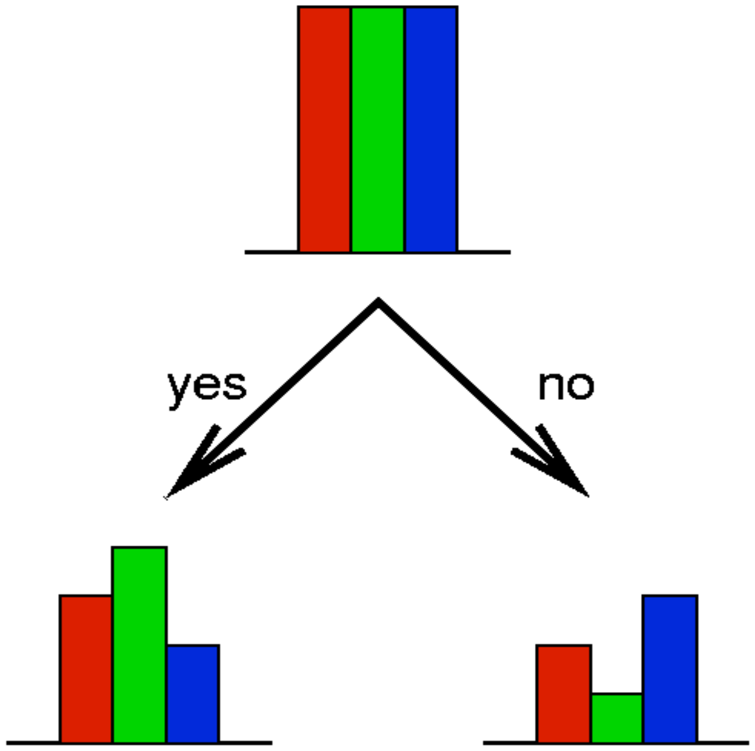
Learning a CART from data: pseudo code

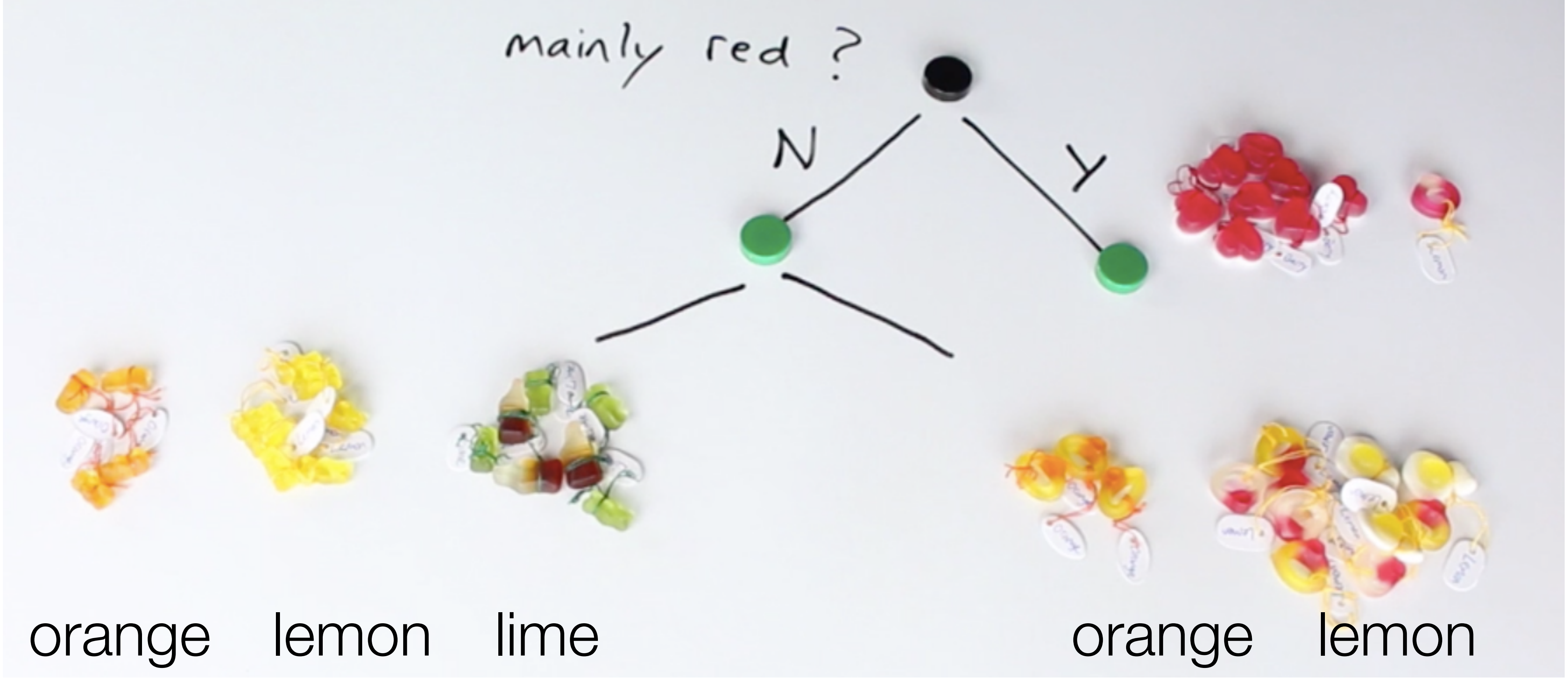
```
• Function: partition()
    • Consider each possible question in turn
        • Choose the question that splits the data into the most
          consistent two subsets
    • Place this question in the tree
    • Partition the data using question
    • Send the resulting subsets of data down the branches of
      the tree
    • Recurse: for each subset, call partition()
```

- To start the algorithm, we make a tree with only one node (the root), place all of the data there, and call partition() on it
- This type of algorithm is called a **greedy algorithm** – at a given point during the training procedure, a decision is made which gives the best outcome at that point, with no regard to how it will affect future outcomes. There is no backtracking

Partitioning

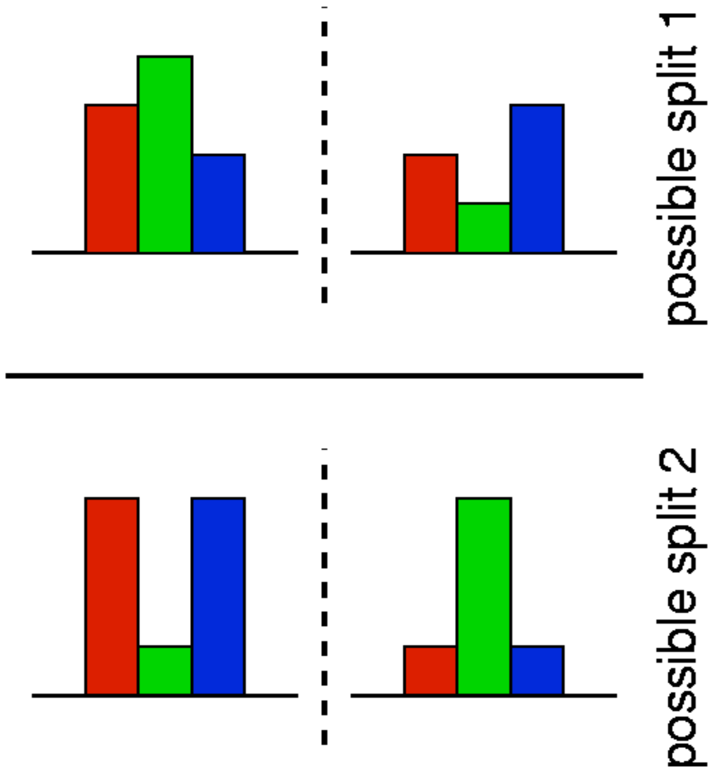
- Try each yes/no question in turn.
- Here is what happens for one question we are trying:




mainly red ?
N
Y
orange
lemon
lime
orange
lemon

Entropy measures ‘purity’

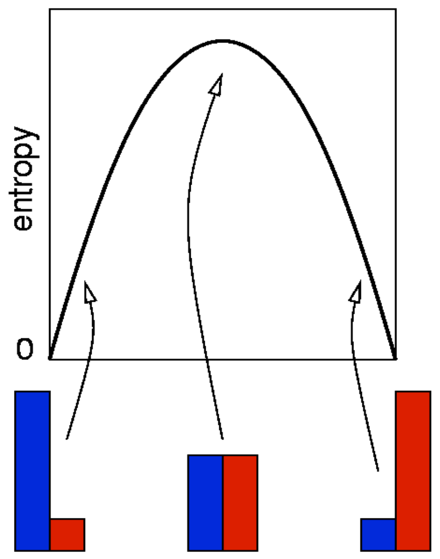
- Low entropy means highly predictable.
- Here is what happens for two different questions we are trying, which each give a different split of the data:



In the second question (lower figure), the total entropy is lower, so this is a better split of the data

Entropy more formally:

- Entropy is: $$H = -\sum_x p(x) log(p(x))$$



Entropy is zero when things are 100% predictable, e.g., everything is blue

How big should the tree grow?

- We want to stop the tree-building algorithm at some point
- Need a criterion for when to stop
  - When none of the remaining questions usefully split the data
  - Limit the depth of the tree
  - When the number of data points in a partition is too small
- Don’t simply want to continue until we run out of questions because:
  - Not all questions usefully split the data (perhaps because not all predictees are informative)
  - Can’t reliably measure goodness of split for small data partitions

When can CART be used

- When there are a number of predictors, possibly of mixed types
- When we don’t know which are the most important ones
- When some predictors might not be useful
- When we can ask yes/no questions about the predictors

Text Processing for Speech Synthesis

Prosody

Prosody and intonation

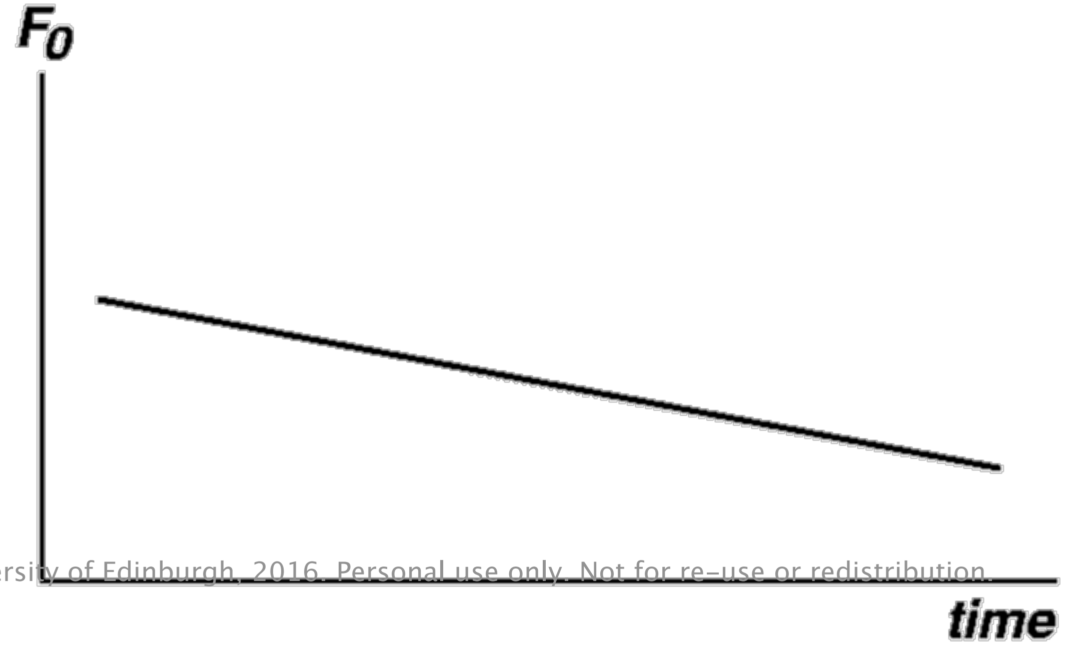
- Recap:
  - We have processed the text into tokens and then into words
  - We have determined a sequence of phonemes
- We now turn to **suprasegmental** aspects of synthesis.
- We will adopt Jurafsky's terminology, in which **prosody** has three aspects:
  - Tune - e.g. question vs. statement
  - Structure – phrasing
  - Prominence – a property of syllables, including stress and accent
- **Intonation** is usually taken to mean the tonal/melodic aspects of prosody

Acoustic correlates of prosody

- The main acoustic consequences of prosody are:
  - F0
  - Amplitude
  - Duration
- Accented syllables tend to have higher F0, longer duration, and may have increased amplitude.
- Phrase-final syllables often get lengthened.
- F0 often goes down at the end of a phrase
- Acoustic cues aid human processing of speech, e.g., lengthening signals the end of phrase
- Some acoustic cues affect meaning: e.g., questions may have rising F0 at the end of the utterance (in English, at least)

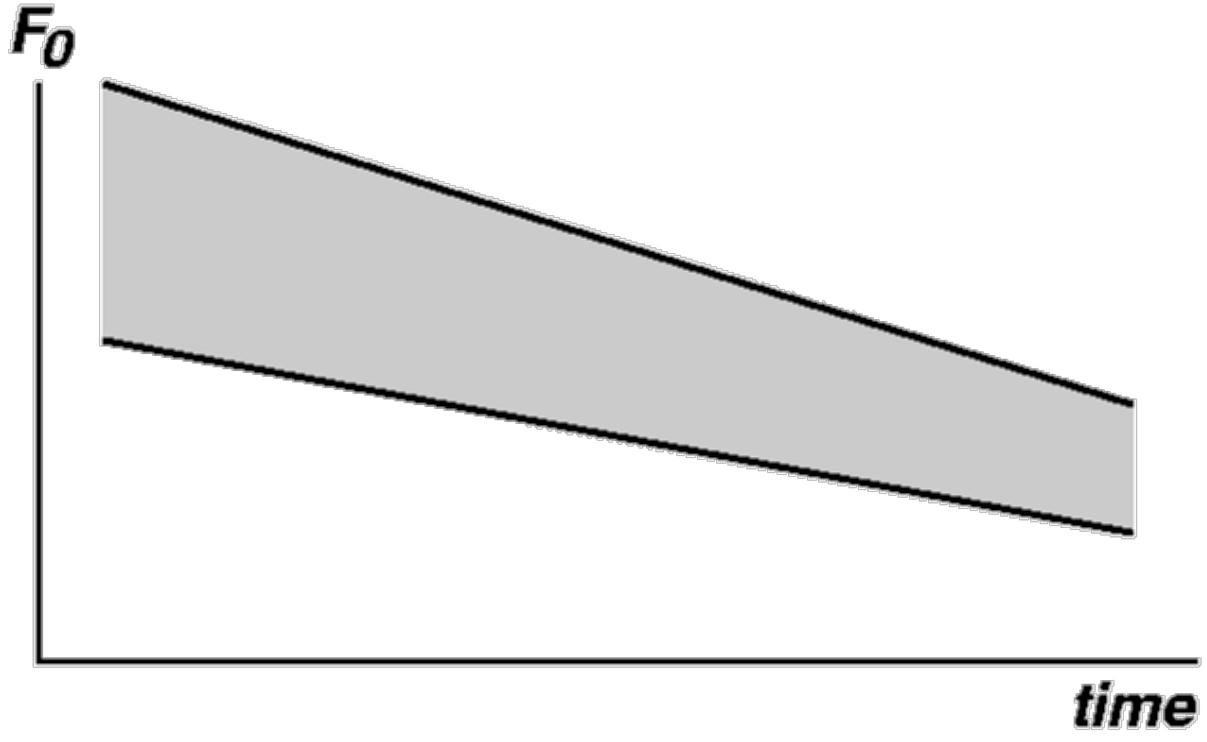
Tune

- The tune of an utterance has two components:
  - The global pitch contour shape
  - Localised pitch accents
- Most utterances show an overall downward trend in F0 called **declination**. We run out of breath: air flow & pressure decrease, vocal folds vibrate more slowly



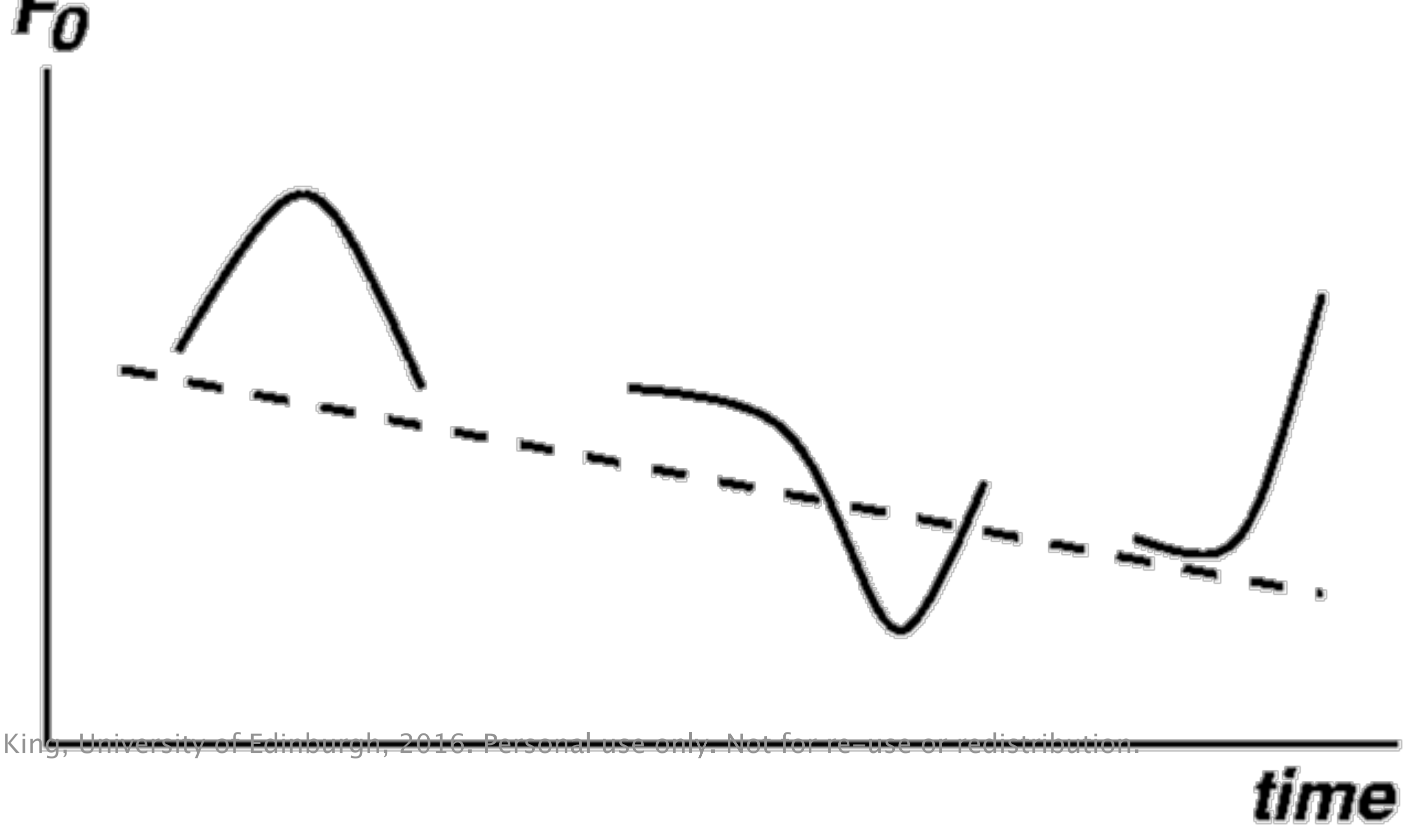
Tune: baseline and topline

- Not only does the mean value of F0 decrease with time, the range does too.



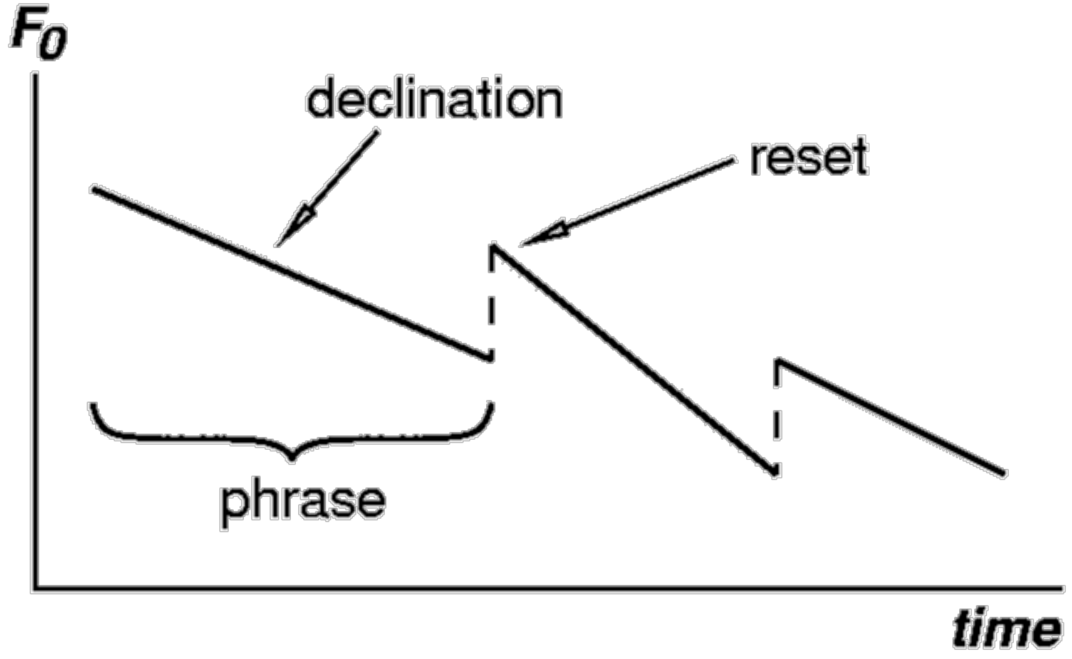
Tune: pitch accents

- Superimposed on the baseline are pitch accents.
- Pitch accents are located on (some) stressed syllables.
    - Lexical stress indicates *possible* positions for pitch accent placement



Structure

- The structure of an utterance is reflected in F0, segment durations and boundary pauses



Boundaries are marked by some combination of pauses, boundary tones, phrase-final syllable lengthening and resets in F0 declination.
Often, there is no actual pause, although listeners may perceive one.

Intonational phrases vs. syntactic phrases

- If a known-to-be-correct syntactic structure was available (e.g. from a language generation system), then we *might* use it
    - The relationship between **syntactic phrasing** and **intonational phrasing** is non-trivial, and syntactic parsing of text is error-prone
    - However, prosody is more related to the **information structure** than conventional syntactic structure: e.g., given vs. new information, contrasts, list structures,…
- Phrase length is also affected by the need to **breathe** at regular intervals - breath phrases - the locations of these are not predictable from syntax
- **So, with current technology, a traditional deep syntactic analysis (e.g., a full parse tree) may not be very useful for TTS**

Structure: intonational phrases

- What exactly constitutes an intonational phrase?
  - There is no single accepted definition - opinions vary
- For synthesis, we need to convey the phrase structure using prosody
- How do listeners perceive phrase structure?
  - Phrase boundaries
- So, instead of phrases, we think in terms of **phrase boundaries** instead
- The strength of the boundary determines perception of phrase structure

$|_{\text{major}}$ I want $|_{\text{minor}}$ to buy $|_{\text{minor}}$ the blue one $|_{\text{major}}$

- We could even say there are phrase boundaries after every word, some of which have zero strength

Prominence

- In the citation form of a word, the syllables have a particular pattern of prominence
- This is called **lexical stress** and can be specified in the lexicon

```
("vegetarian" jj (((v e) 2) ((jh i) 0) ((t eir) 1) ((r iy @ n) 0)))
("the" (dt full) (((dh ii) 0)))
("the" (dt full) (((dh ii) 1)))
("the" (dt reduced) (((dh @) 0)))
```

- But
  - Lexical stress only defines a *potential* pattern of prominence
  - In connected speech, actual syllable prominence is affected by other properties of the utterance
  - Word compounding (“chocolate cake”, “coat hook”) changes things

Prominence

- Lexical stress is marked in the lexicon. It defines a *potential* pattern of prominences, but is not the whole story
- Which syllables receive stress also depends on discourse factors like
    - Focus words and phrases
    - Given vs. new words
- plus factors in the same utterance
    - words in the context
    - positions of phrase boundaries
- Not all of these are taken into account, even in state-of-the-art systems. Factors beyond the current sentence are very rarely considered.

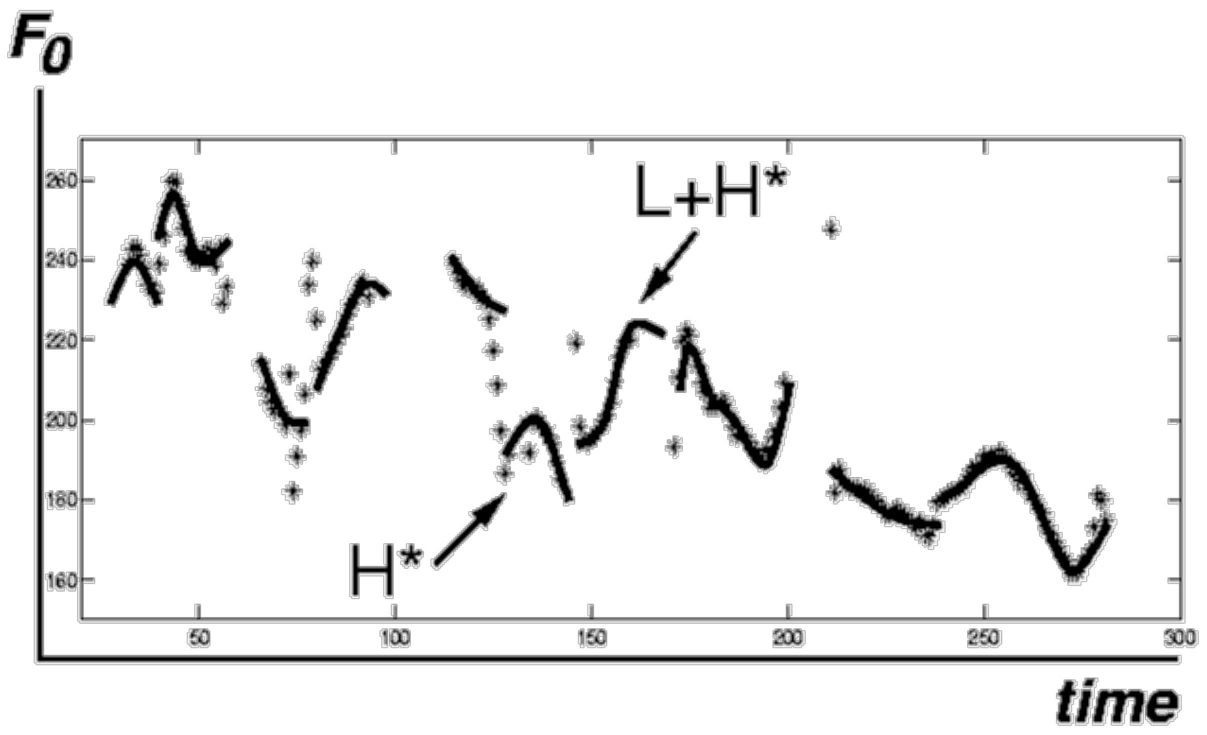
Realising prosody

- How do we realise stressed syllables?
  - by manipulating amplitude, duration and F0
- Need a representation for accents and boundaries
  - a popular choice is **ToBI** (tones and break indices); this is a symbolic system
- Then we need a (statistical) model which can predict this from the available predictors
  - We’ll use a model learned from data
  - That means we need labelled data
  - Which implies **hand-labelling** speech with Tobl, phrase breaks, etc

ToBI

- Two basic tones H (high) and L (low)
  - these combine to make rise and fall patterns: L+H, H+L
- Use
  - * to mark alignment with the stressed syllable
  - % to mark a boundary tone
- Final accent inventory is: L*, H*, L*+H, L+H*, H+L*
- Final boundary tone inventory is: L%, H%
- Additionally, every word is given a break index of 0-4, to mark the strength of the following boundary (0 means “no boundary”)

Annotate training data using ToBI



ToBI provides a stylised symbolic representation suitable for hand-annotation of data, and for computation

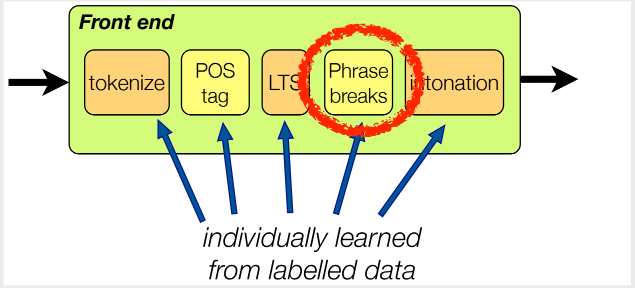
ToBI transcription

- ToBI provides a complete system for transcribing F0 contours, in much the same way as a phoneme set allows transcription of the acoustic signal
  - it is thus possible to manually label fairly large amounts of data
  - which provides the opportunity to train statistical models
- *Notes*
  - *this is a simplified description of ToBI*
  - *ToBI is far from perfect*
  - *ToBI is not the only system available, there are non-symbolic (i.e., parametric) systems too*
  - *ToBI is specific to English, but variants exist for some other languages*

Synthesising Prosody

- What have we got so far?
    - acoustic correlates of prosody are known
    - symbolic representation of pitch accents, boundary tones and boundary strengths is available (e.g., ToBI)
    - a set of utterances with manually labelled pitch contours
- So, automatic synthesis of prosody is now possible
- Note:
    - incorrect accents (wrong place, wrong type, wrong size) are more noticeable than missing accents to listeners; they result in a low overall listener rating
    - therefore, most synthesisers play it safe and aim for fairly neutral prosody

Example process 3



- Phrase-break prediction
  - binary classifier using sequence as input
- But
  - trained on annota
    spoken data
  - therefore very sm
    training set

This sequence is the annotated training data for our phrase break predictor

DT	NB
CD	NB
CD	NB
NN	NB
Break !	
JJ	NB
NN	B

Automatic phrase boundary prediction

- Task: predict boundary position and strength
- Equivalent task: predict a boundary strength after every word (some are zero)
- **Predictee**
  - *break strength*
    - Festival uses just 3 boundary strengths (instead of ToBI’s 5): Major (BB [big break]), Minor (B [break]), No break (NB)
- **Predictors**
  - *contextual features* of current and neighbouring syllables (similar to intonational event prediction - see next slide)
- Models
  - CART
  - Markov model with N-gram

Automatic intonation event prediction: placement

- Step 1: placement
    - **Predictee**
        - *placement* (whether a syllable receives an accent)
    - **Predictors**
        - Syllable position in phrase
        - Syllable context
        - Lexical stress
        - Lexical stress of neighbours
        - Break strength of this word and neighbouring words
        - POS tags

Automatic intonation event prediction: type

- Step 2: type
  - **Predictee**
    - *accent type*
      - For ToBI, one of: L*, H*, L*+H, L+H*, H+L* or a boundary tone L% or H% (using a CART as a **classification tree**)
      - In parametric models, a parameterised representation of accent height, duration, etc. (using a CART as a **regression tree**)
  - **Predictors**
    - again, a number of factors relating to the syllable in question and its context

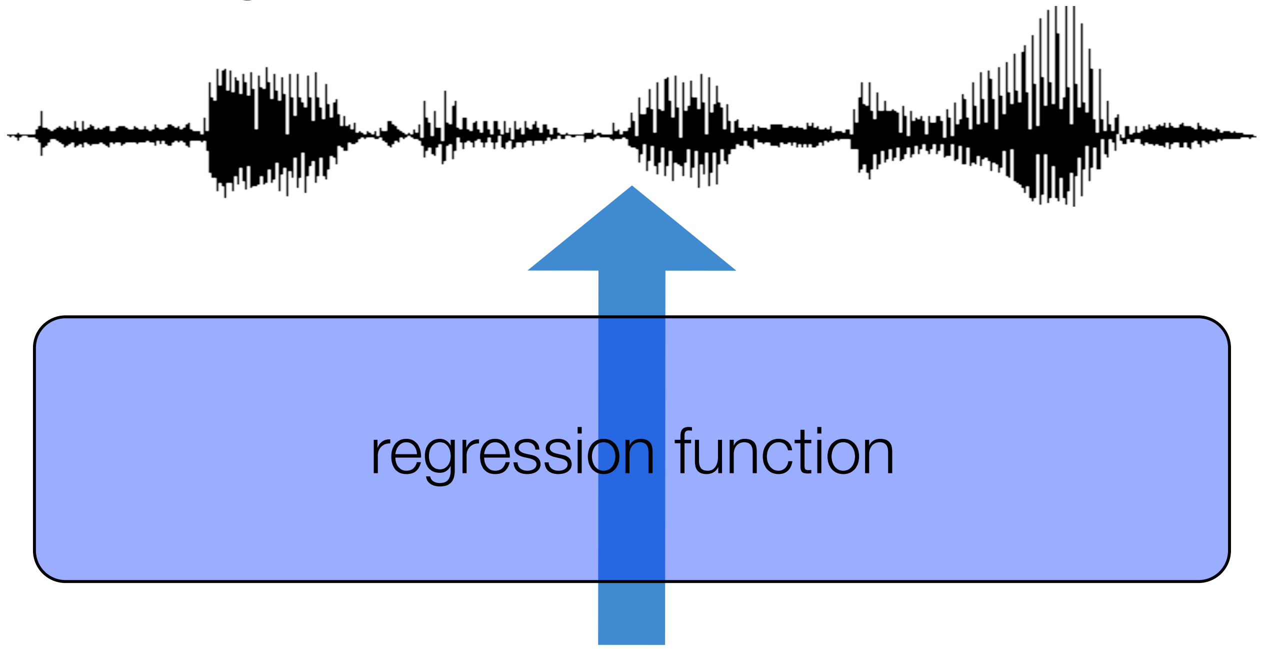
Automatic intonation event prediction: realisation

- Step 3: realisation
    - The ToBI symbol must now be realised as actual F0 values.
    - Typically predict F0 at 3 points per syllable
    - It will not come as surprise that this prediction too can be done using a model trained on data
        - We're now predicting continuous values
        - Use a CART : this time as a **regression** tree

Text Processing for Speech Synthesis

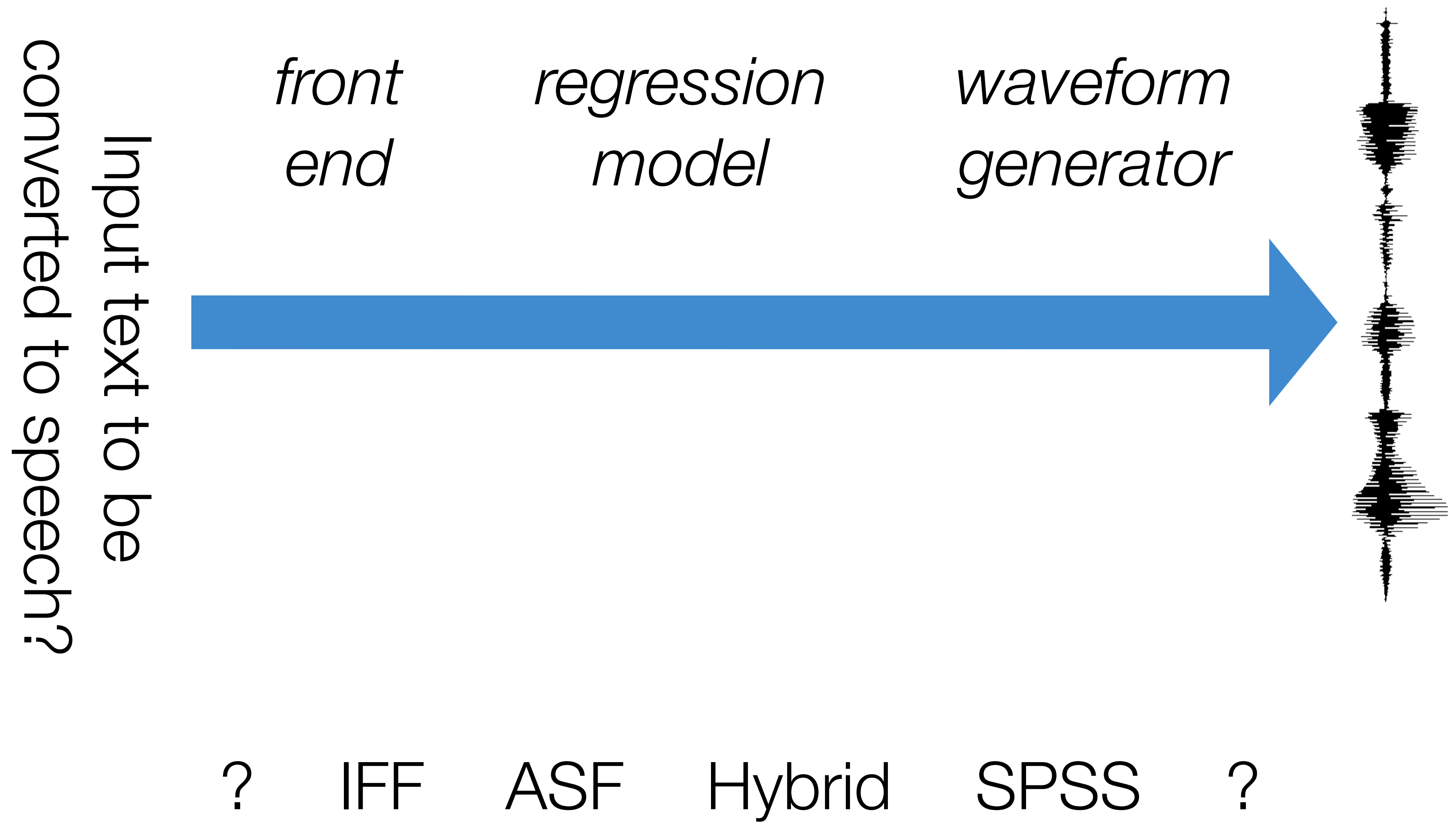
Annotating the database with the linguistic features

Looking ahead: speech synthesis as a regression problem



Input text to be
converted to speech?

How much of the regression does each stage in the full TTS pipeline do?

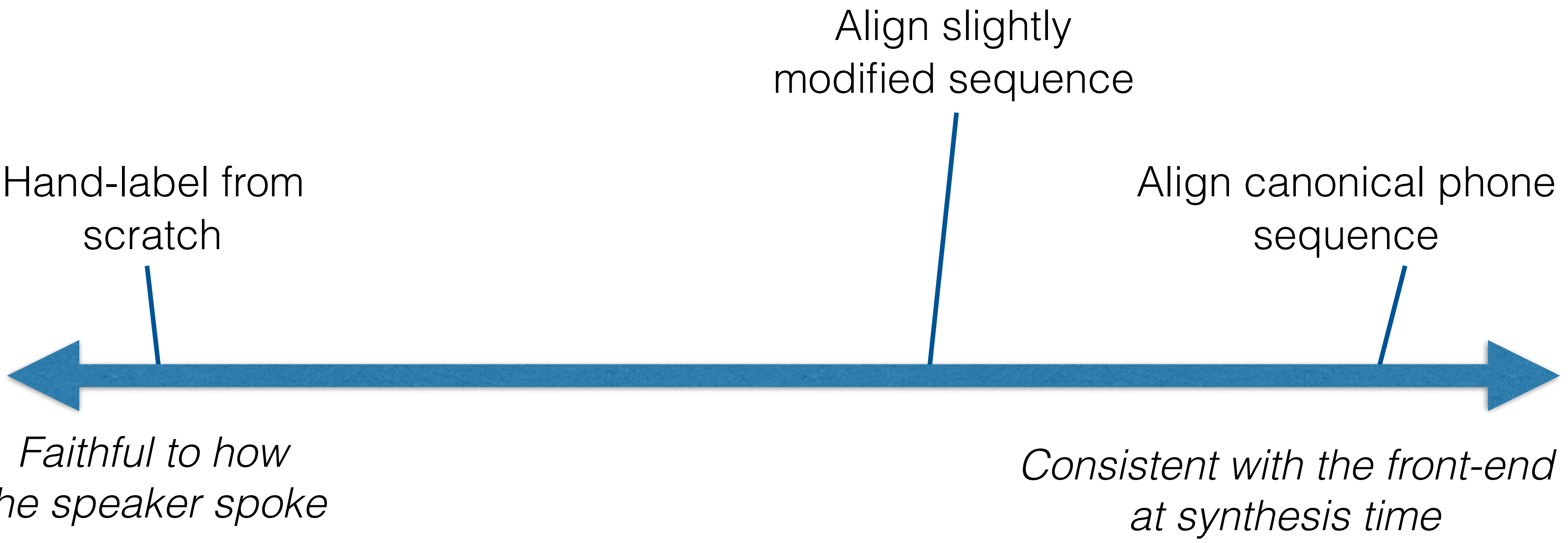


What are the annotations on the database?

- front-end for classical unit selection does **not** predict **every speech parameter**
- many systems predict **some speech parameters** - it's a matter of degree:
    - **highly-explicit** approach: e.g., predict target unit durations and include that in the target cost
    - **typical** approach (e.g., Festival's *multisyn*): predict reliable information from the text, including POS, phrase breaks, phoneme string that includes some connected speech effects (vowel reduction, linking-r)
    - **more implicit** approach: rely more on text-based features, minimal prediction beyond that (limitation: have to learn from limited data)
        - in **unit selection**: assume selected speech will contain the necessary effects
        - in **statistical parametric systems**: learn probabilistic relationship between those features and acoustic properties

Labelling the database

- Why not hand-label?
    - cost & consistency


Align slightly modified sequence
Hand-label from scratch
Align canonical phone sequence
Faithful to how the speaker spoke
Consistent with the front-end at synthesis time

Forced alignment

- Standard technique from Automatic Speech Recognition
- The same as normal automatic speech recognition, except
    - highly constrained language model
    - we record the model- (or even state-) level alignment during decoding

Performing forced alignment

- Let’s assume we have a fully-trained set of acoustic models
- Language model
  - constructed from the known word sequence for the current sentence
  - i.e., language model is different for each sentence
- Pronunciation model
  - the same dictionary we will use for synthesis
  - can include pronunciation variation
  - plus optional rule-based variations
    - vowel reduction
    - short silence insertion between words

The language model - simplest version

```
there    was    a    change    now
```

Where do the acoustic models come from?

- Could borrow then from an existing ASR system
- Tend to get better results with simpler, but speaker-dependent models
  - trained on the speech database
- Hang on: training on the “test data”? Isn’t that cheating?
  - no - it’s not “test data” !

- Ingredients needed to train speaker-dependent phone models
  - speech data, parameterised as MFCCs
  - phone-level transcription of that data

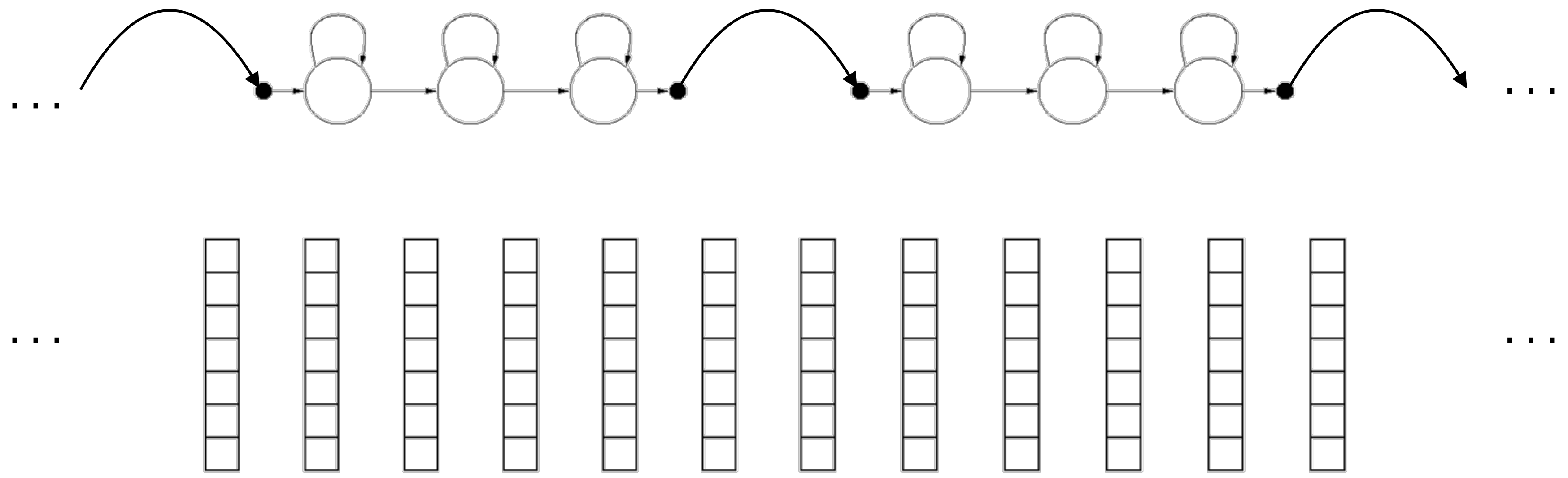
“Flat start” training

- In Speech Processing, we considered training whole word models on data where the word (i.e., model) boundaries were known

- ***Recap:***
    - ***Viterbi training (slides from Speech Processing), then generalisation to Baum-Welch***

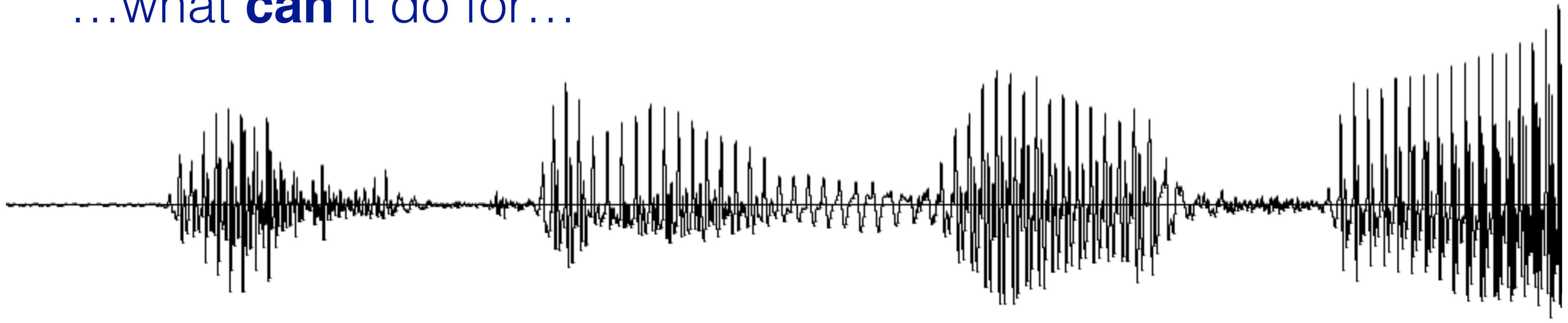
“Flat start” training

- Note how we **did not need a state-level alignment**
    - the training procedure iteratively refines that
- Generalise this to **not needing a phone- or word-level alignment** (still need sentence-level alignment, or possibly a larger unit)



Labelling vowel reduction

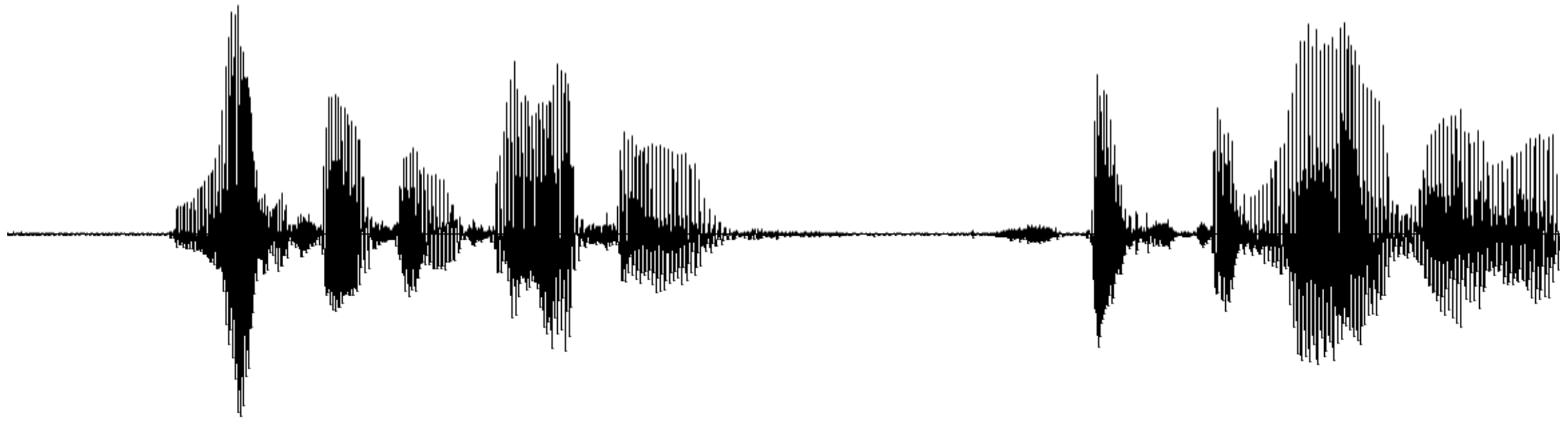
"…what **can** it do for…"



k ae n

ax

Labelling between-word short silses



The language model - with vowel reduction and optional between-word short silses

there   was   a   change   now

sil   dh eh r   w aa z   ax   ch ey n jh   n aw   sil

ax   sp   ax   sp   sp   ax   sp   ax

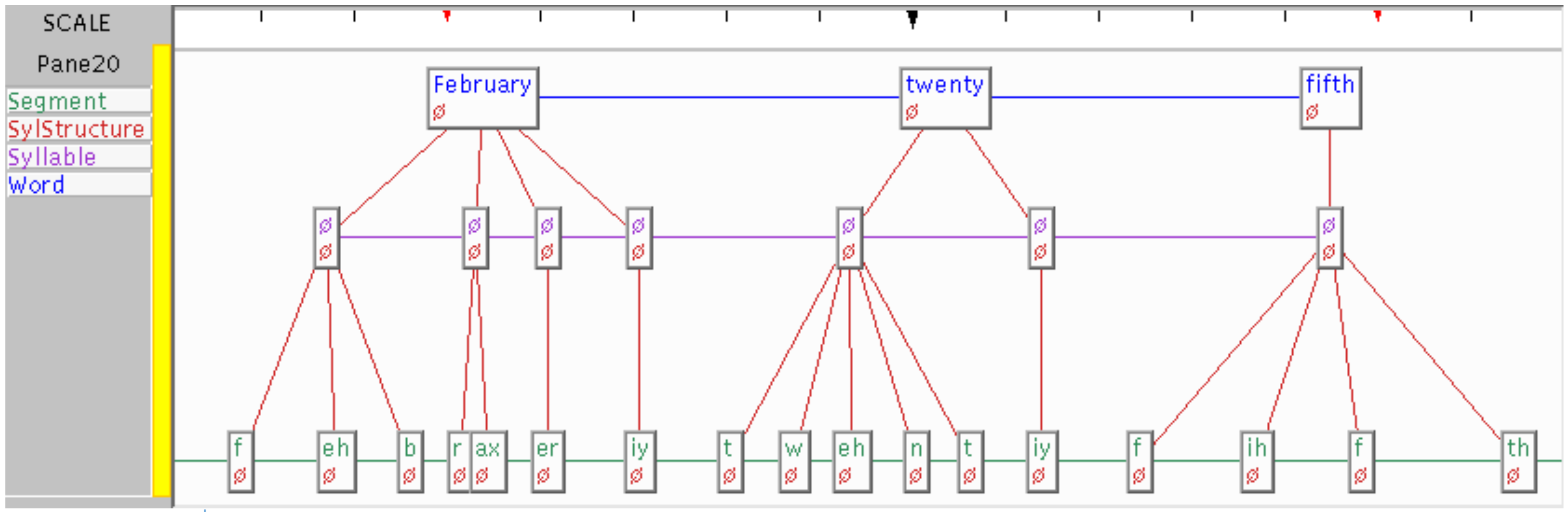
Combining segmentation with supra-segmental structure

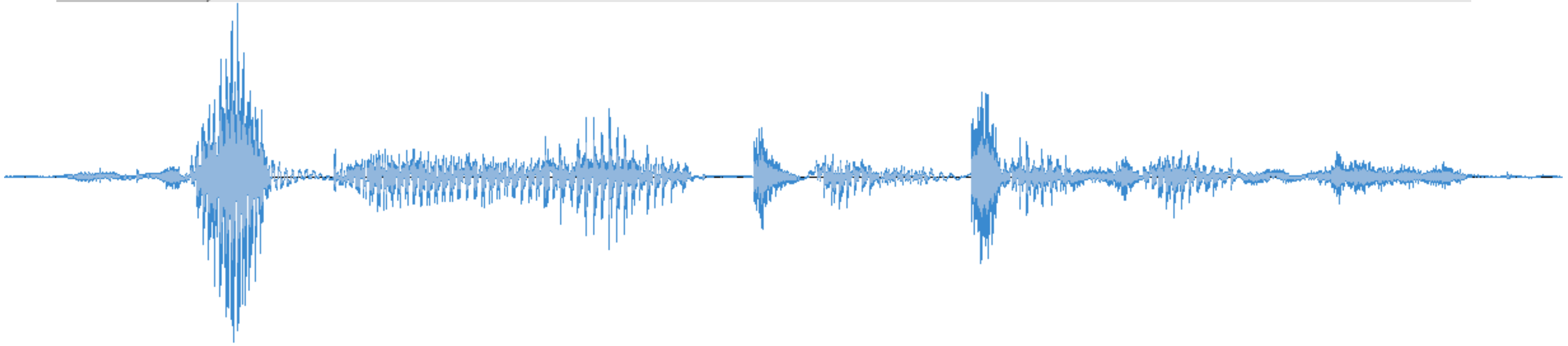
```
     there     was     a     change     now

sil  dh eh r   w aa z   ax   ch ey n jh   n aw  sil

       ax  sp   ax   sp   sp   ax      sp   ax
```

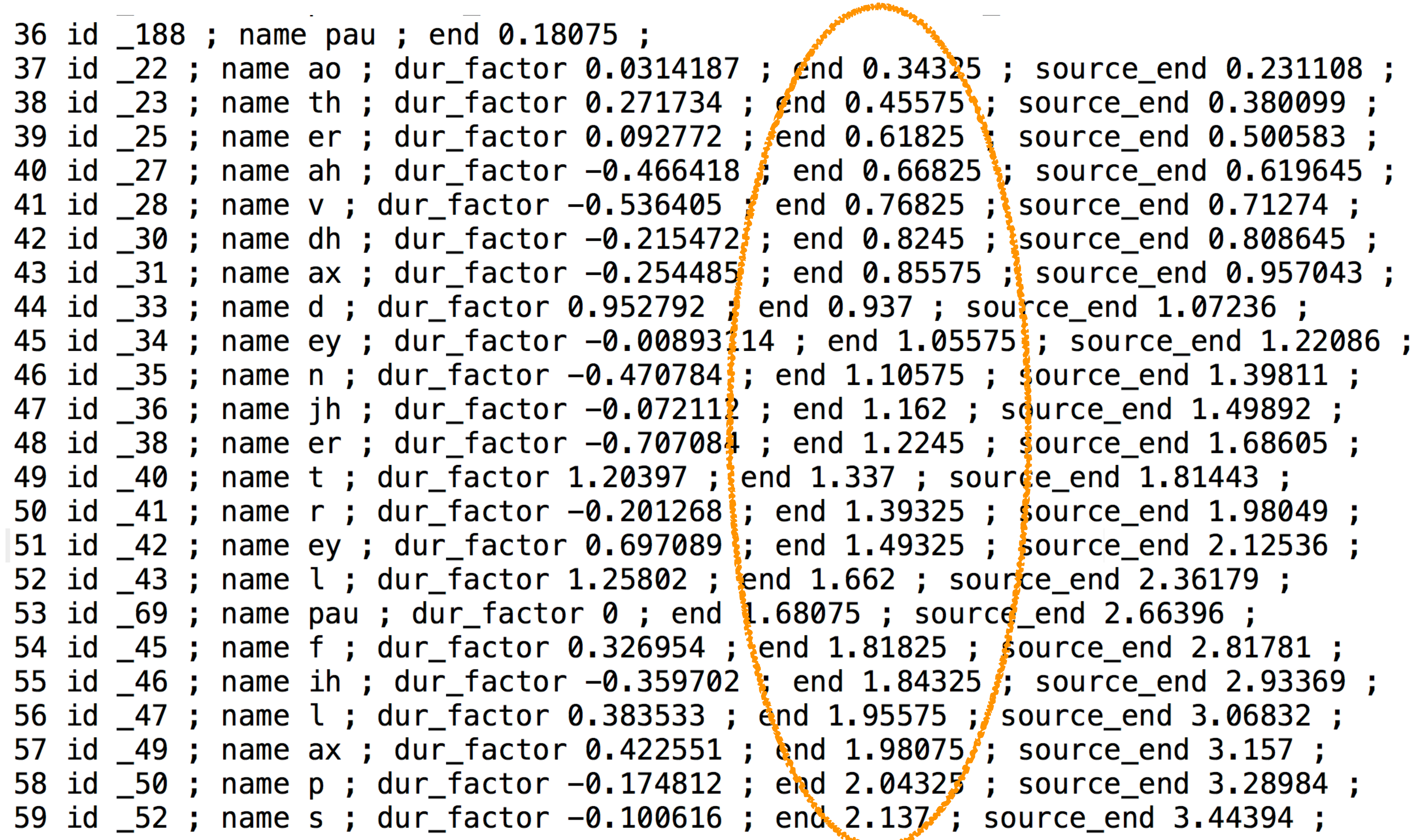
Combining segmentation with supra-segmental structure





sil f eh b r ax er iy sp t w eh n t iy f ih f yh sil

Combining segmentation with supra-segmental structure: add segment end times to linguistic specification



```
id _188 ; name pau ; end 0.18075 ;
id _22 ; name ao ; dur_factor 0.0314187 ; end 0.34325 ; source_end 0.231108 ;
id _23 ; name th ; dur_factor 0.271734 ; end 0.45575 ; source_end 0.380099 ;
id _25 ; name er ; dur_factor 0.092772 ; end 0.61825 ; source_end 0.500583 ;
id _27 ; name ah ; dur_factor -0.466418 ; end 0.66825 ; source_end 0.619645 ;
id _28 ; name v ; dur_factor -0.536405 ; end 0.76825 ; source_end 0.71274 ;
id _30 ; name dh ; dur_factor -0.215472 ; end 0.8245 ; source_end 0.808645 ;
id _31 ; name ax ; dur_factor -0.254485 ; end 0.85575 ; source_end 0.957043 ;
id _33 ; name d ; dur_factor 0.952792 ; end 0.937 ; source_end 1.07236 ;
id _34 ; name ey ; dur_factor -0.00893114 ; end 1.05575 ; source_end 1.22086 ;
id _35 ; name n ; dur_factor -0.470784 ; end 1.10575 ; source_end 1.39811 ;
id _36 ; name jh ; dur_factor -0.072112 ; end 1.162 ; source_end 1.49892 ;
id _38 ; name er ; dur_factor -0.707084 ; end 1.2245 ; source_end 1.68605 ;
id _40 ; name t ; dur_factor 1.20397 ; end 1.337 ; source_end 1.81443 ;
id _41 ; name r ; dur_factor -0.201268 ; end 1.39325 ; source_end 1.98049 ;
id _42 ; name ey ; dur_factor 0.697089 ; end 1.49325 ; source_end 2.12536 ;
id _43 ; name l ; dur_factor 1.25802 ; end 1.662 ; source_end 2.36179 ;
id _69 ; name pau ; dur_factor 0 ; end 1.68075 ; source_end 2.66396 ;
id _45 ; name f ; dur_factor 0.326954 ; end 1.81825 ; source_end 2.81781 ;
id _46 ; name ih ; dur_factor -0.359702 ; end 1.84325 ; source_end 2.93369 ;
id _47 ; name l ; dur_factor 0.383533 ; end 1.95575 ; source_end 3.06832 ;
id _49 ; name ax ; dur_factor 0.422551 ; end 1.98075 ; source_end 3.157 ;
id _50 ; name p ; dur_factor -0.174812 ; end 2.04325 ; source_end 3.28984 ;
id _52 ; name s ; dur_factor -0.100616 ; end 2.137 ; source_end 3.44394 ;
```

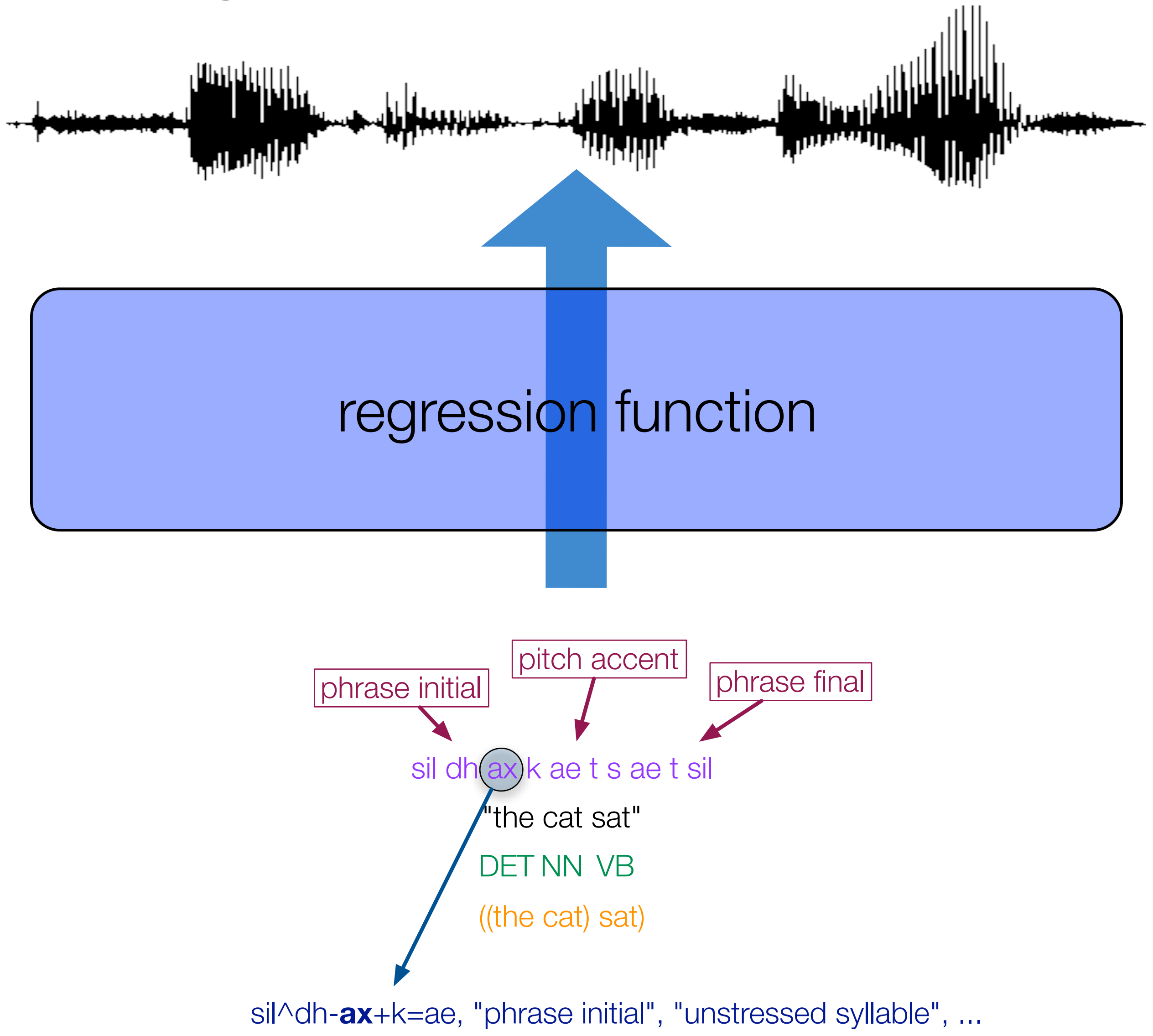
Text Processing for Speech Synthesis

If we don’t have

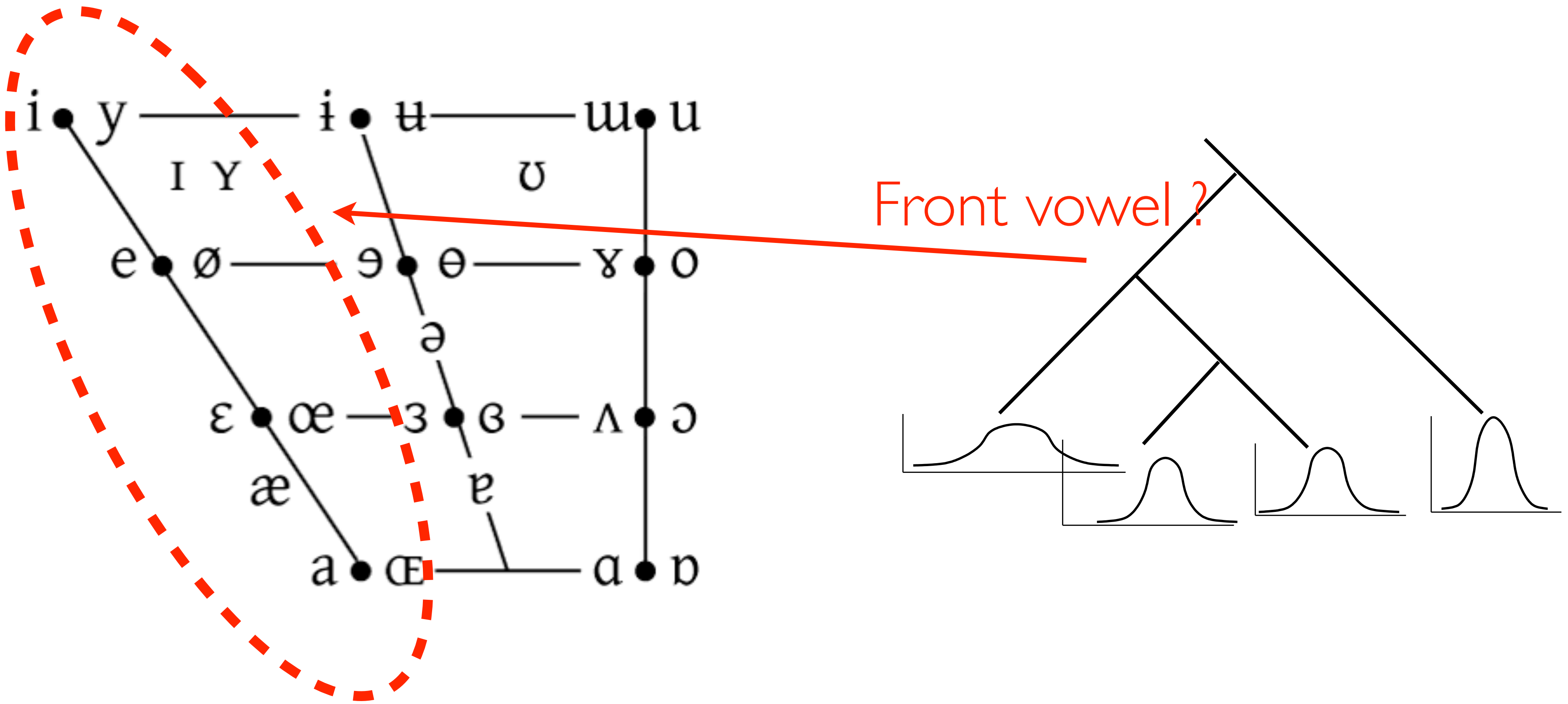
- linguistic resources (dictionary, POS tagger, …)
- annotated data (phrase breaks, intonation,…)
- detailed knowledge of the language (letter-to-sound, lexical stress)

then what can we do?

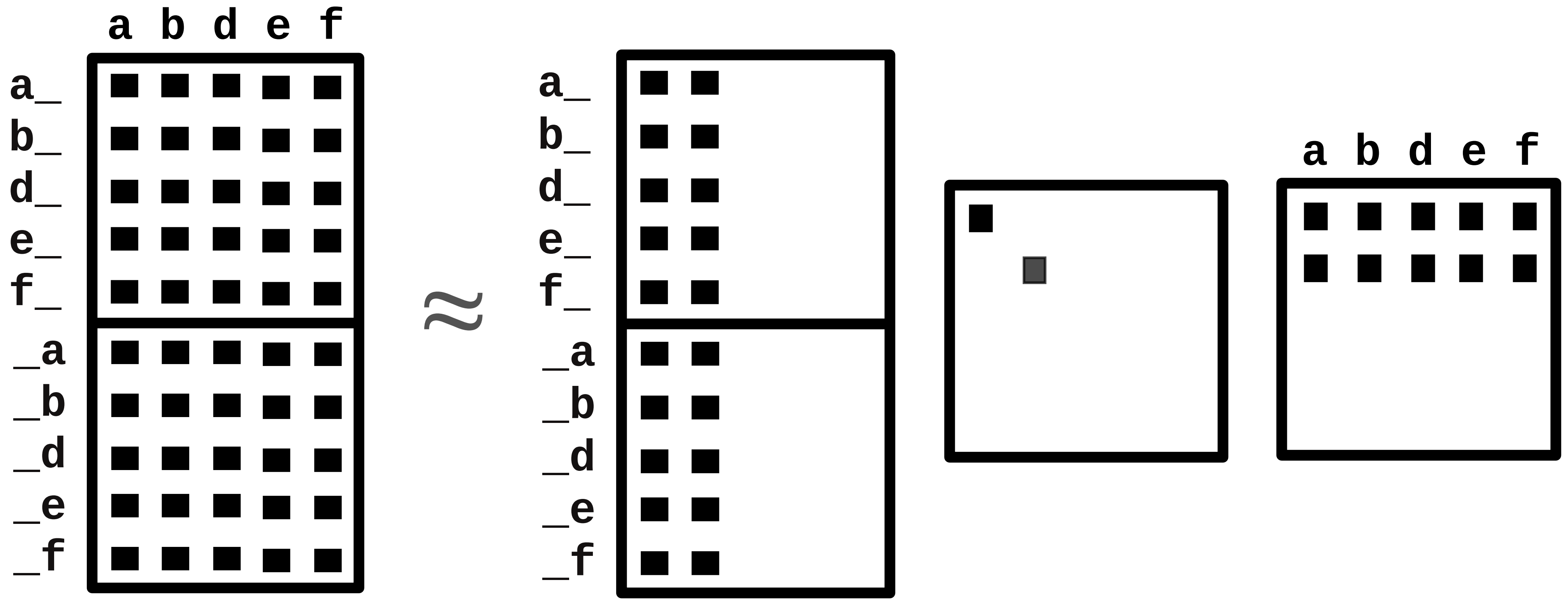
Looking ahead: speech synthesis as a regression problem



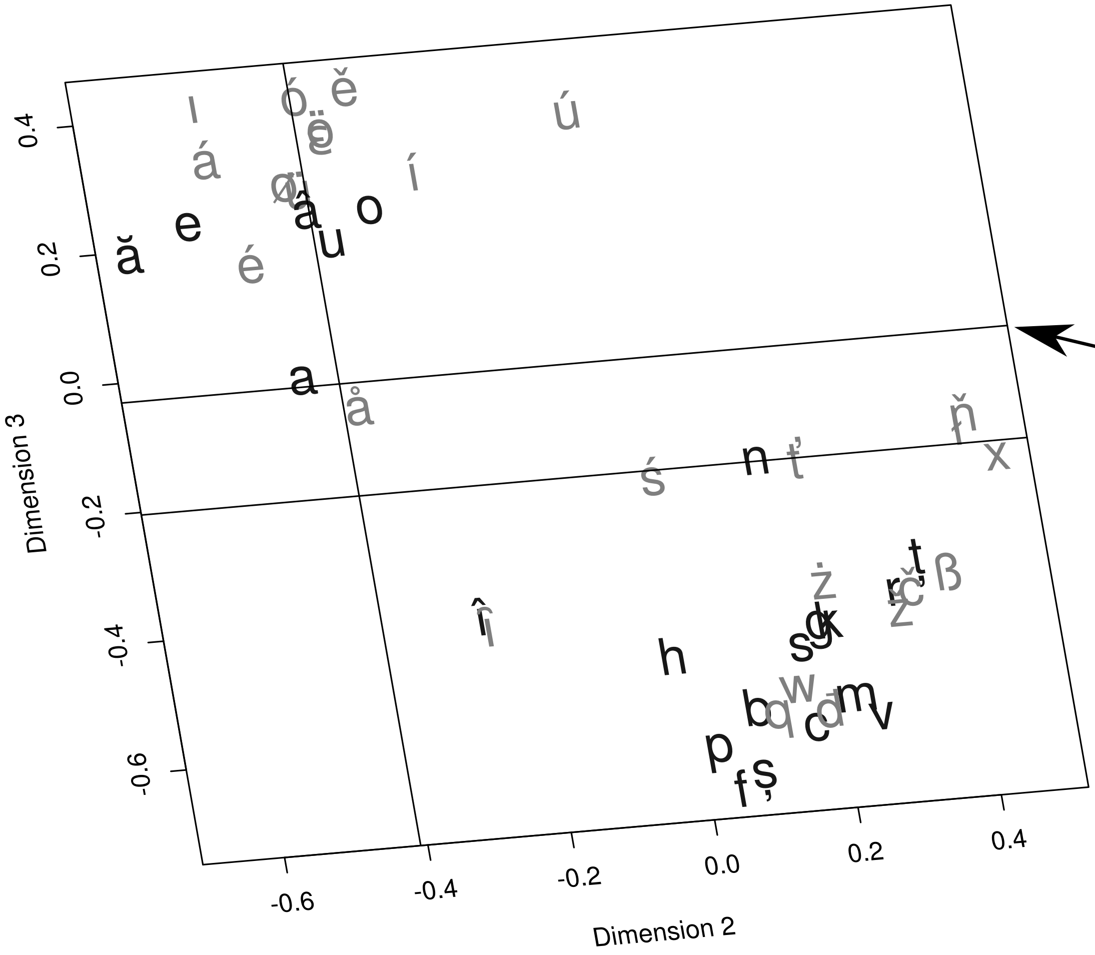
Replacing the phoneme set

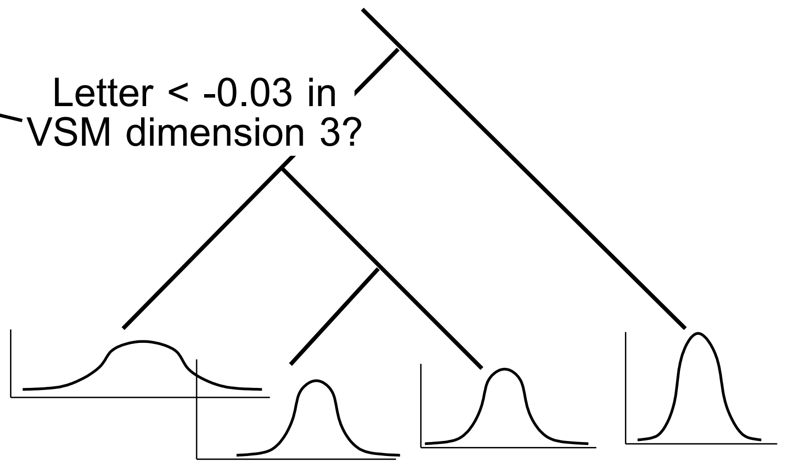


Replacing the phoneme set

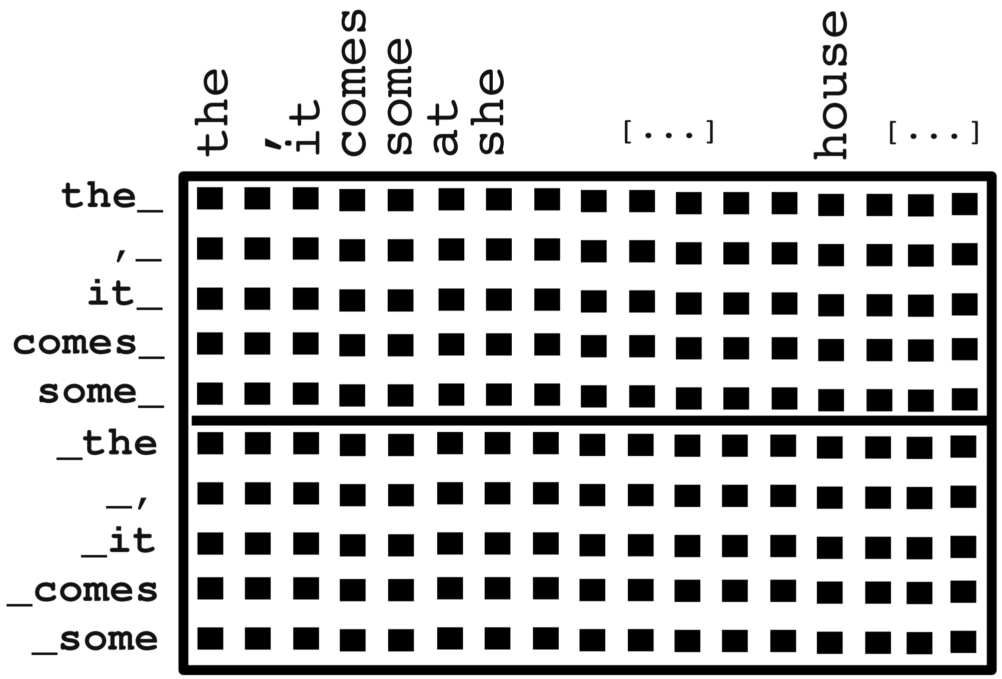


Replacing the phoneme set





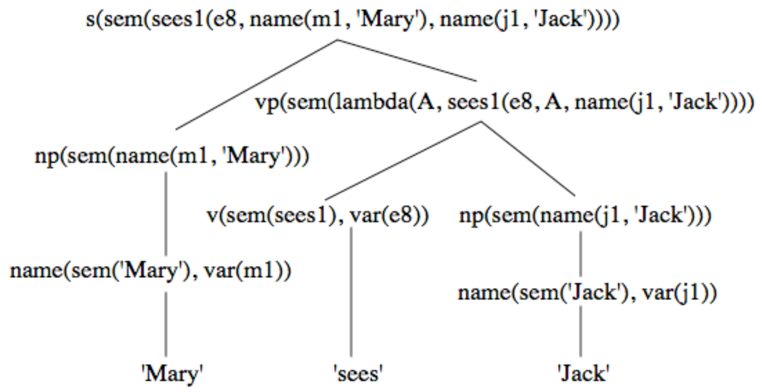
Replacing part-of-speech tags



Open research questions
1. Naive vs. knowledge-rich features

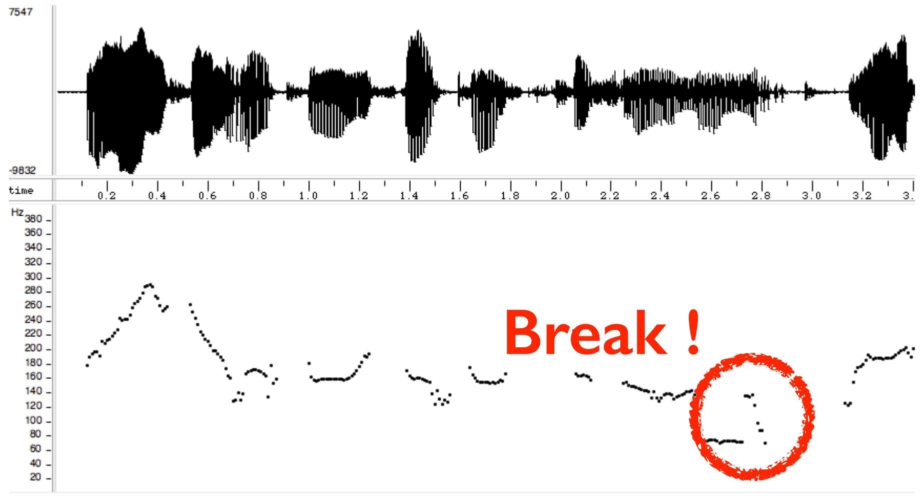
- some shallow features are very powerful
  - e.g., punctuation predicts phrase breaks with high precision
- how to combine features at many different "depths"?

```
Since 1968, voter
initiated
propositions...
```


s(sem(sees1(e8, name(m1, 'Mary'), name(j1, 'Jack'))))
vp(sem(lambda(A, sees1(e8, A, name(j1, 'Jack'))))
np(sem(name(m1, 'Mary')))
v(sem(sees1), var(e8))
np(sem(name(j1, 'Jack')))
name(sem('Mary'), var(m1))
name(sem('Jack'), var(j1))
'Mary'
'sees'
'Jack'

Tree credit: James F. Allen

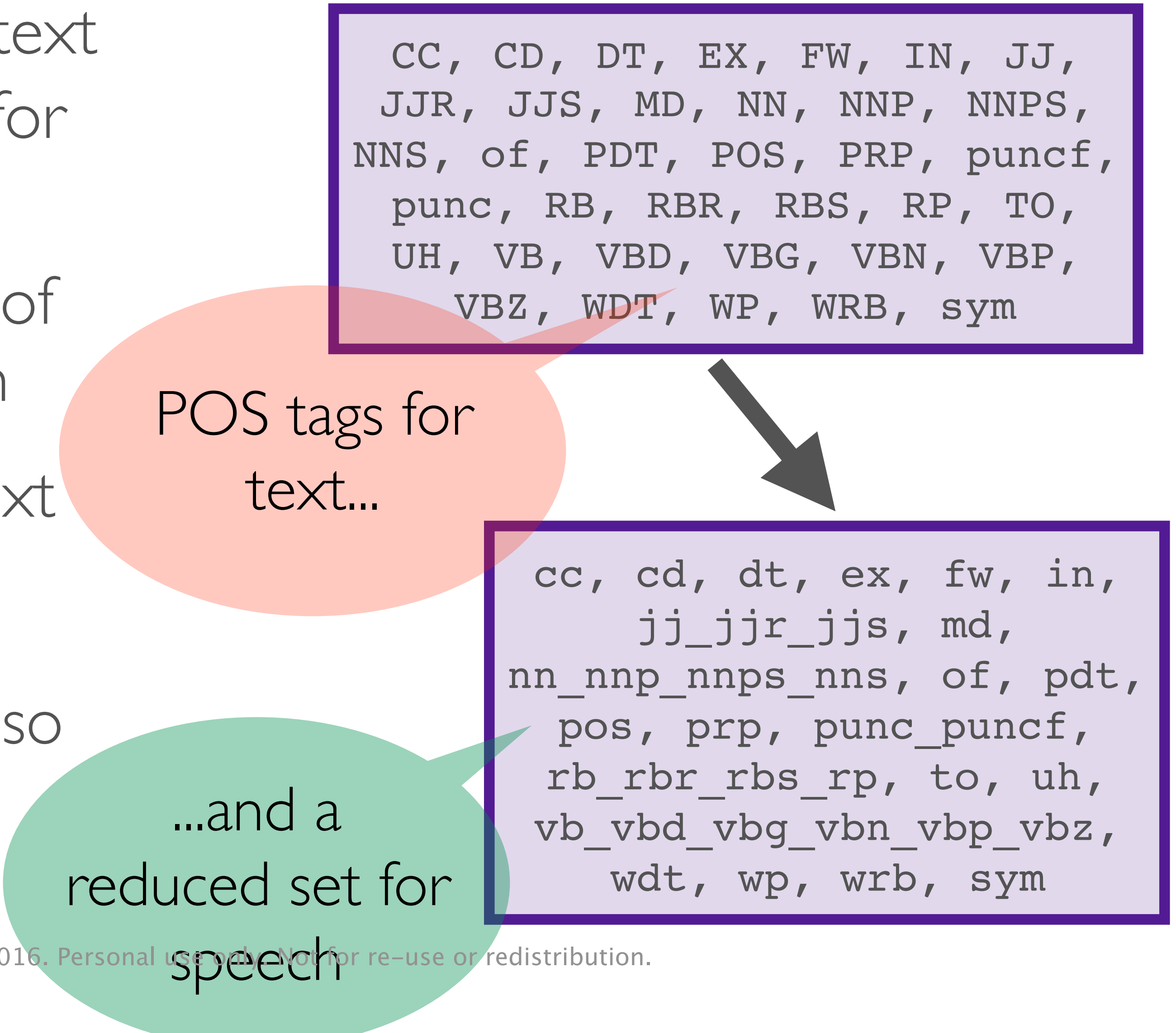
Open research questions
2. Make use of partially-labelled data



- labelled data is highly **informative**
  - but limited in **quantity** so cannot use it directly
- can we use it to infer context from **unlabelled** data ?

Open research questions
3. Discover linguistic categories

- Categories defined for text unlikely to be **optimal** for speech
- Joint discovery / design of categories that are both
  - **predictable** from text
  - **relevant** to speech
- (Not just categories - also continuous features)



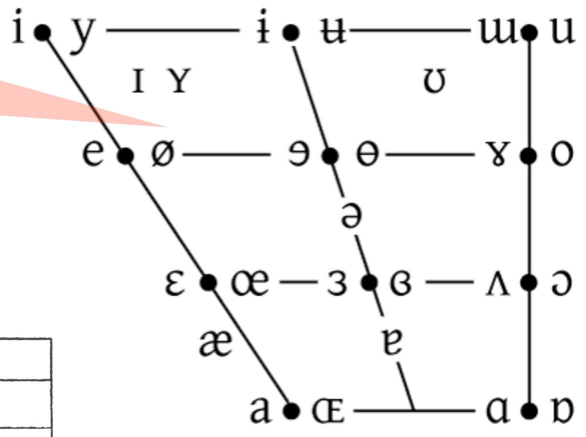
```
CC, CD, DT, EX, FW, IN, JJ,
JJR, JJS, MD, NN, NNP, NNPS,
NNS, of, PDT, POS, PRP, puncf,
punc, RB, RBR, RBS, RP, TO,
UH, VB, VBD, VBG, VBN, VBP,
VBZ, WDT, WP, WRB, sym
```

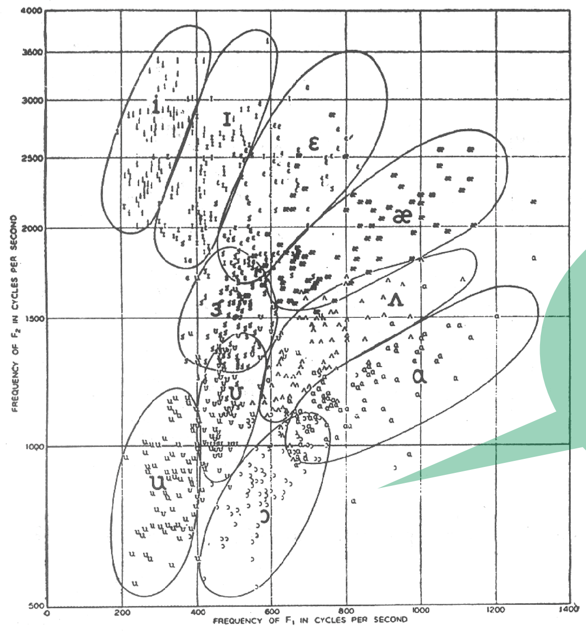
```
cc, cd, dt, ex, fw, in,
jj_jjr_jjs, md,
nn_nnp_nnps_nns, of, pdt,
pos, prp, punc_puncf,
rb_rbr_rbs_rp, to, uh,
vb_vbd_vbg_vbn_vbp_vbz,
wdt, wp, wrb, sym
```

Open research questions
4. Combine prior knowledge with data

- we do **know** some very useful things about spoken language
- yet this prior knowledge is still **incomplete**
- how to combine it with things **learned** from data ?

Trust the prior knowledge...





...or believe the data ?

Formant scatter plot credit: Keith Johnson