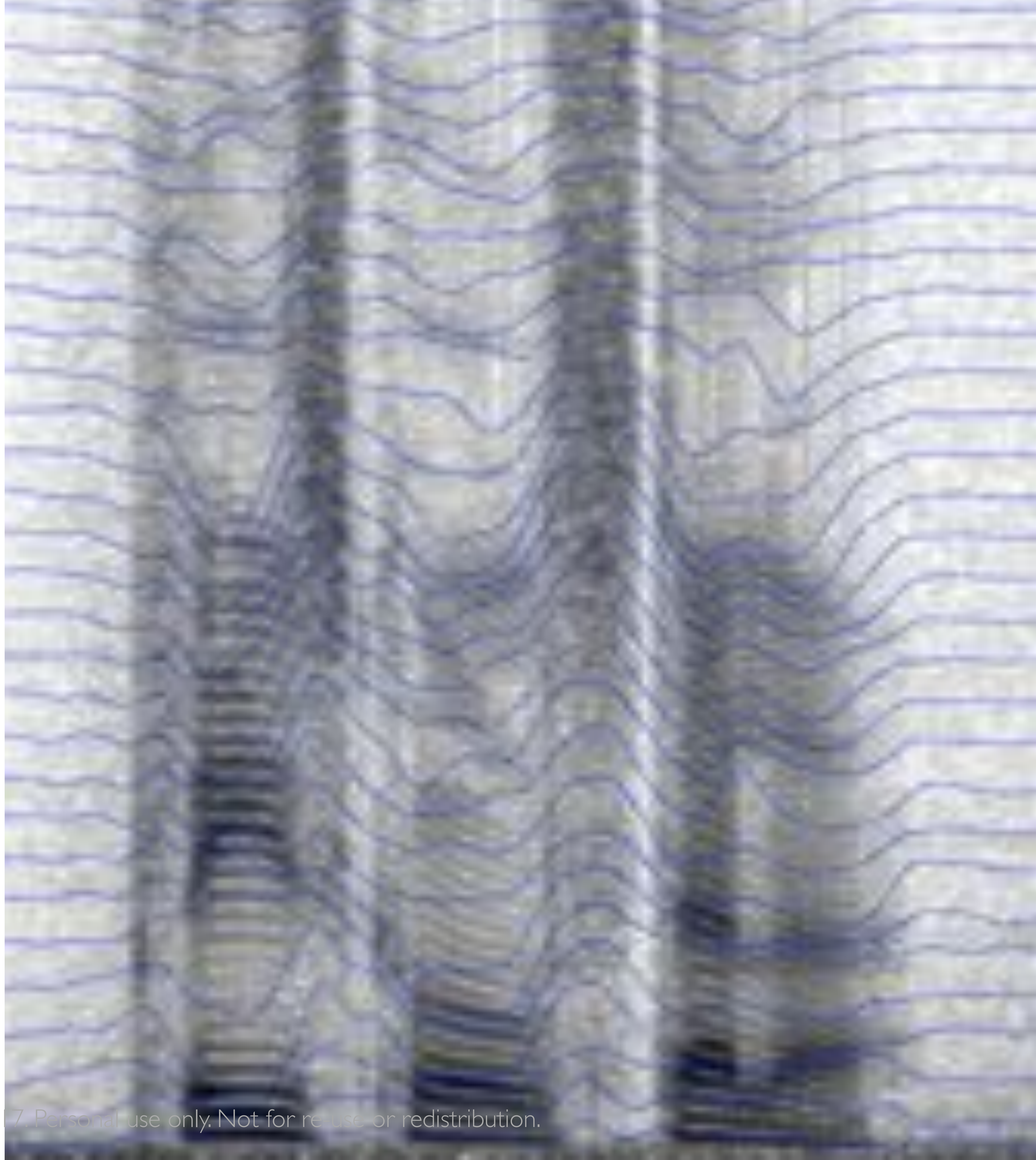
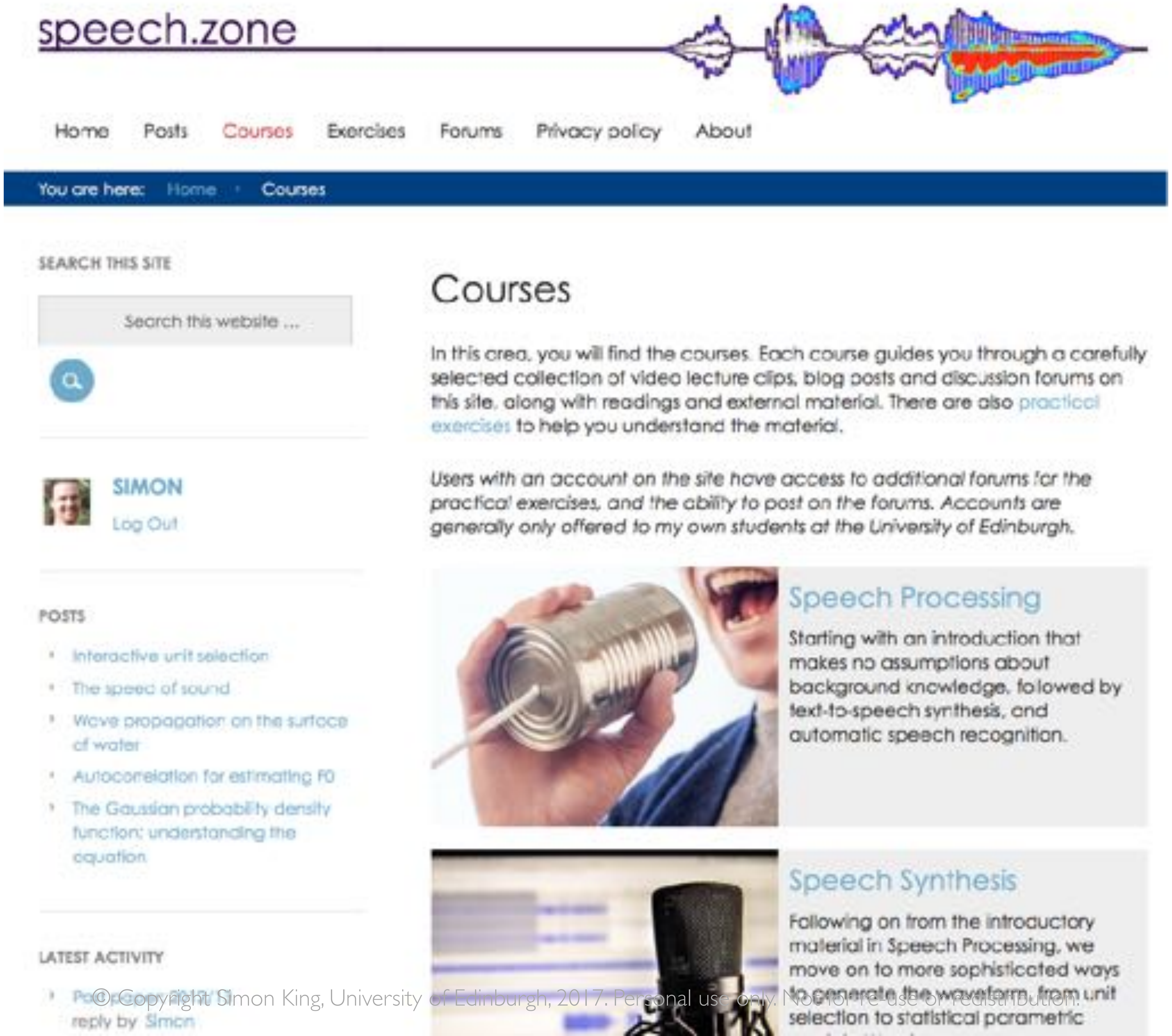
Statistical Parametric Speech Synthesis

- HMM-based (2 hours)
- DNN-based (1 hour)

Simon King
University of Edinburgh




speech.zone

Home | Posts | Courses | Exercises | Forums | Privacy policy | About

You are here: Home › Courses

SEARCH THIS SITE

Search this website …

SIMON

Log Out

POSTS

- Interactive unit selection
- The speed of sound
- Wave propagation on the surface of water
- Autocorrelation for estimating F0
- The Gaussian probability density function: understanding the equation

LATEST ACTIVITY

- reply by Simon

Courses

In this area, you will find the courses. Each course guides you through a carefully selected collection of video lecture clips, blog posts and discussion forums on this site, along with readings and external material. There are also practical exercises to help you understand the material.

Users with an account on the site have access to additional forums for the practical exercises, and the ability to post on the forums. Accounts are generally only offered to my own students at the University of Edinburgh.

Speech Processing

Starting with an introduction that makes no assumptions about background knowledge, followed by text-to-speech synthesis, and automatic speech recognition.

Speech Synthesis

Following on from the introductory material in Speech Processing, we move on to more sophisticated ways to generate the waveform, from unit selection to statistical parametric

Contents

- The big picture
  - text-to-speech, viewed as regression
- Getting ready for regression
  - feature extraction from text
  - feature extraction from speech
- Doing regression
  - using a decision tree: so-called "HMM-based TTS"
  - using more powerful and general regression models: neural networks

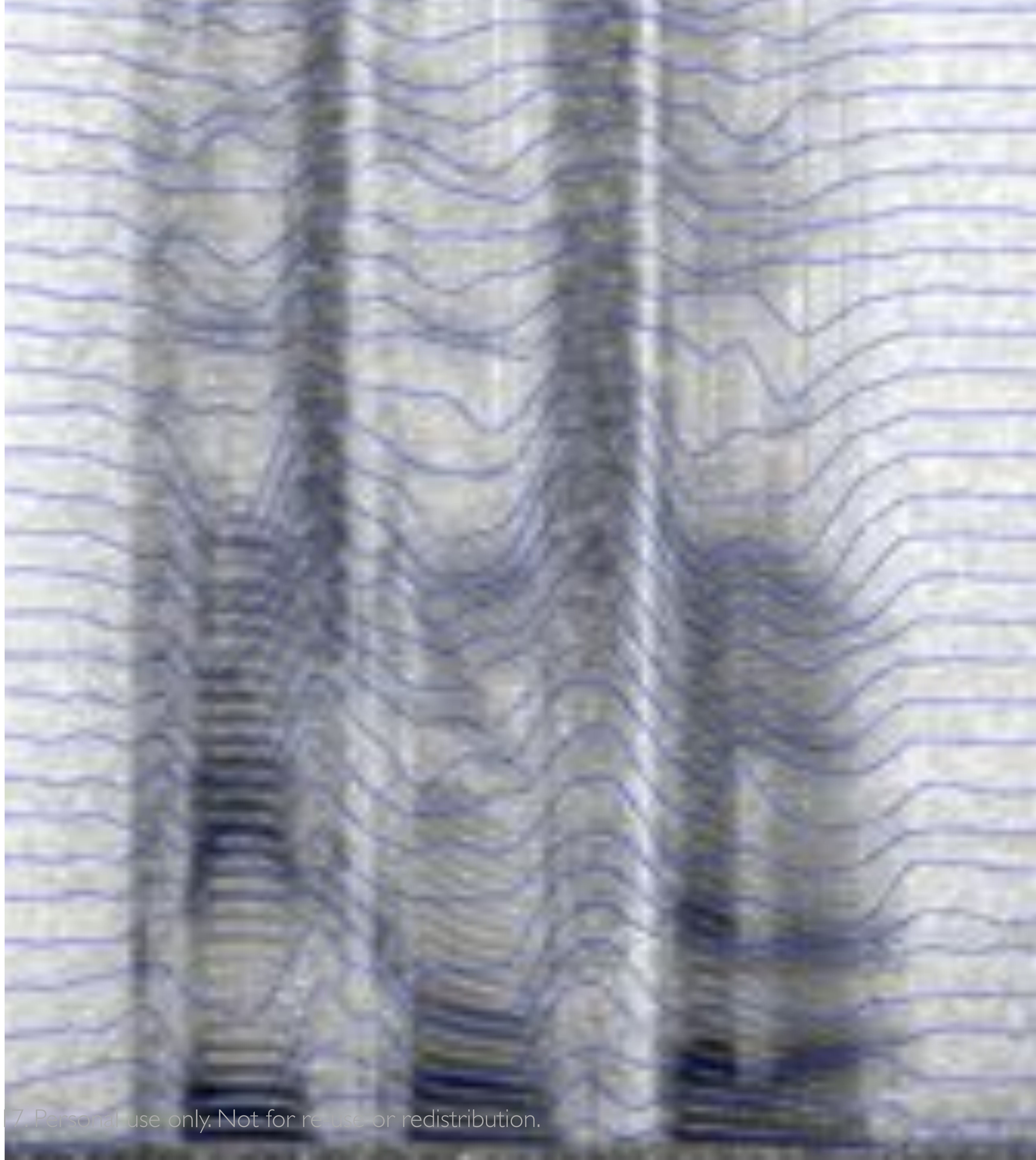
This lecture

My next lecture

Statistical Parametric Speech Synthesis

- <u>HMM-based (2 hours)</u>
- DNN-based (1 hour)

Simon King
University of Edinburgh



Text-to-speech key challenges

- We can identify four main challenges for any builder of a TTS system.
  1. Semiotic classification of text
  2. Decoding natural-language text
  3. Creating natural, human-sounding speech
  4. Creating intelligible speech
- We can also identify two current and future main challenges
  1. Generating affective and augmentative prosody
  2. Speaking in a way that takes the listener's situation and needs into account

(Taylor 2009, Section 3.6, page 51)

What properties of text do we need to know about?

"*it is not necessary to go all the way and uncover the meaning from the written signal;*

we have to perform just the job of text decoding, not also that of text understanding

……

by and large, the identity and order of the words to be spoken is all we require to synthesise speech;

no higher-order analysis or understanding is necessary."

(Taylor 2009, Section 3.1.2, page 29)

but Taylor adds two caveats:

- word sense disambiguation (e.g., "polish")
- prosody

What properties of speech do we need to know about?

- To start us thinking about the issues involved in creating synthetic speech, let's think first about what speech is "made of", because
  - in speech synthesis, we need to say **new** things (i.e., utterances not in our recorded database)
  - in speech recognition, we need to **generalise** from the examples in the training data to the speech we have to recognise

- It is convenient to think about speech as a **linear** sequence of units
  - enables a concatenative approach to speech synthesis
  - in speech recognition, allows us to string together models of small units (e.g. phonemes) to make models of larger units (e.g. words)

What you have learned so far

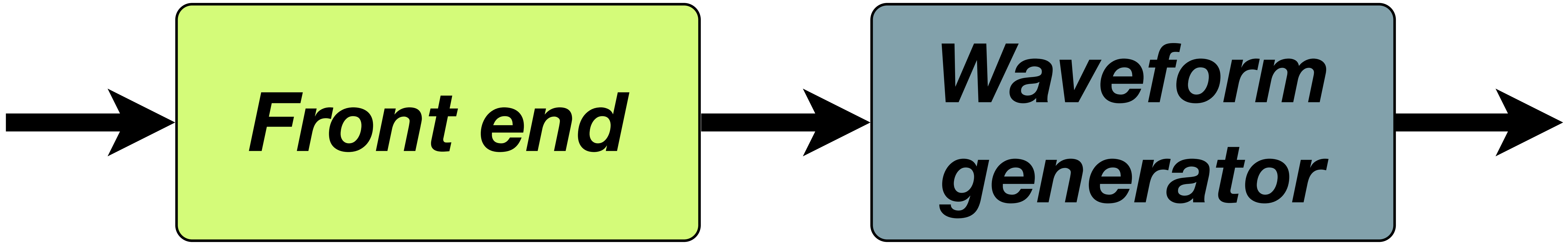
- <u>Unit selection synthesis</u>
  - how the target cost function uses the linguistic specification, by **querying** each feature (usually individually)
  - join cost encourages continuity of acoustic features

- <u>Speech **signal** modelling (vocoding)</u>
  - why we don't use the waveform
  - generalising the source-filter model

Contents

- The big picture
  - **text-to-speech, viewed as regression**
- Getting ready for regression
  - feature extraction from text
  - feature extraction from speech
- Doing regression
  - using a decision tree: so-called "HMM-based TTS"
  - using more powerful and general regression models: neural networks } My next lecture

The classic two-stage pipeline of unit selection



text

linguistic specification

waveform

"the cat sat"

The end-to-end problem we want to solve

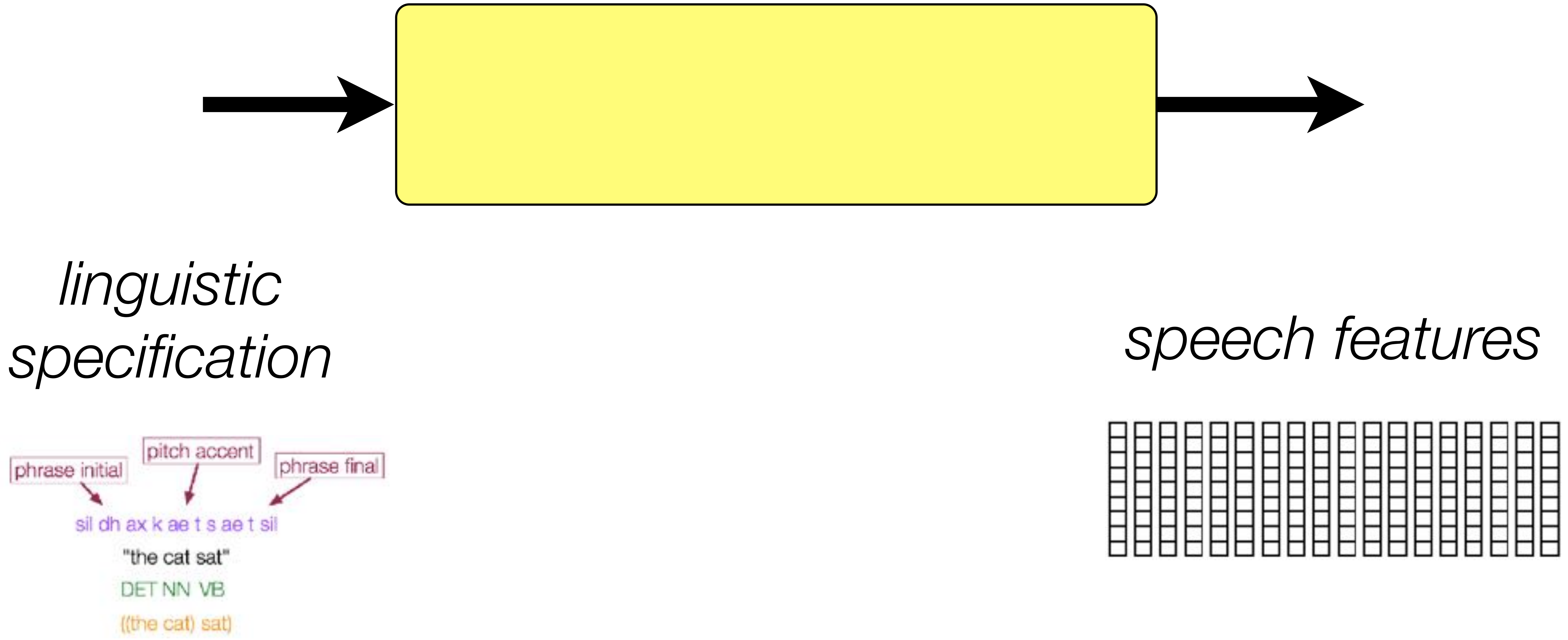
Text-to-Speech

text

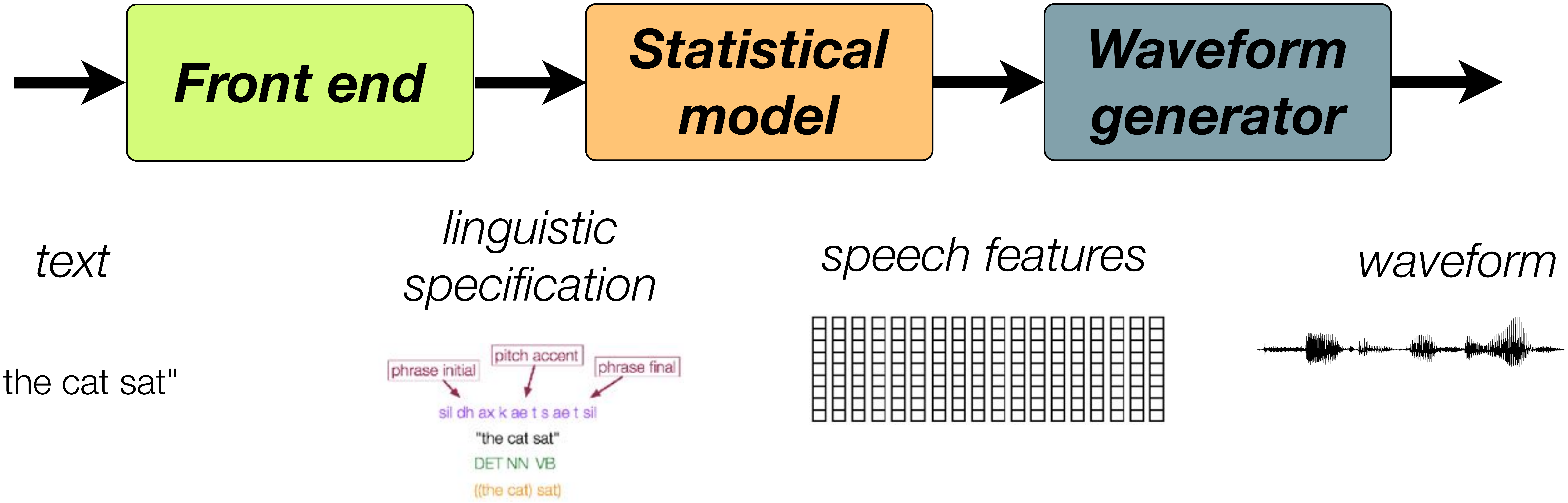
waveform

"the cat sat"

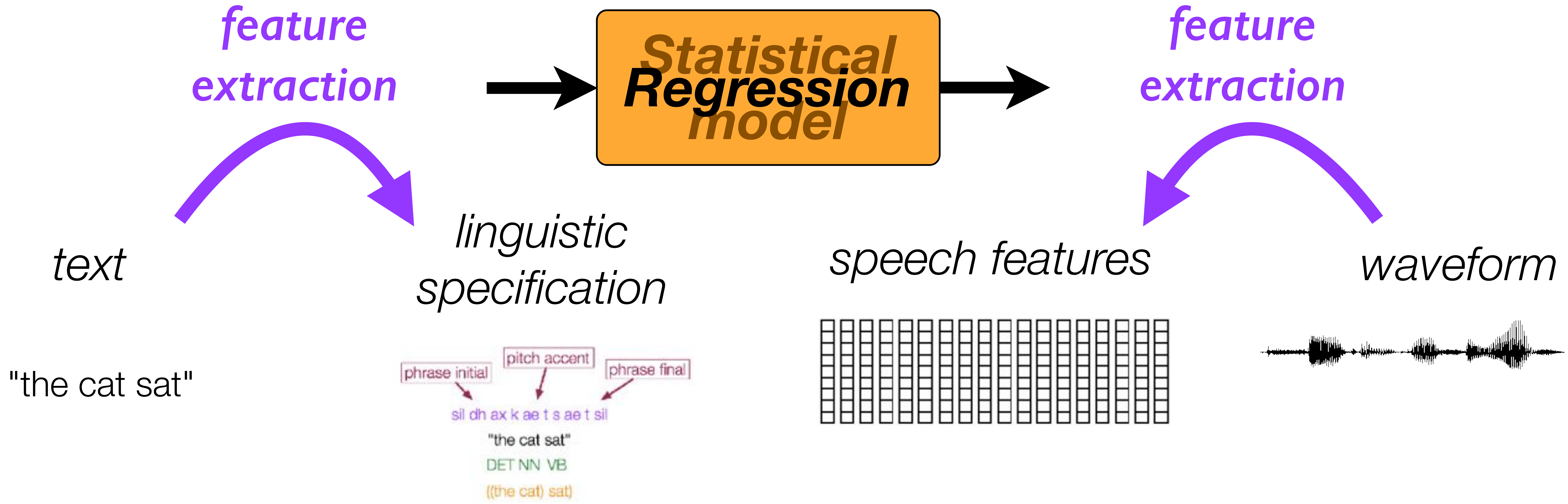
A problem we can actually solve with machine learning



The classic three-stage pipeline of statistical parametric speech synthesis

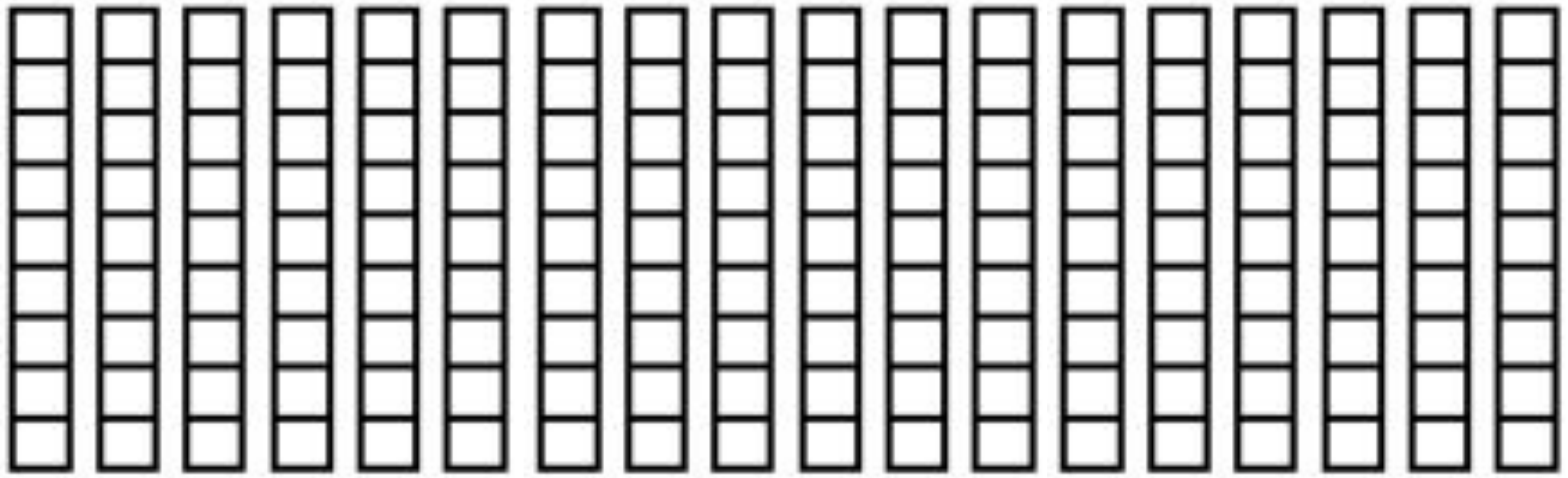


The classic three-stage pipeline of statistical parametric speech synthesis

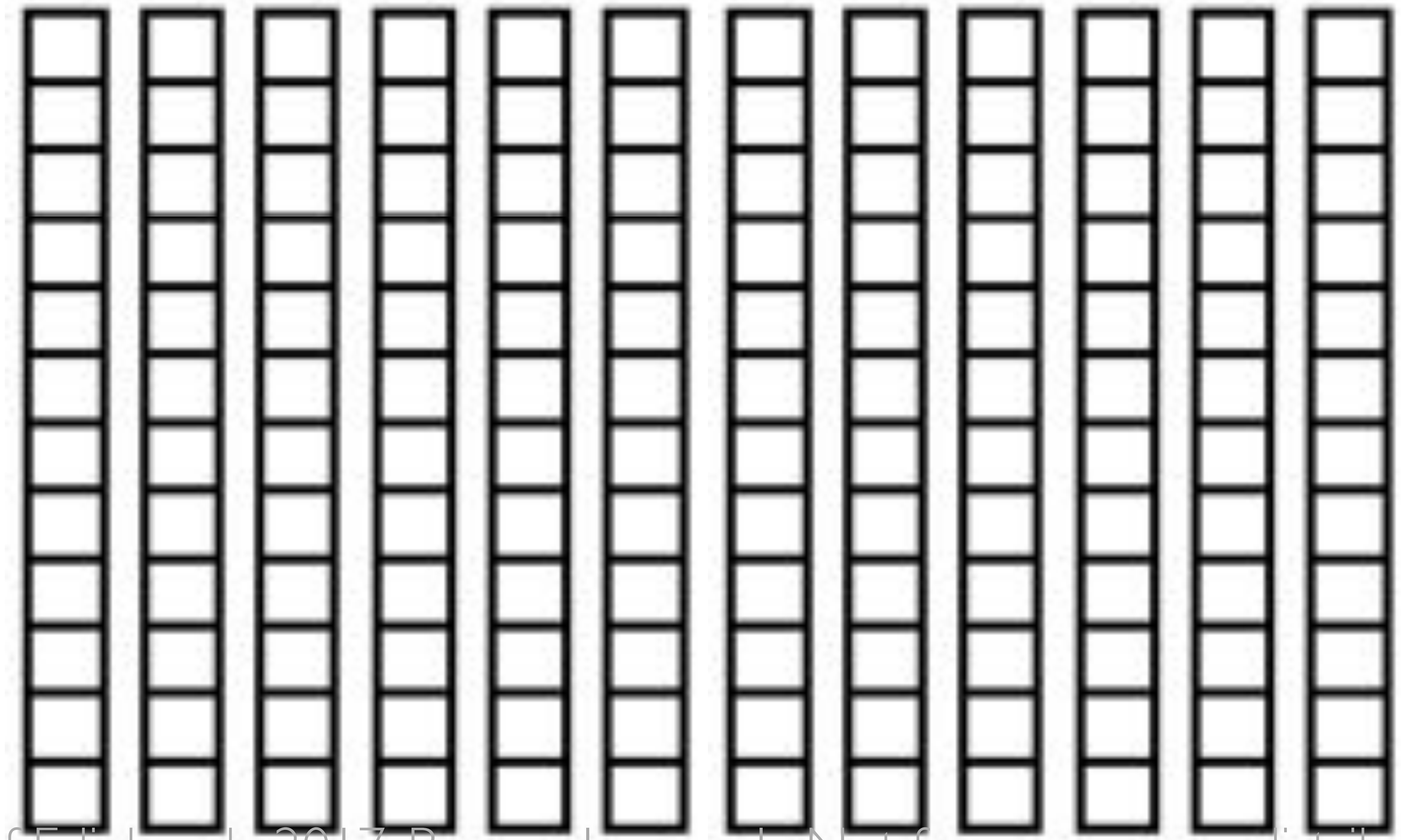


We can describe the core problem as **sequence-to-sequence regression**

output sequence
(speech features)



input sequence
(linguistic specification)



Orientation

- Unit selection
  - selection of waveform units based on
    - target cost
    - join cost
- Speech signal modelling
  - generalised source+filter model

- Statistical parametric synthesis
  - predict **speech parameters** from **linguistic specification**

Orientation

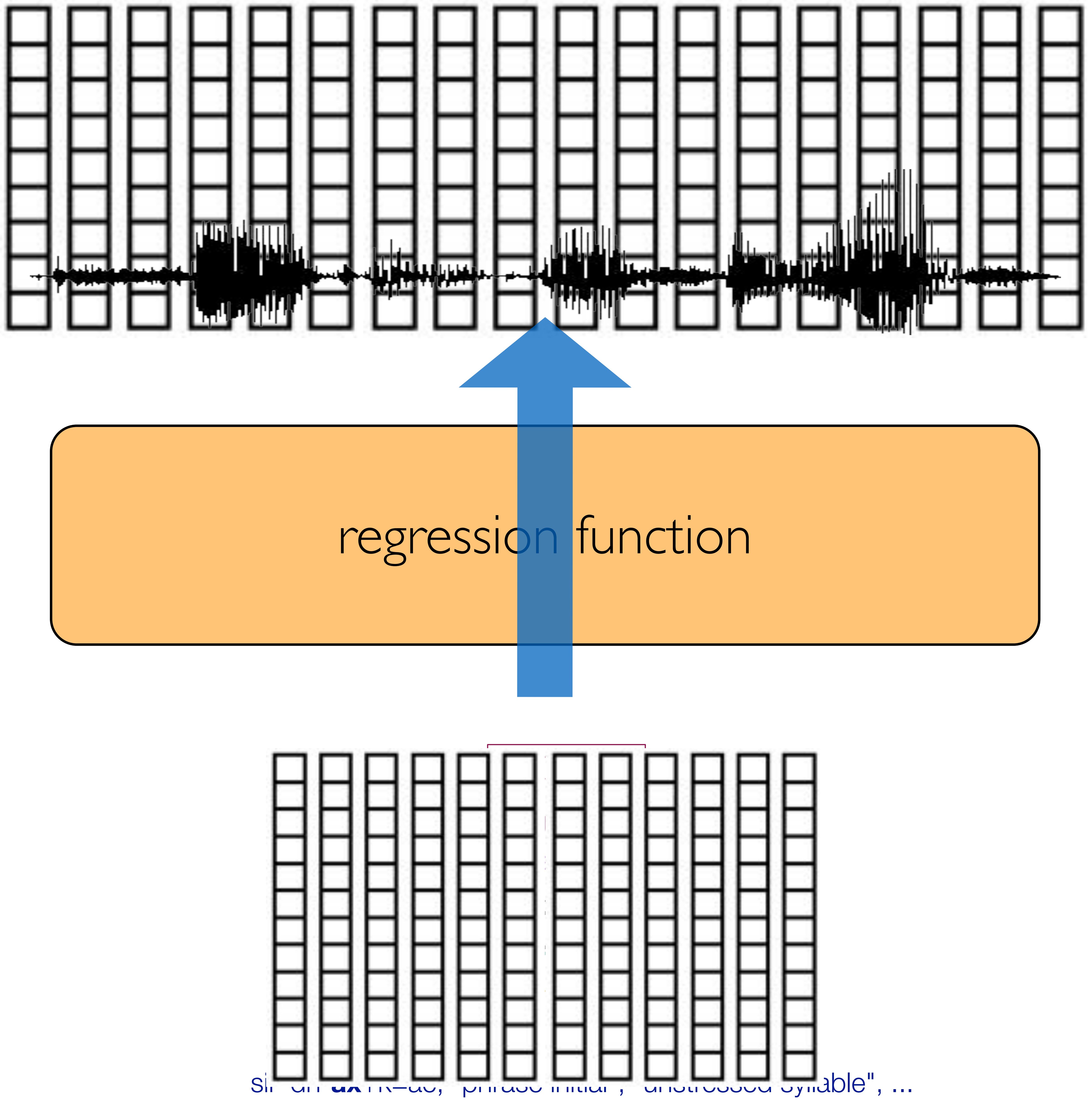
- Unit selection
  - selection of waveform units based on
    - target cost
    - join cost
- Speech signal modelling
  - generalised source+filter model
- Statistical parametric synthesis
  - predict **speech parameters** from **linguistic specification**

Let's just consider the type of target cost that is based only on the **linguistic specification**

There are several ways to do this, but we need to be able to
- **separate** excitation & spectral envelope
- **reconstruct** the waveform

A **regression** task!

Orientation



- Statistical parametric synthesis
  - predict **speech parameters** from **linguistic specification**

What are the input features ?

Just the linguistic features !

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

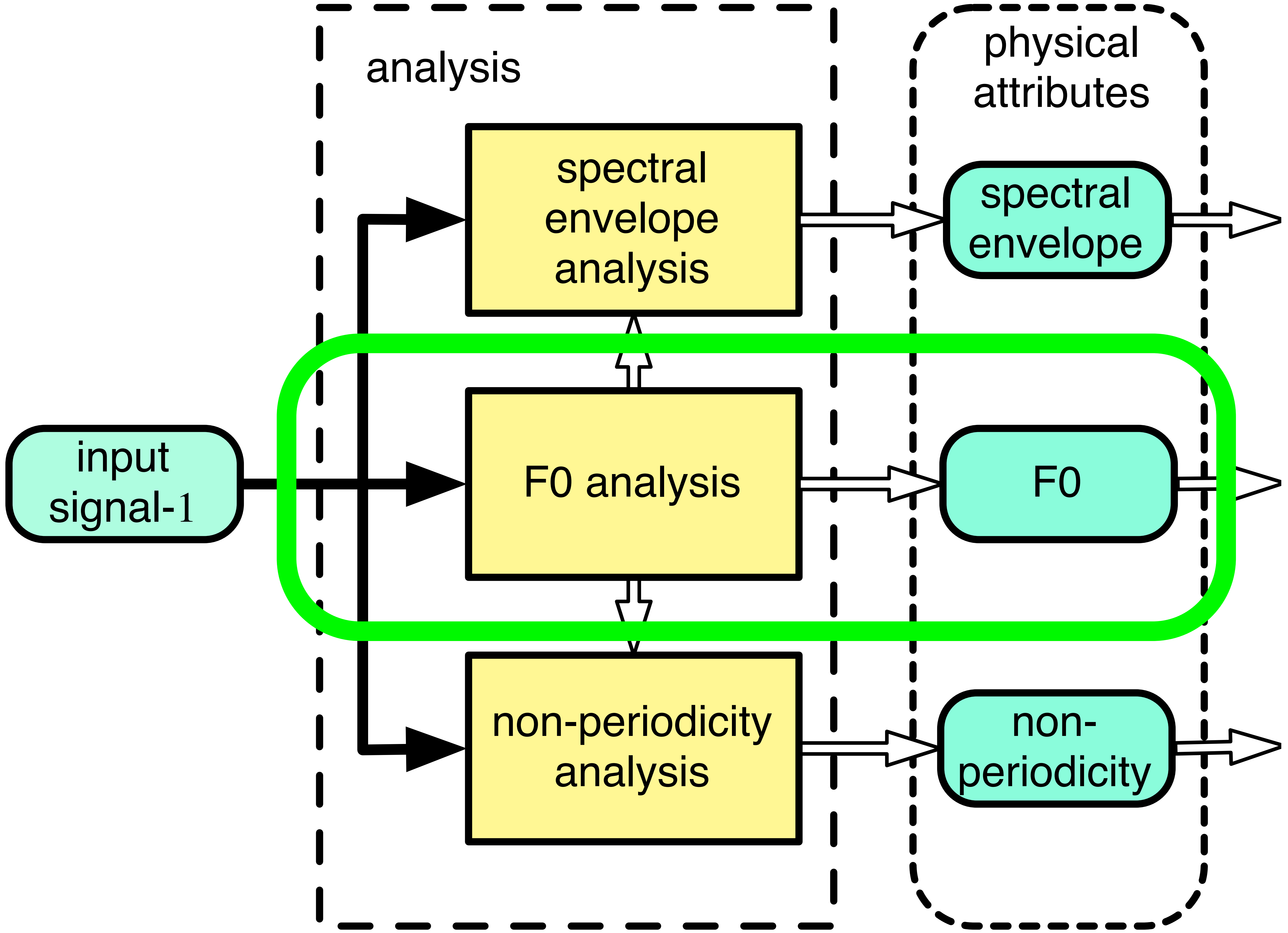
DET NN VB

((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

input feature vector

What are the output features (i.e., speech parameters) ?



speech parameters

output feature vector

What next?

- Feature extraction + *feature engineering*
  - constructing the input features
  - constructing the output features

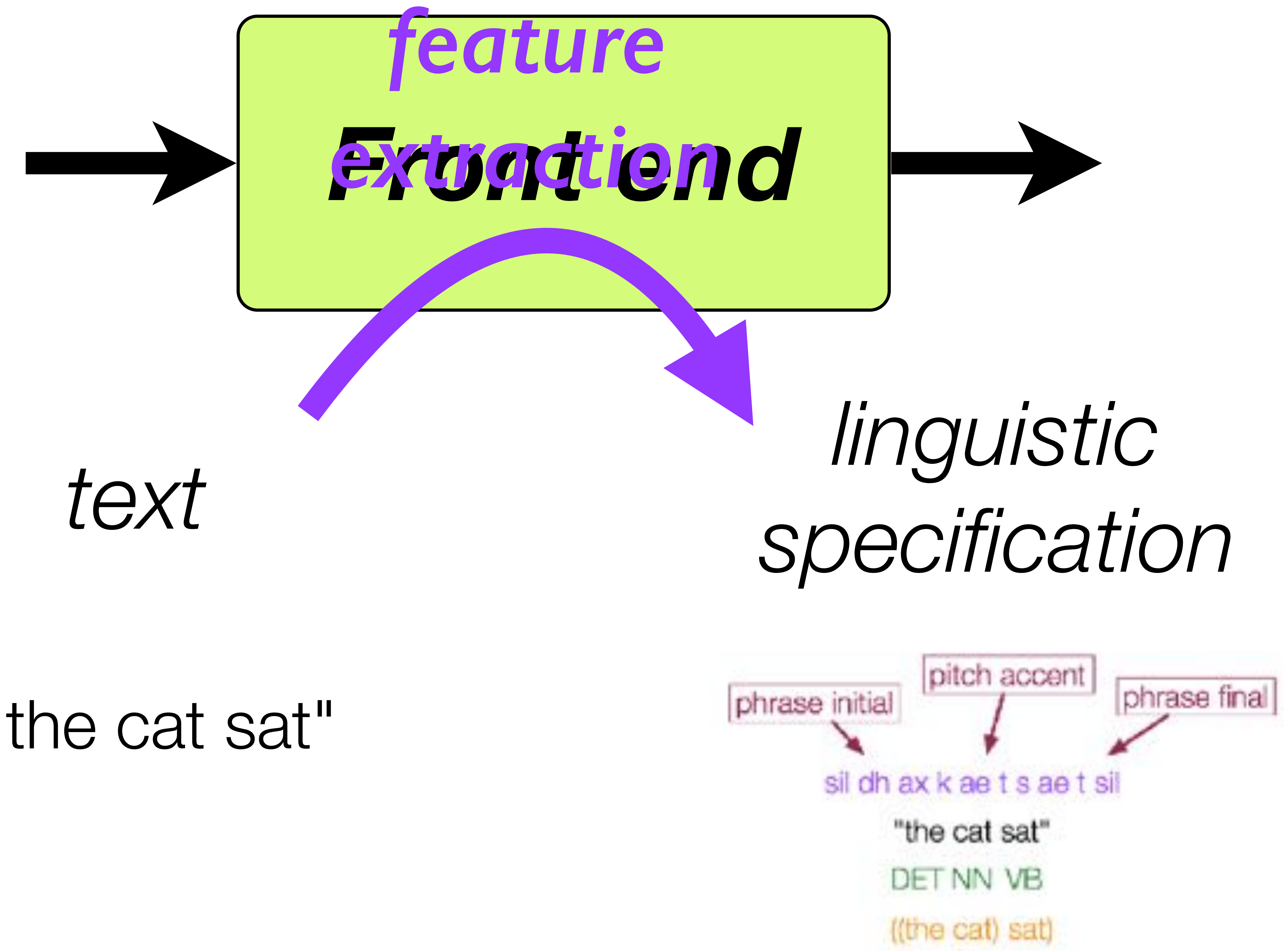
- Then, performing the regression


PAUSE/STILL

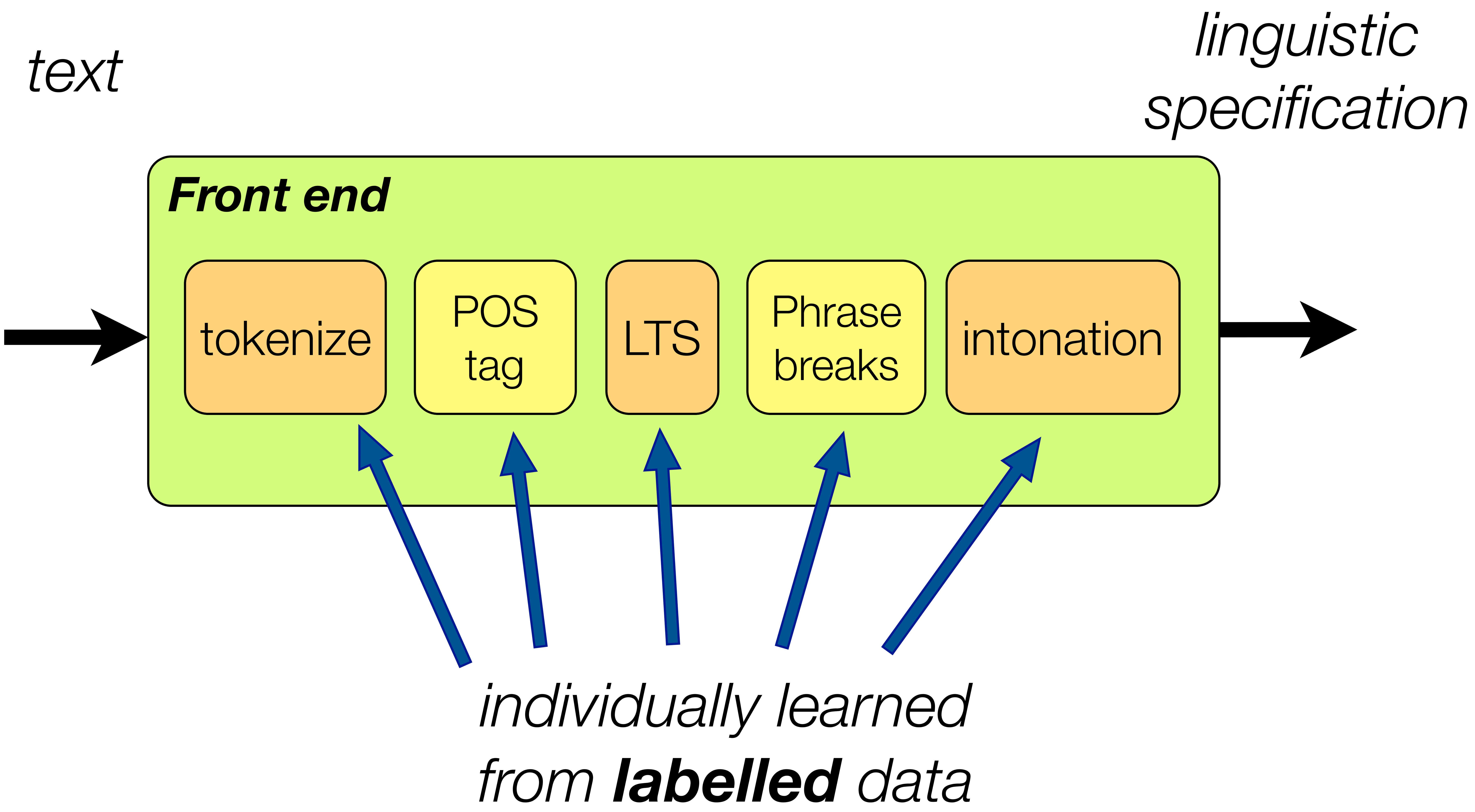
Contents

- The big picture
  - text-to-speech, viewed as regression ✔
- Getting ready for regression
  - **feature extraction from text**
  - feature extraction from speech
- Doing regression
  - using a decision tree: so-called "HMM-based TTS"
  - using more powerful and general regression models: neural networks } My next lecture

Extracting features from text using the front end

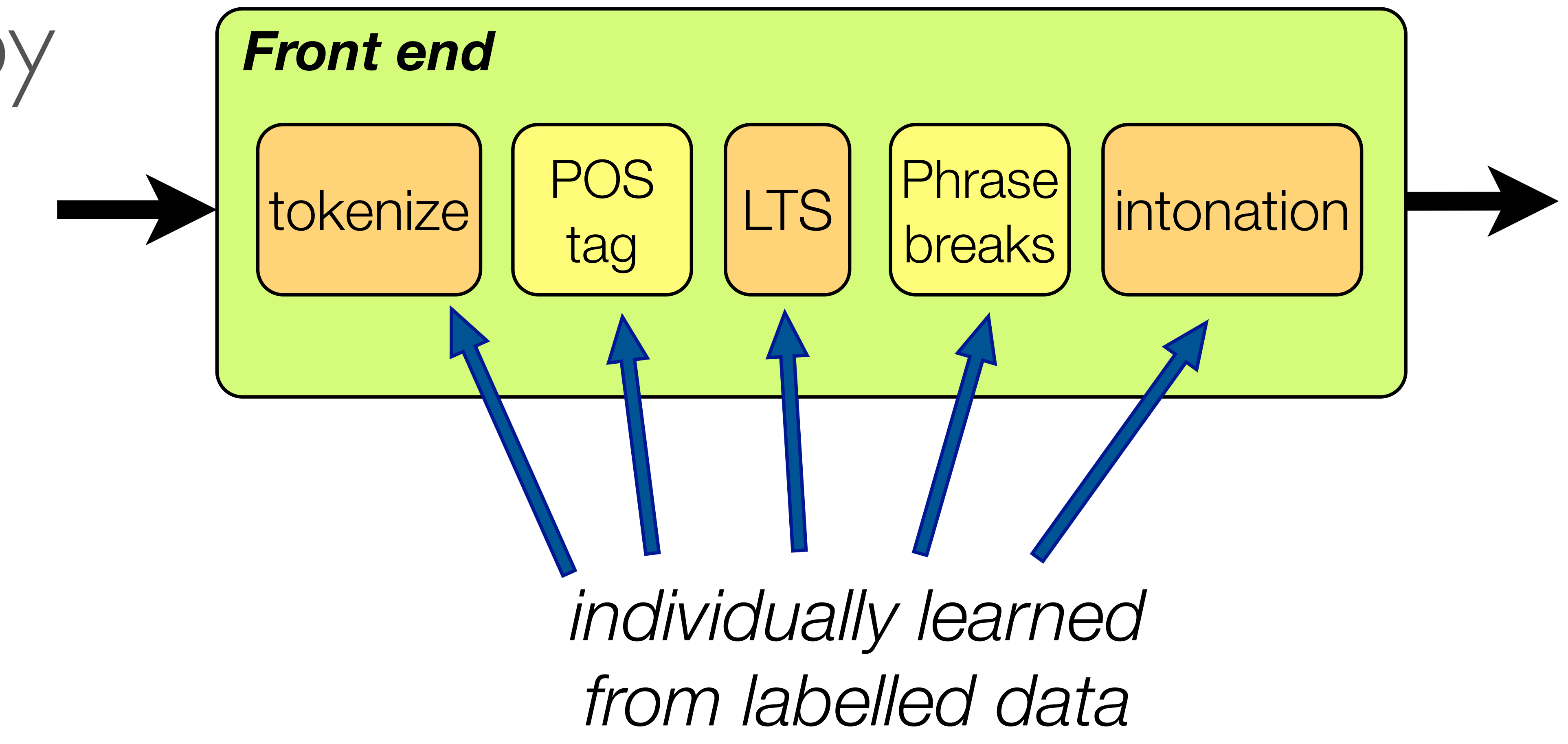


Text processing pipeline

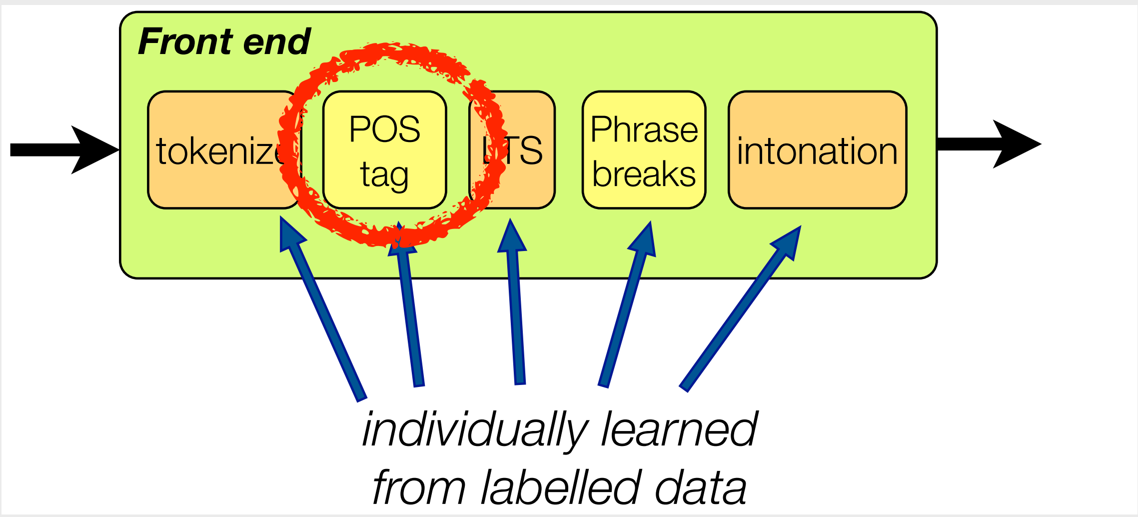


Text processing pipeline

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



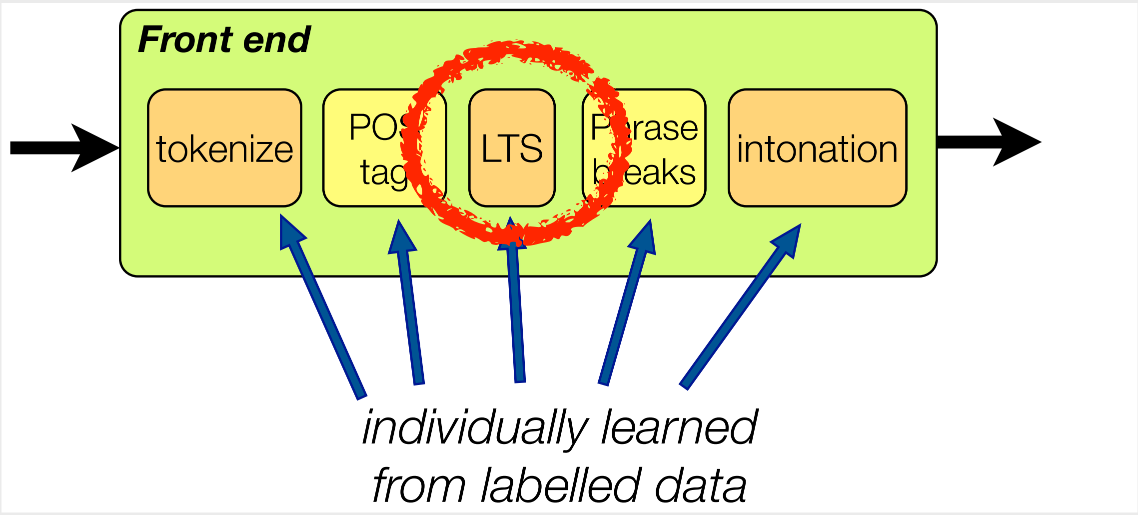
POS tagging



- Part-of-speech tagger
- Accuracy is very high
- But
  - 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
```

Pronunciation / LTS



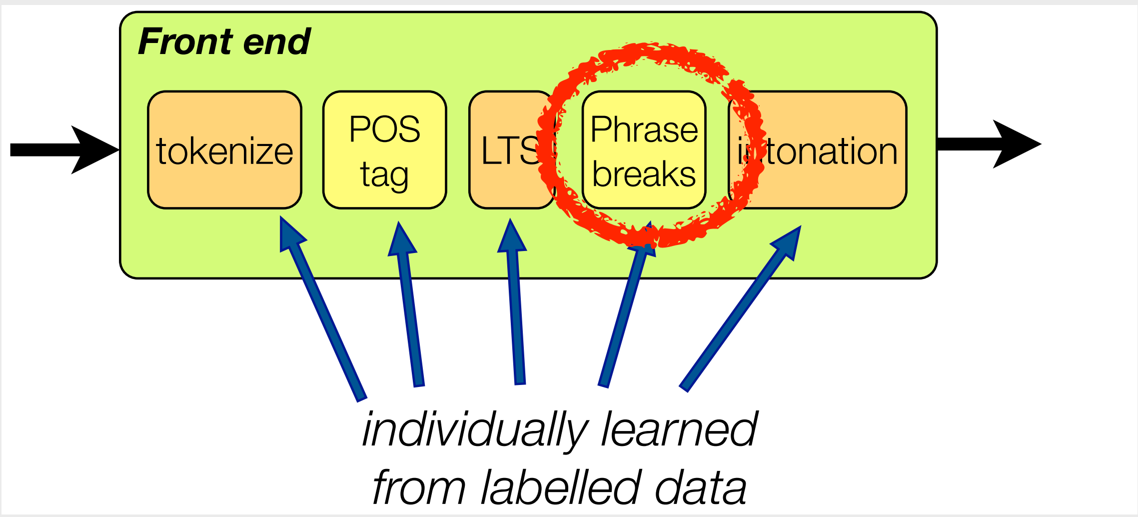
- Pronunciation model
  - dictionary look-up, *plus*
  - letter-to-sound model
- <u>But</u>
  - need deep **knowledge** of the language to design the phoneme set
  - human **expert** must write dictionary

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

```
ADVOCATING   AE1 D V AH0 K EY2 T IH0 NG
ADVOCATION   AE2 D V AH0 K EY1 SH AH0 N
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 IH0 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

Predict phrase breaks

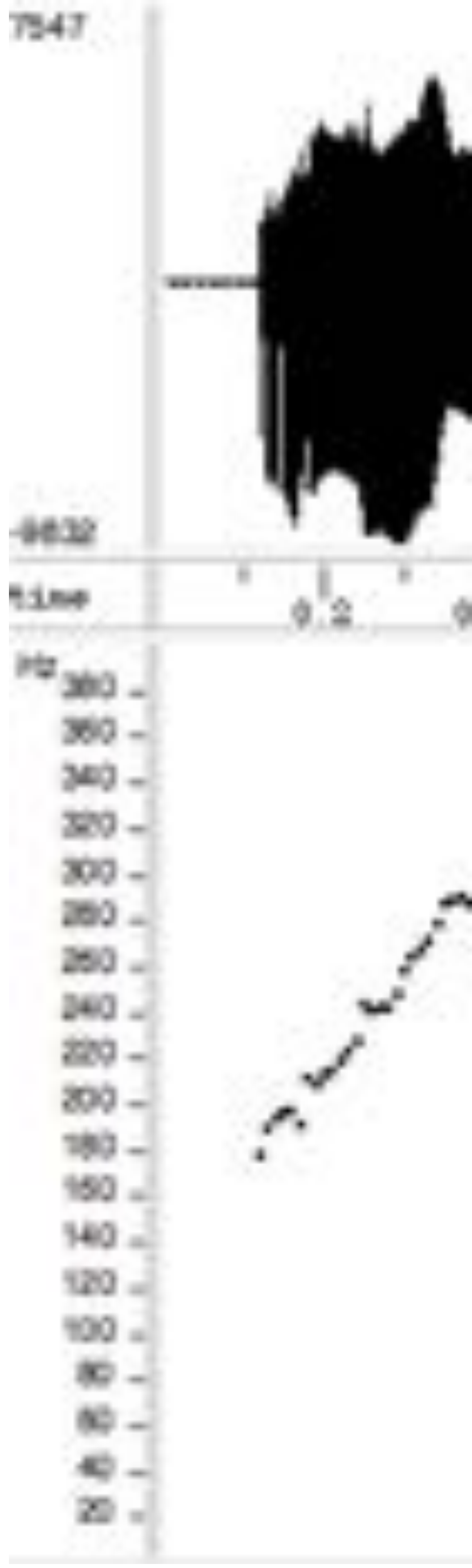


- Phrase-break prediction
  - binary classifier using sequence as input
- But
  - trained on **annotated** spoken data
  - therefore very **small** training set

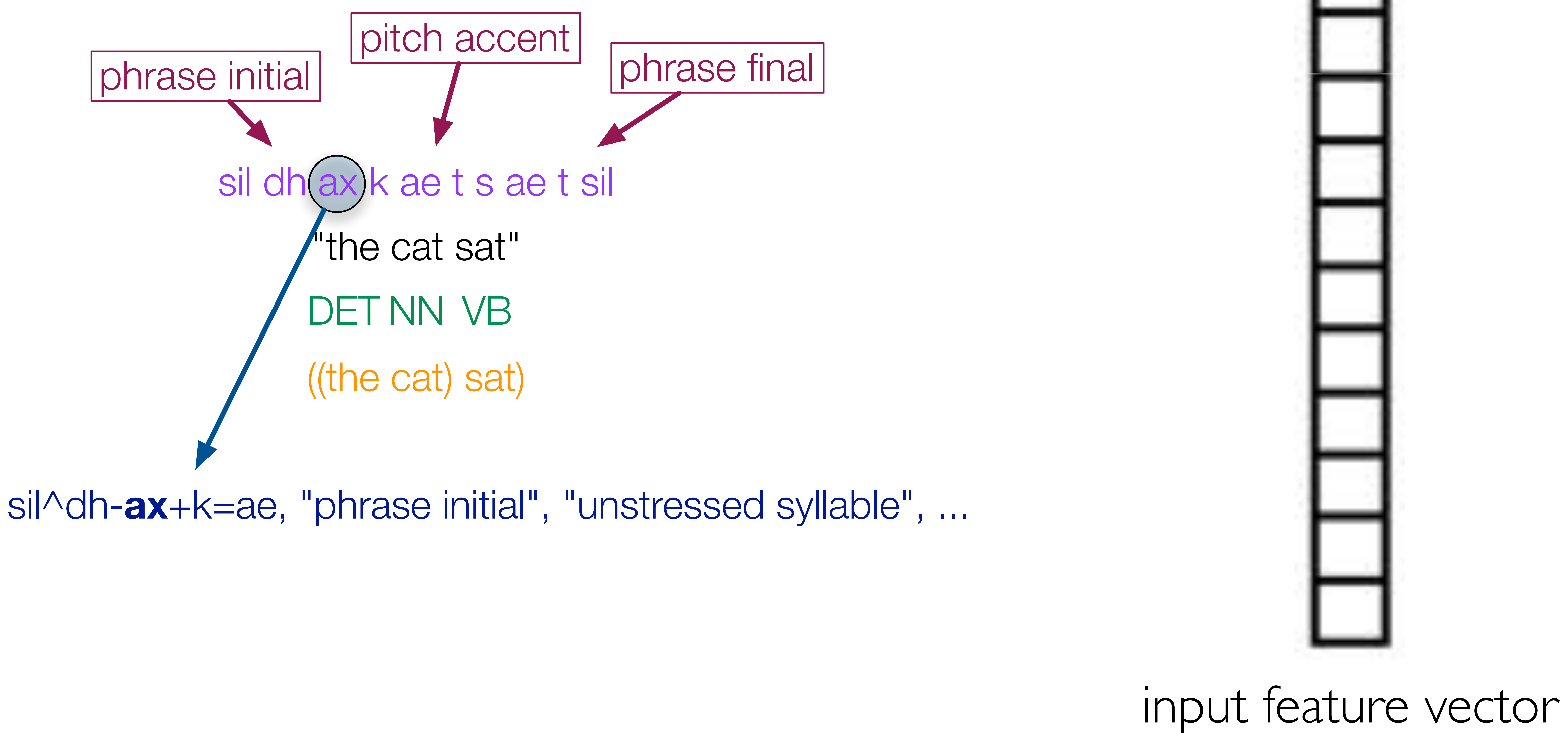
This sequence is the annotated training data for our phrase break predictor

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

break !



We will need to convert the linguistic specification to a vector …. later !



Contents

- The big picture
  - text-to-speech, viewed as regression ✔
- Getting ready for regression
  - feature extraction from text ✔
  - **feature extraction from speech**
- Doing regression
  - using a decision tree: so-called "HMM-based TTS"
  - using more powerful and general regression models: neural networks } My next lecture

Feature extraction from speech

- speech parameters
- representations suitable for modelling (*feature engineering*)
- converting back to a waveform

Orientation

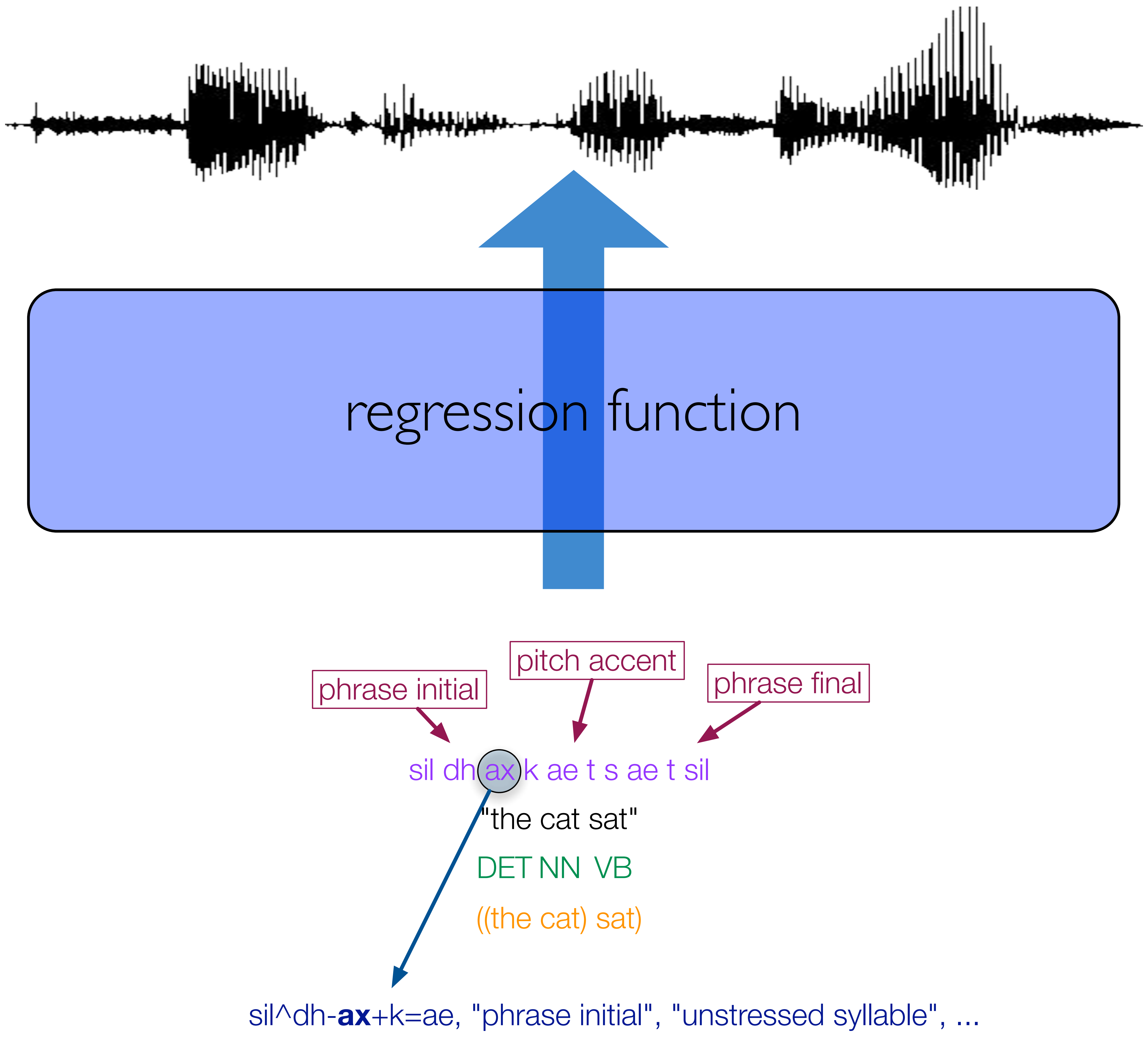
- <u>So far</u>: speech signal **analysis**
  - epochs
  - F0
  - spectral envelope

- <u>Now</u>: speech signal **modelling**
  - speech parameters
  - representations suitable for modelling
  - converting back to a waveform

Orientation

- So far: speech signal **analysis**
  - epochs
  - F0
  - spectral envelope

- Now: speech signal **modelling**
  - speech parameters
  - representations suitable for modelling
  - converting back to a waveform



Orientation

- So far: speech signal **analysis**
  - epochs
  - F0
  - spectral envelope

- Now: speech signal **modelling**
  - speech parameters
  - representations suitable for modelling
  - converting back to a waveform

- smooth spectral envelope
- fundamental frequency (F0)
- *aperiodic energy*

Feature extraction from speech

- speech parameters
- representations suitable for modelling (*feature engineering*)
- converting back to a waveform

Representations of the speech parameters that are suitable for modelling

- Many vocoders are conceptually based on a source-filter model
  - except they use an *excitation signal* + *spectral envelope*, not the "true" source+filter

- excitation signal
  - a periodic signal (e.g., a pulse train) at a frequency of F0
  - switched on and off by a voiced/unvoiced (V/UV) decision
- spectral envelope
  - we need a **representation** that is amenable to statistical modelling
- aperiodic energy
  - spectrally-shaped noise

Representations of the speech parameters that are suitable for modelling

- We want parameters that are
  - **fixed** in number (per frame) and as **low dimensional** as possible
  - at a **fixed** frame rate
  - a good **separation** of prosodic and segmental identity aspects of speech
    - so that we can model (and/or modify) either of them independently
  - **well behaved** and stable, when we perturb them (e.g., by averaging, or modelling error)
    - consecutive frames within a single speech sound
    - frames pooled from several similar sounds
- and for statistical modelling, we may additionally like to have
  - statistically **uncorrelated** parameters (to avoid having to model covariance)

What does STRAIGHT actually produce?

- … and is it suitable for modelling?
- <u>smooth spectral envelope</u>
- <u>F0</u>
- <u>non-periodicity</u>
  - in other words, aperiodic energy

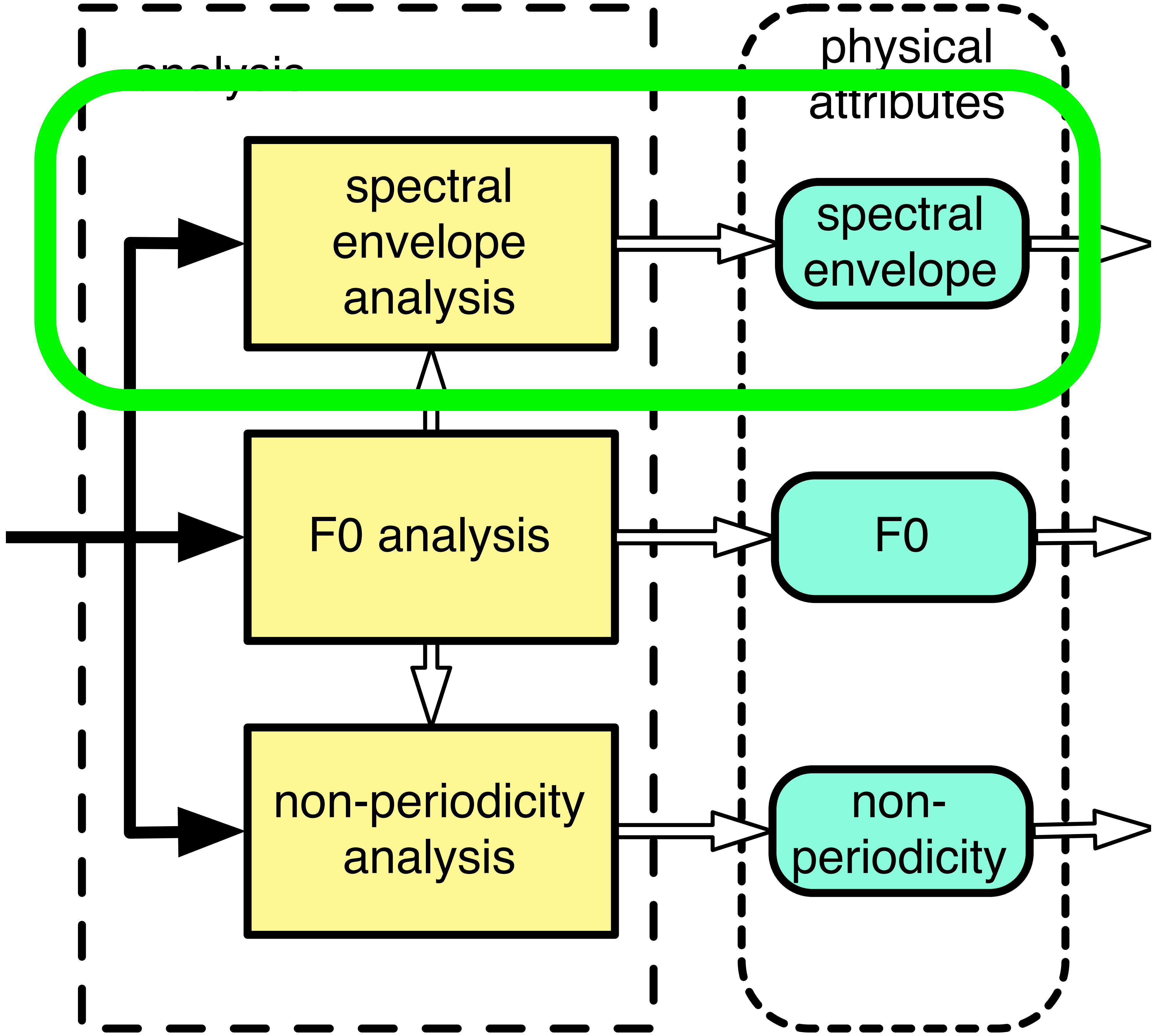


Figure: Hideki Kawahara

What does STRAIGHT actually produce?

- <u>smooth spectral envelope</u>

- high resolution (same as FFT)
- highly-correlated parameters
- probably **not** suitable for statistical modelling
  - at least, not with diagonal-covariance Gaussians

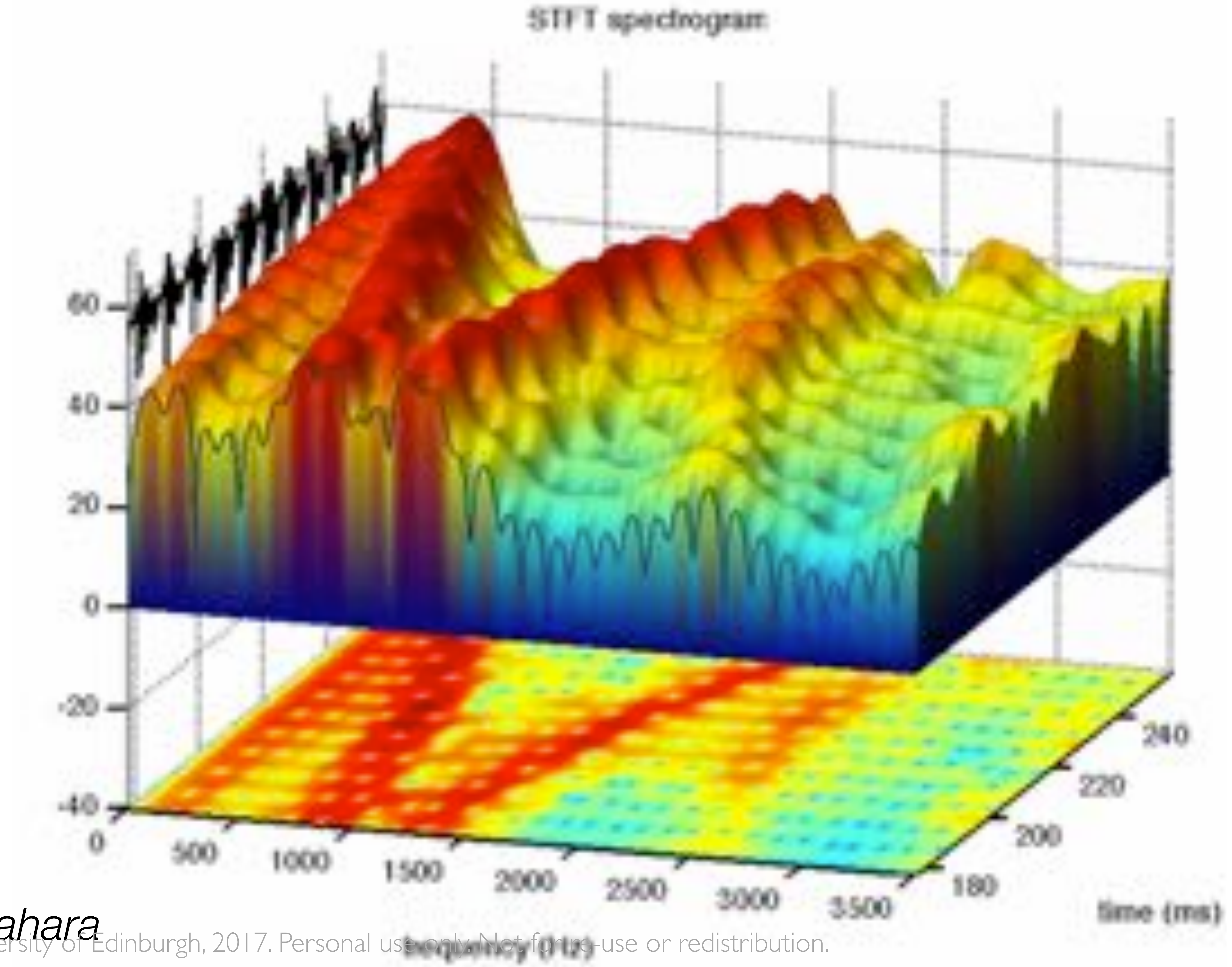


Figure: Hideki Kawahara

Improving the representation of the spectral envelope

- warp frequency scale
- decorrelate
- reduce dimensionality

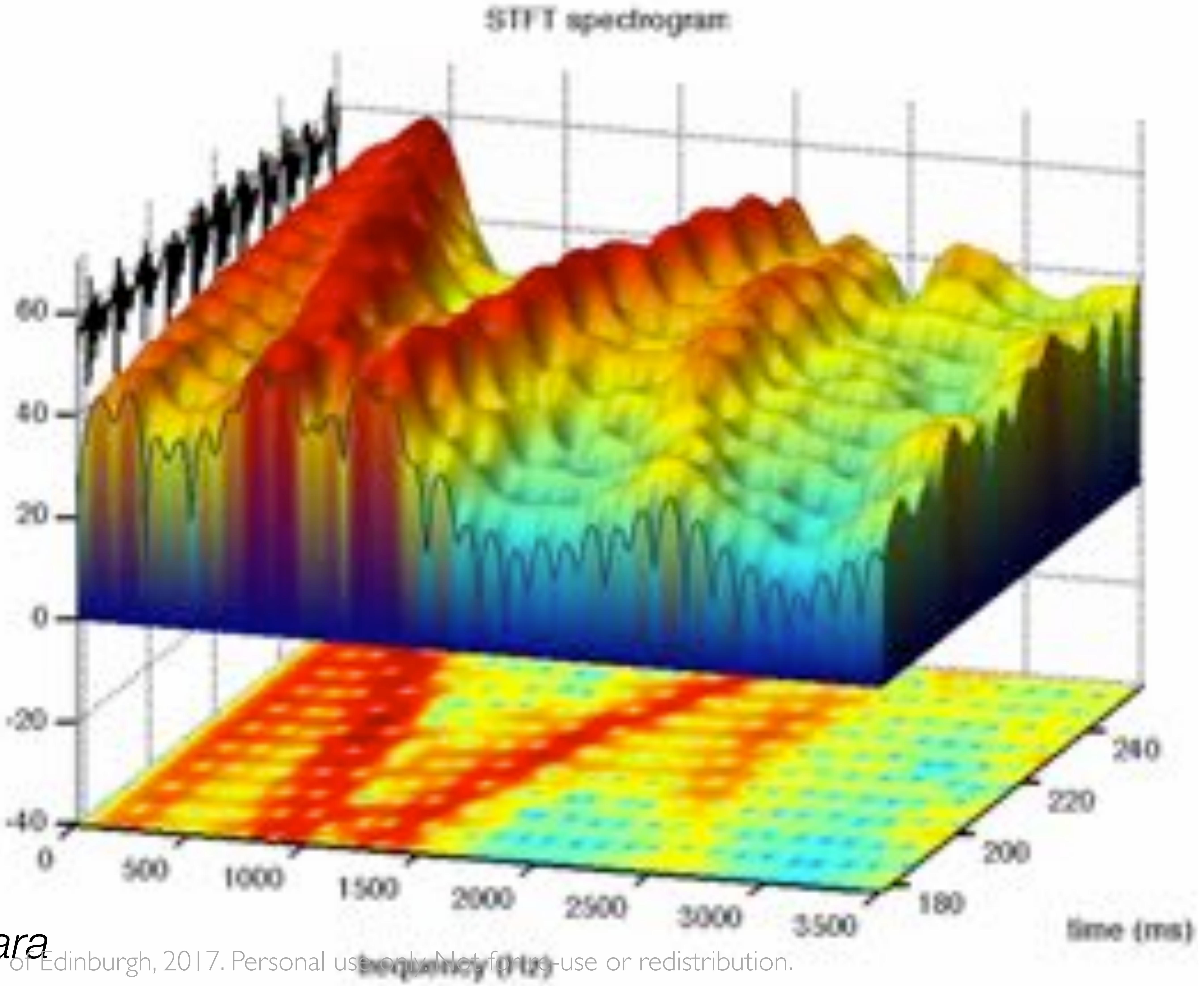


Figure: Hideki Kawahara

Representing the spectral envelope as the Mel-cepstrum

- Not quite the same as the MFCCs we use in ASR, but basically the **same motivation**

- <u>warp the frequency scale</u>
  - instead of lossy discrete filterbank, use a **continuous** function (all-pass filter)
- <u>decorrelate</u>
  - convert from spectrum to **cepstrum**
- <u>reduce dimensionality</u>
  - **truncate** the cepstrum
  - in ASR, we kept the first 12 coefficients
  - in synthesis, we'll use a lot more, perhaps the first 40-60 coefficients

What does STRAIGHT actually produce?

- <u>aperiodic energy</u>
  - effectively the ratio between periodic and aperiodic energy, at each frequency

- high resolution (same as FFT)
- highly-correlated parameters

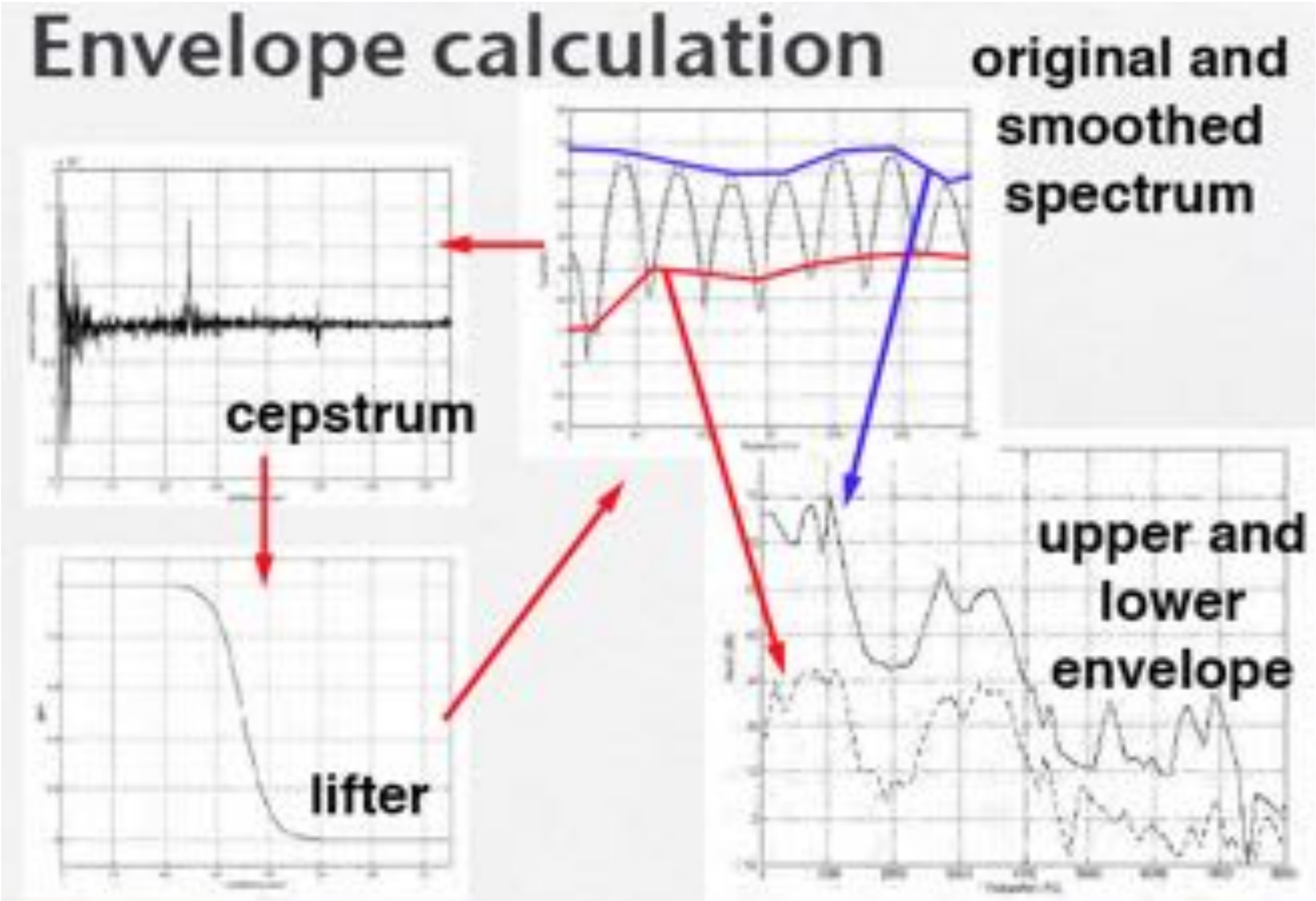
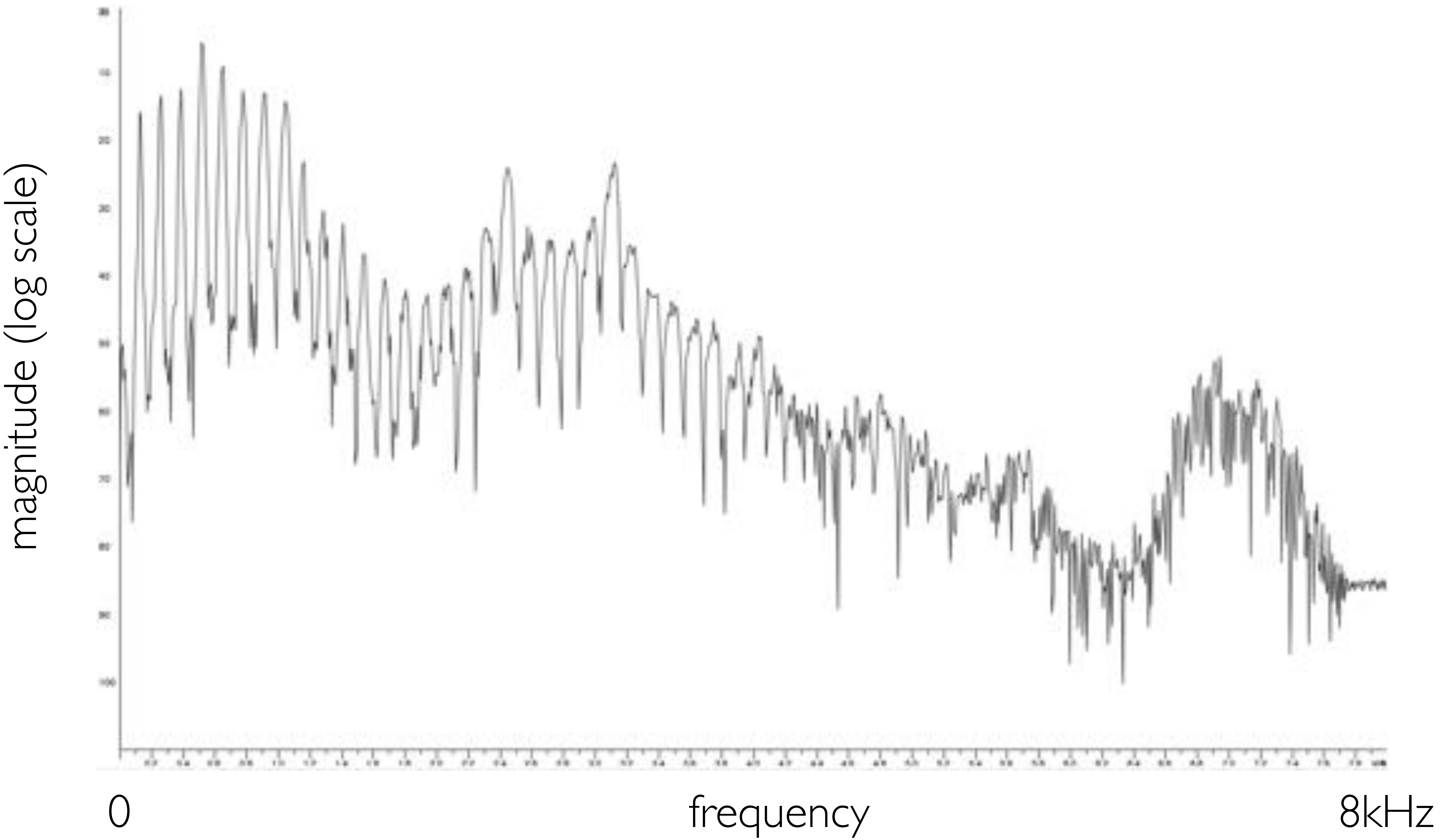


Figure: Hideki Kawahara


magnitude (log scale)
0
frequency
8kHz

Improving the representation of the aperiodic energy

- <u>aperiodic energy</u>

- reduce dimensionality
  - simply reduce resolution by averaging across broad frequency **bands**
  - e.g., between 5 and 25 bands (on a Mel scale, of course)

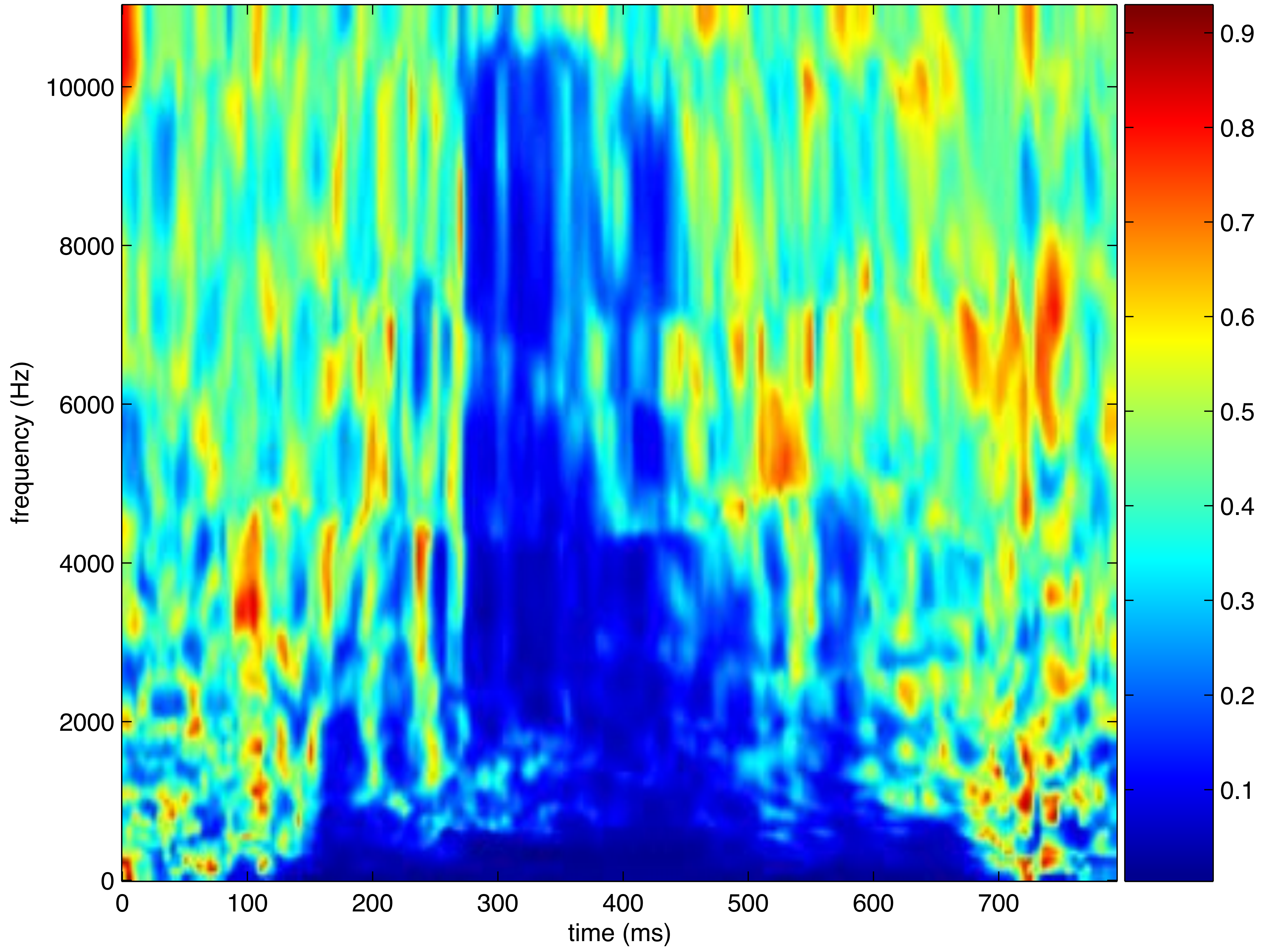
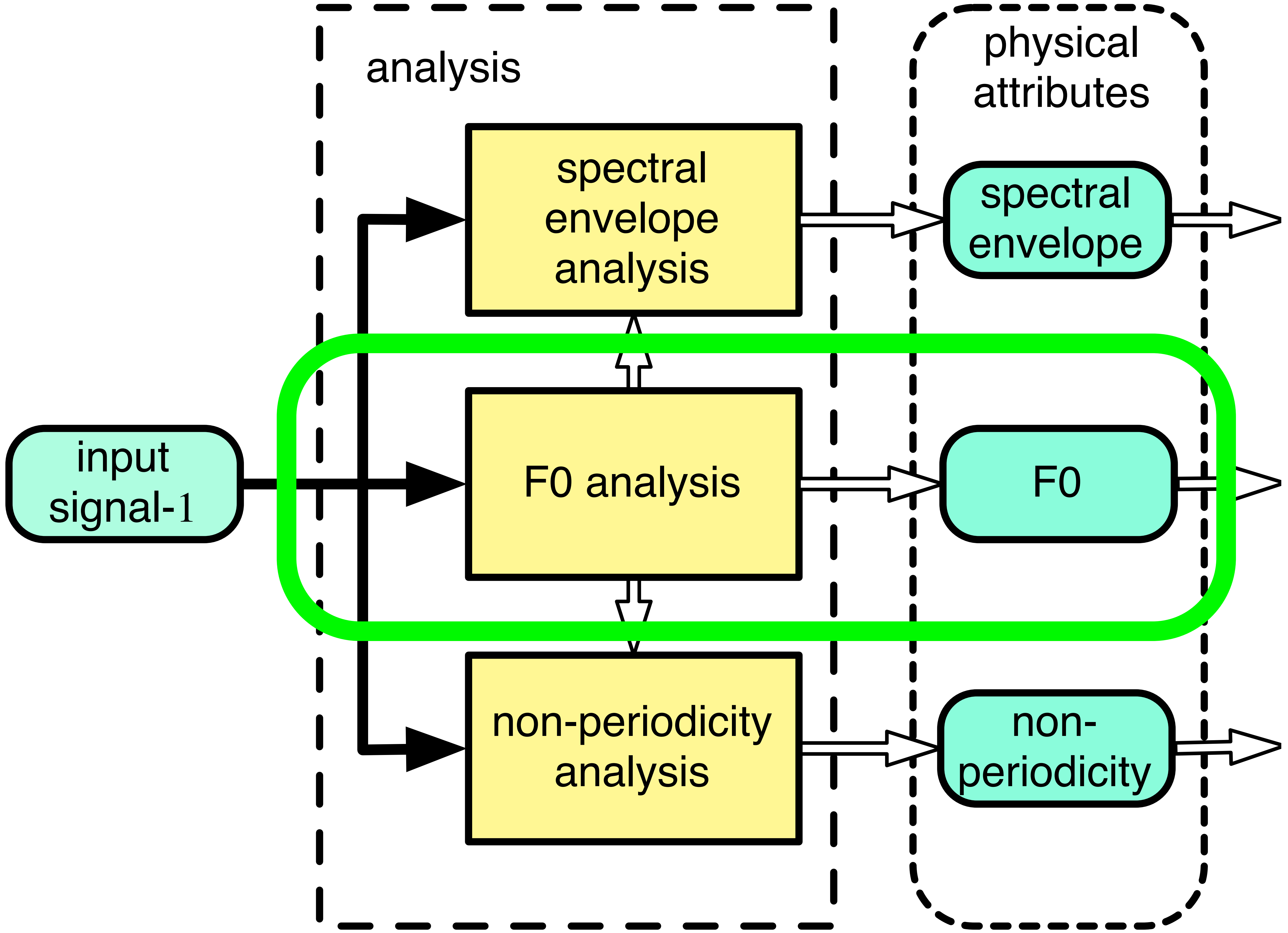


Figure: Hideki Kawahara

Final representation of speech parameters, after feature engineering



speech parameters

output feature vector

Feature extraction from speech

- speech parameters
- representations suitable for modelling (*feature engineering*)
- converting back to a waveform

STRAIGHT analysis and synthesis

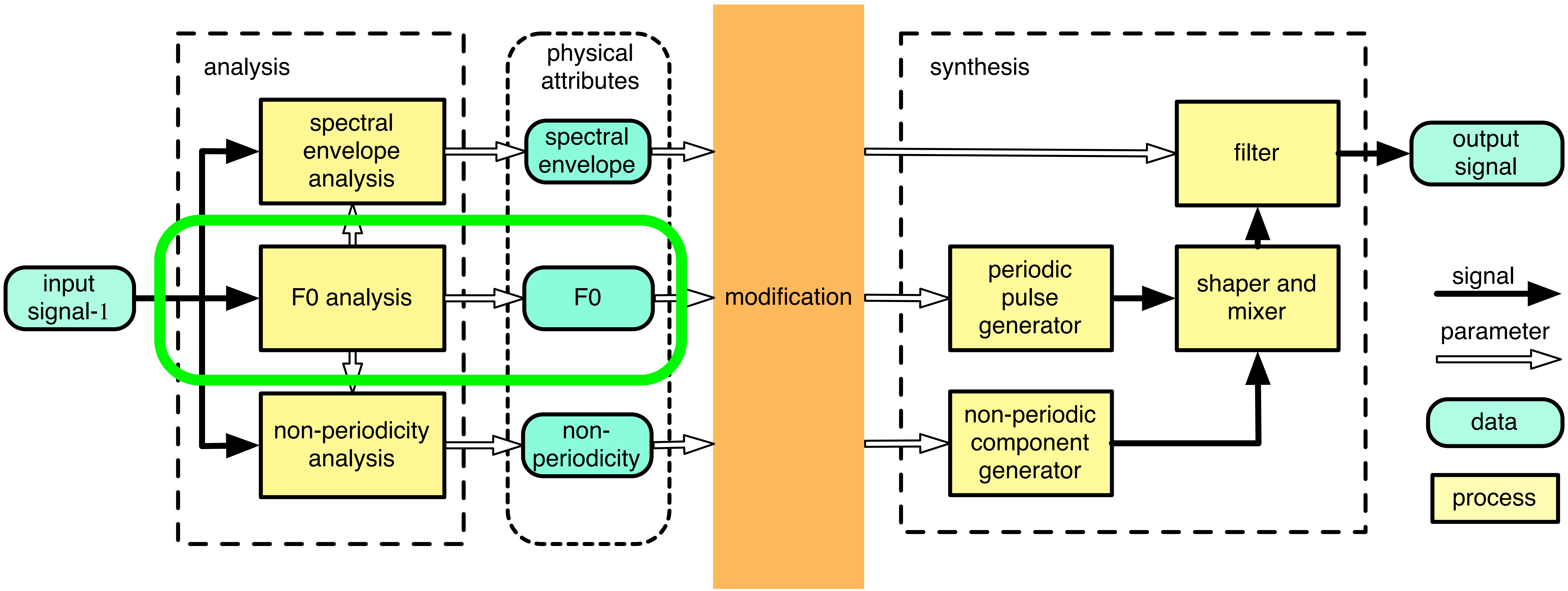


Figure: Hideki Kawahara

Excitation signal

pulse/noise

STRAIGHT

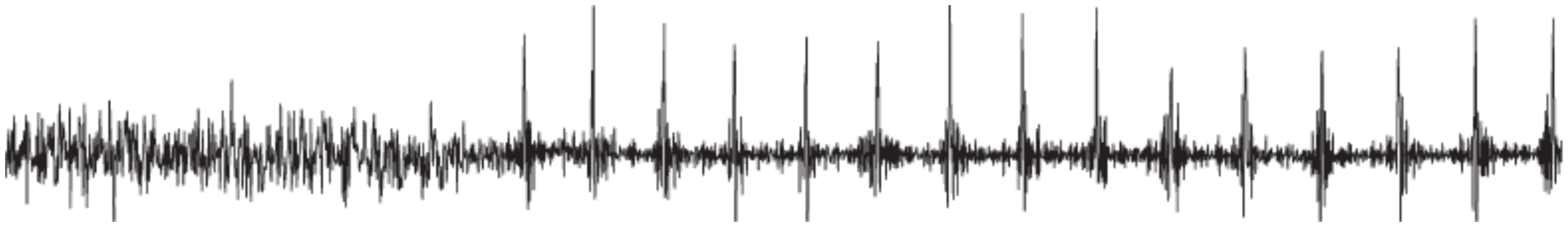


figure from Heiga Zen

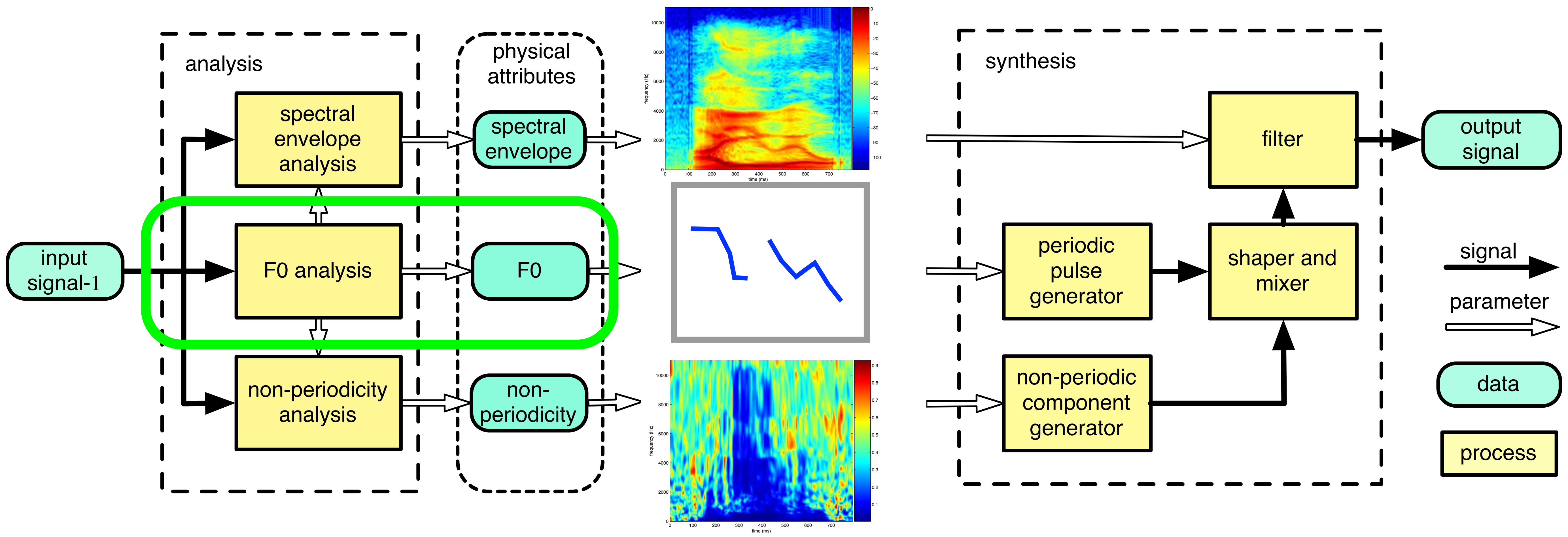
What next?

- We have decomposed speech into
  - F0, plus a V/UV decision
  - smooth spectral envelope, parameterised as the Mel-cepstrum
  - band aperiodicity parameters
- We've seen how to reconstruct the waveform

- Now we can insert a **statistical model** between the analysis and synthesis parts

What next?

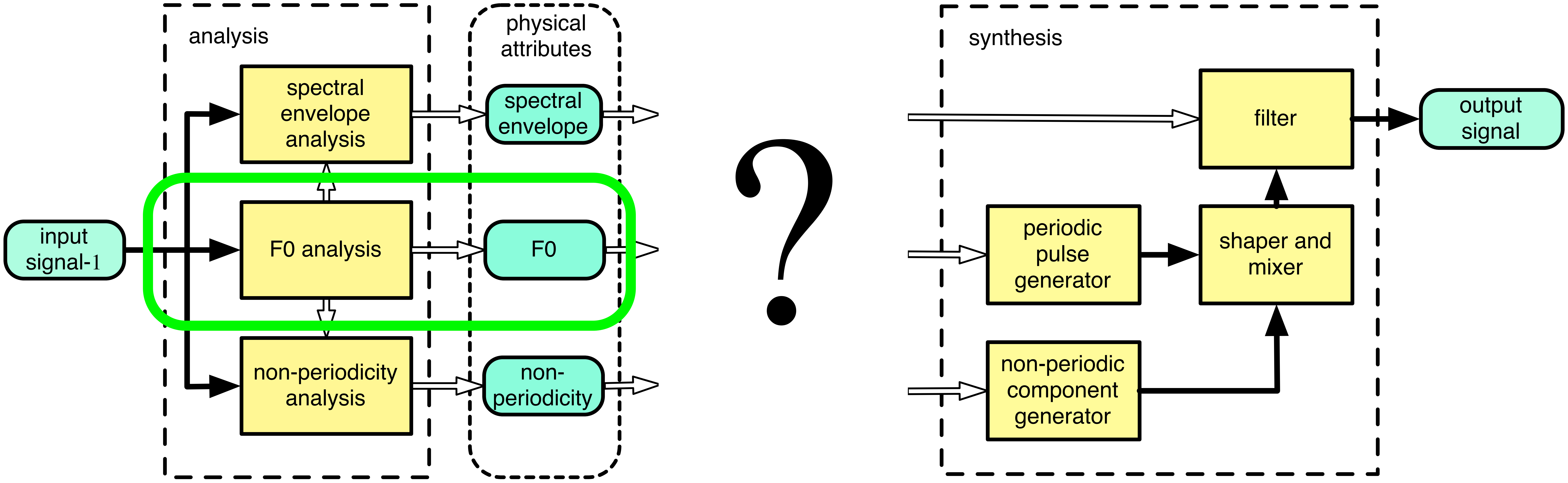
- We have decomposed speech into
  - F0, plus a V/UV decision
  - smooth spectral envelope, parameterised as the Mel-cepstrum
  - band aperiodicity parameters
- We've seen how to reconstruct the waveform



- Now we can insert a **statistical model** between the analysis and synthesis parts

Figures: Hideki Kawahara

What next?



Figures: Hideki Kawahara


PAUSE/STILL

Contents

- The big picture
  - text-to-speech, viewed as regression ✔
- Getting ready for regression
  - feature extraction from text ✔
  - feature extraction from speech ✔
- Doing regression
  - **using a decision tree: so-called "HMM-based TTS"**
  - using more powerful and general regression models: neural networks } My next lecture

What you should already know

- phonemes
  - place, manner, voicing, etc
- front-end text processing
  - linguistic specification

What you should already know

- phonemes
  - place, manner, voicing, etc
- front-end text processing
  - linguistic specification

What you should already know

- <u>phonemes</u>
  - place, manner, voicing, etc

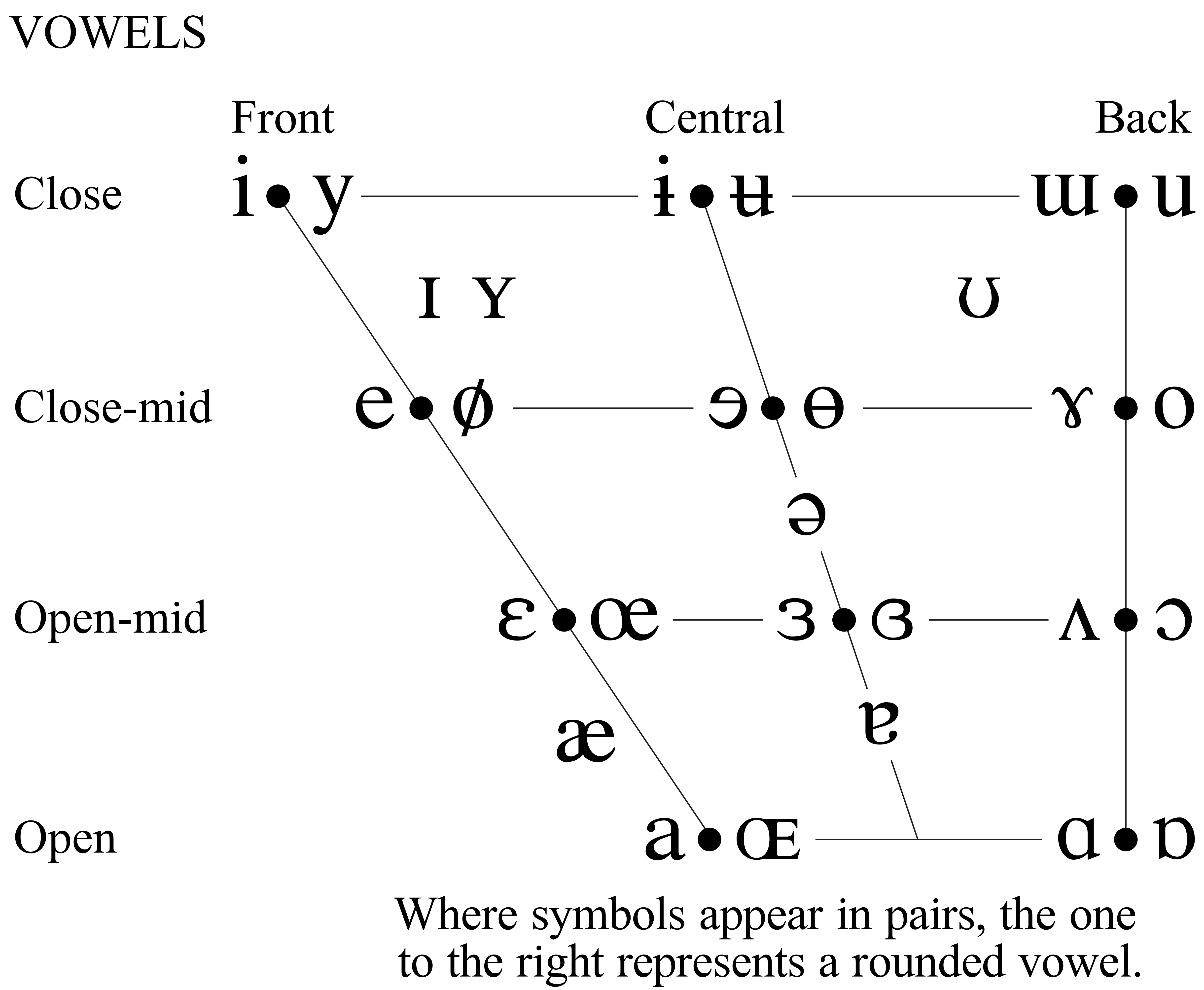
CONSONANTS (PULMONIC)

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k ɡ	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible.

What you should already know

- phonemes
  - place, manner, voicing, etc



What you should already know

- phonemes
  - place, manner, voicing, etc
- <u>front-end text processing</u>
  - linguistic specification

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

phoneme: ax
left context: sil dh
right context: k ae
position in phrase: initial
syllable stress: unstressed
etc....

Remember this problem? We haven't actually solved it yet…

phrase initial

pitch accent

phrase final

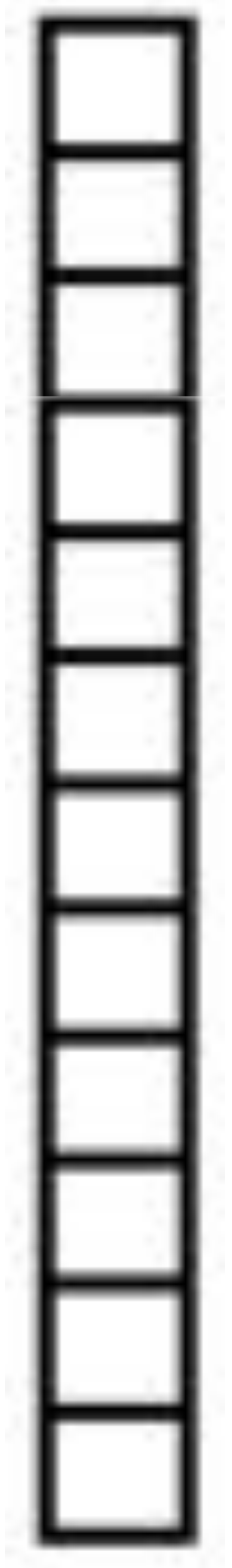
sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...



input feature vector

First solution: convert rich linguistic **structure** into a linear **sequence**

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

phoneme: ax
left context: sil dh
right context: k ae
position in phrase: initial
syllable stress: unstressed
etc....

“Flatten” the linguistic structure, by attaching contextual features to phones

phrase initial
pitch accent
phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

Orientation

- <u>So far</u>:
  - extract rich linguistic features from text using the **front end** (same as in unit selection)
  - flatten those features into a sequence of **context-dependent phones**
- <u>Next</u>:
  - create a **statistical model** for every possible context-dependent phone
  - **train** the model on data
  - use it to **synthesise** new sentences

Our first model: regression tree + Hidden Markov Model

- Two complementary explanations
  - regression
  - context-dependent models
- Duration modelling
- Generation from the model

Two complementary explanations

regression

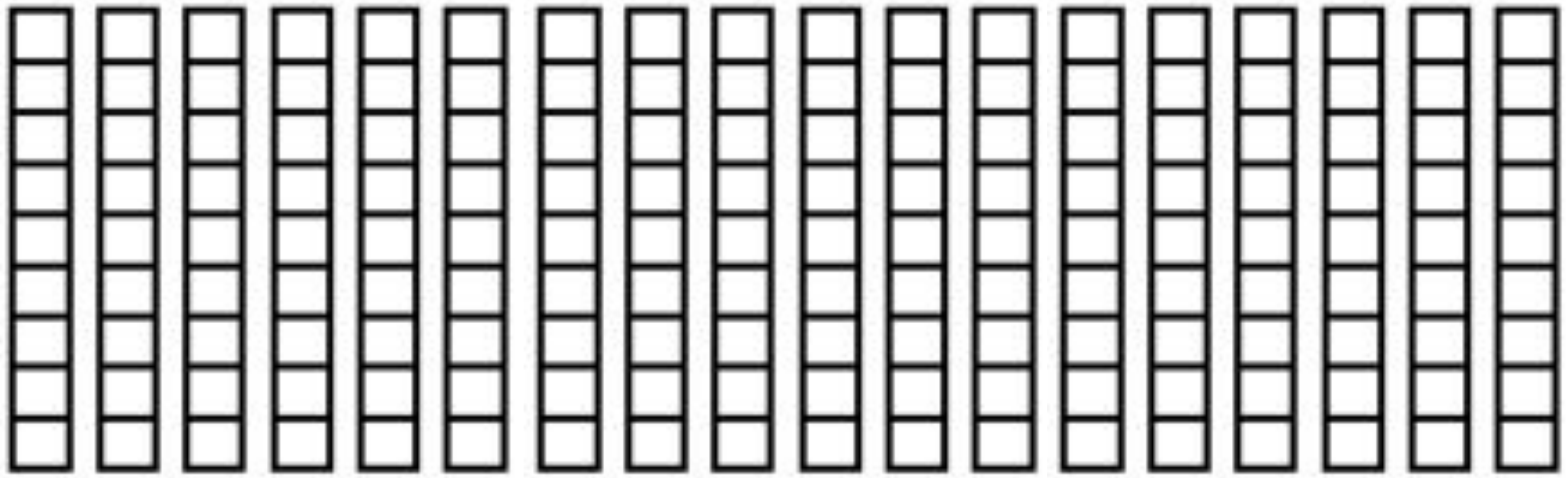
- Describing synthesis as a regression task
    - **prediction** of continuous speech parameters from linguistic features

context-dependent modelling

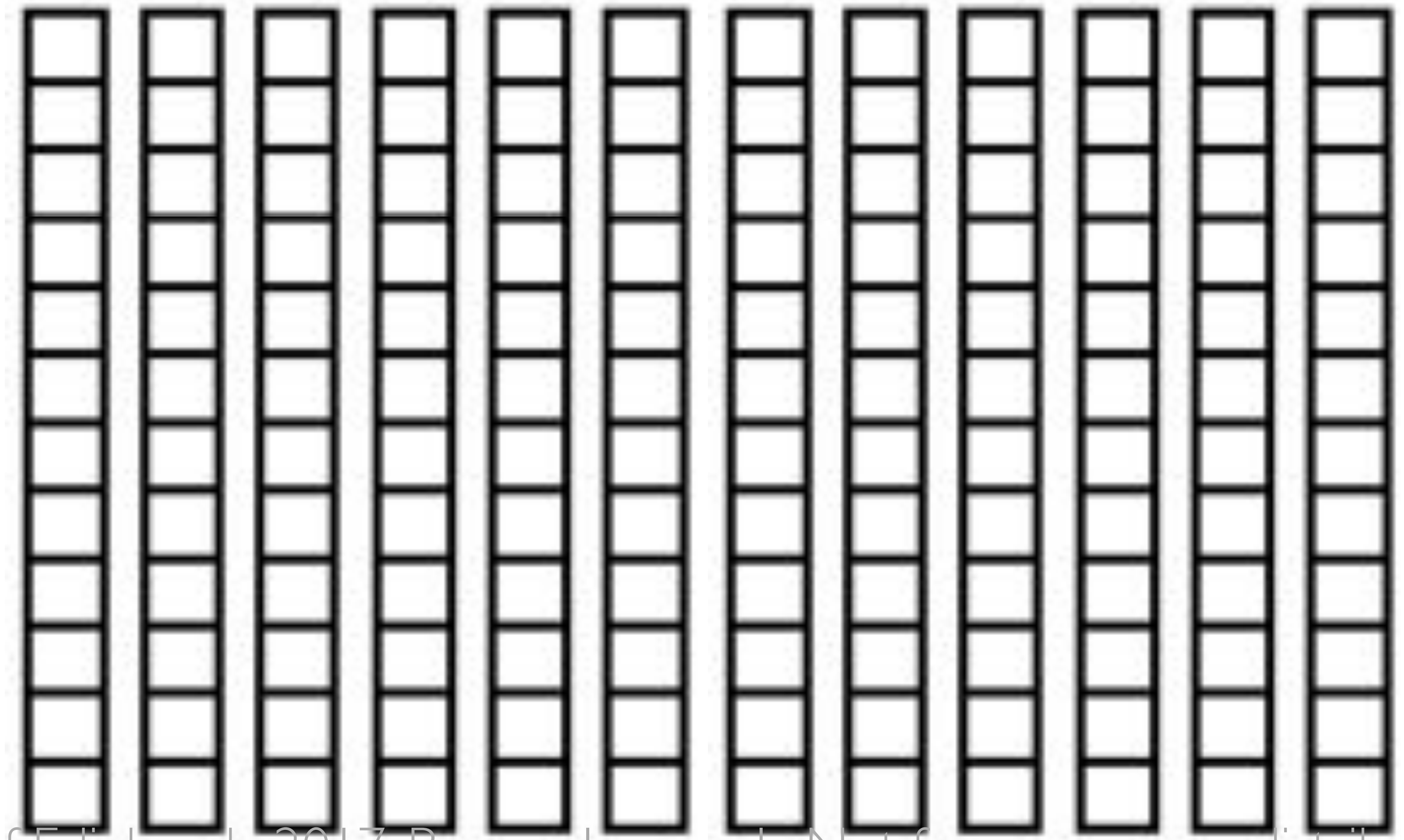
- Practical implementation using context-dependent models
    - **create** *lots* of models: oops! for many, there is **no training data**
    - fix this by **sharing** parameters with existing models ("tying")

Sequence-to-sequence regression = alignment + frame-to-frame regression

output sequence
(speech features)

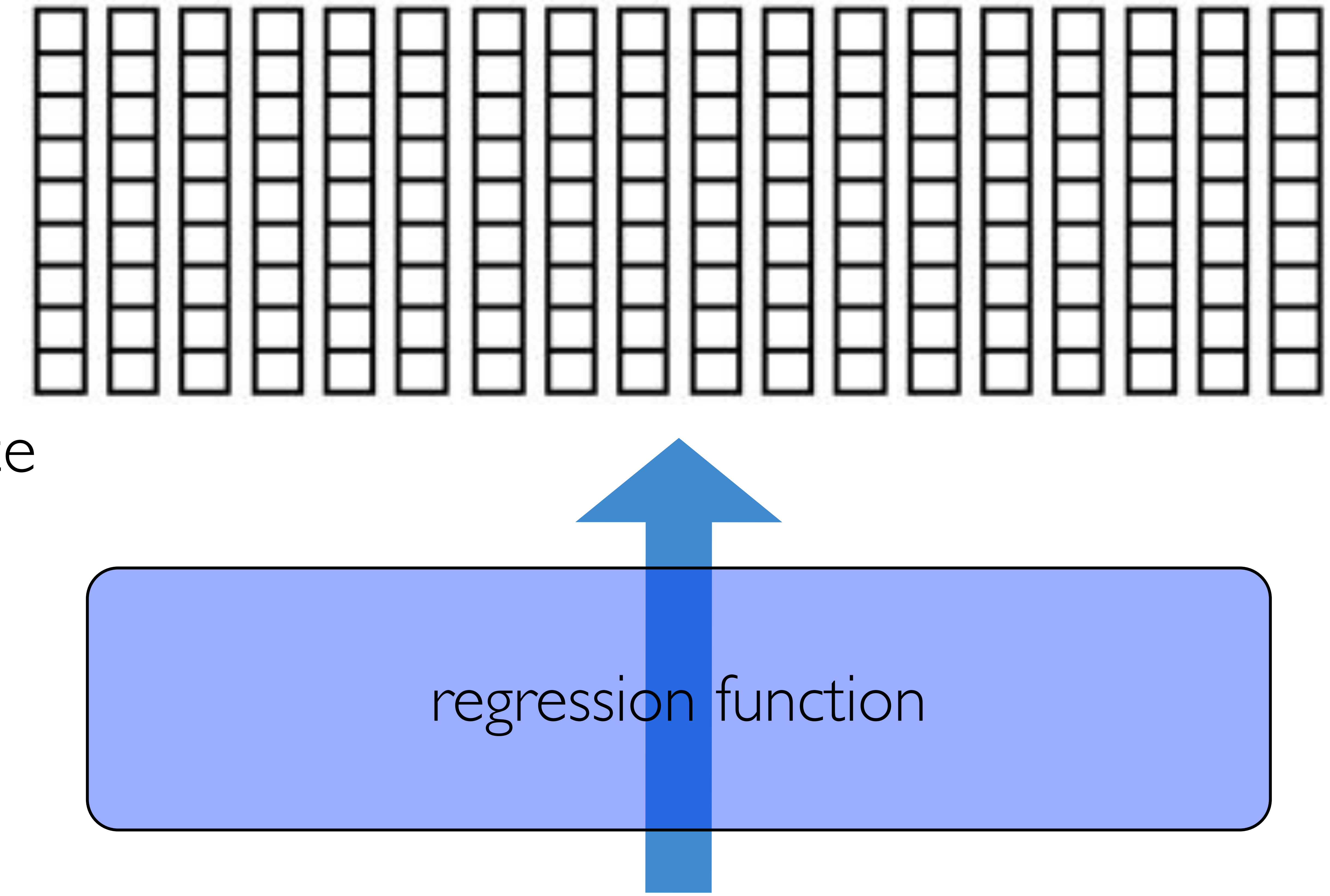


input sequence
(linguistic specification)



Two tasks to accomplish

- Sequencing
  - progress through the phonetic sequence
  - decide durations
  - create a sequence of frames
- Prediction (regression)
  - Given the local linguistic specification, predict one frame of speech parameters

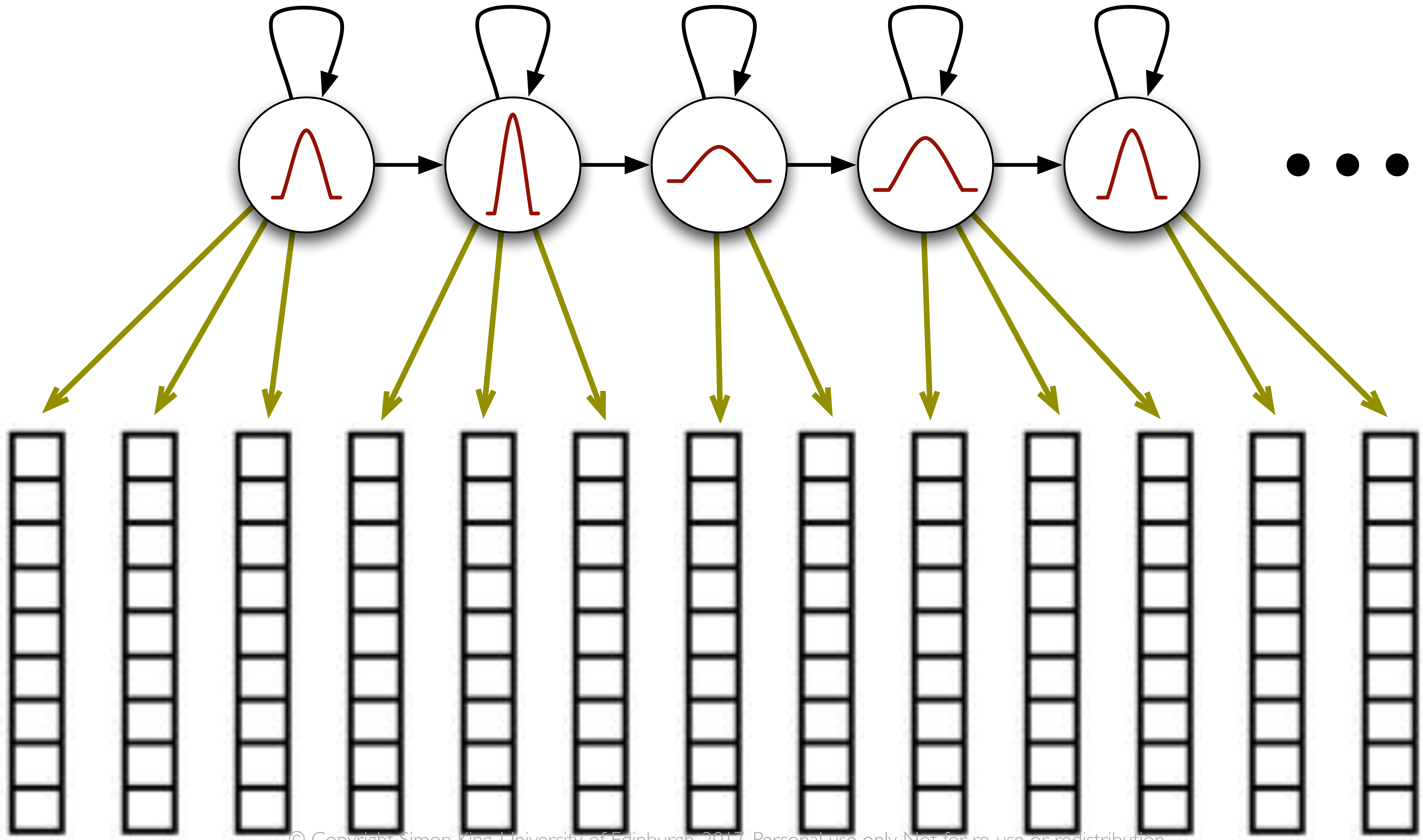



```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

Choose suitable machinery for each task

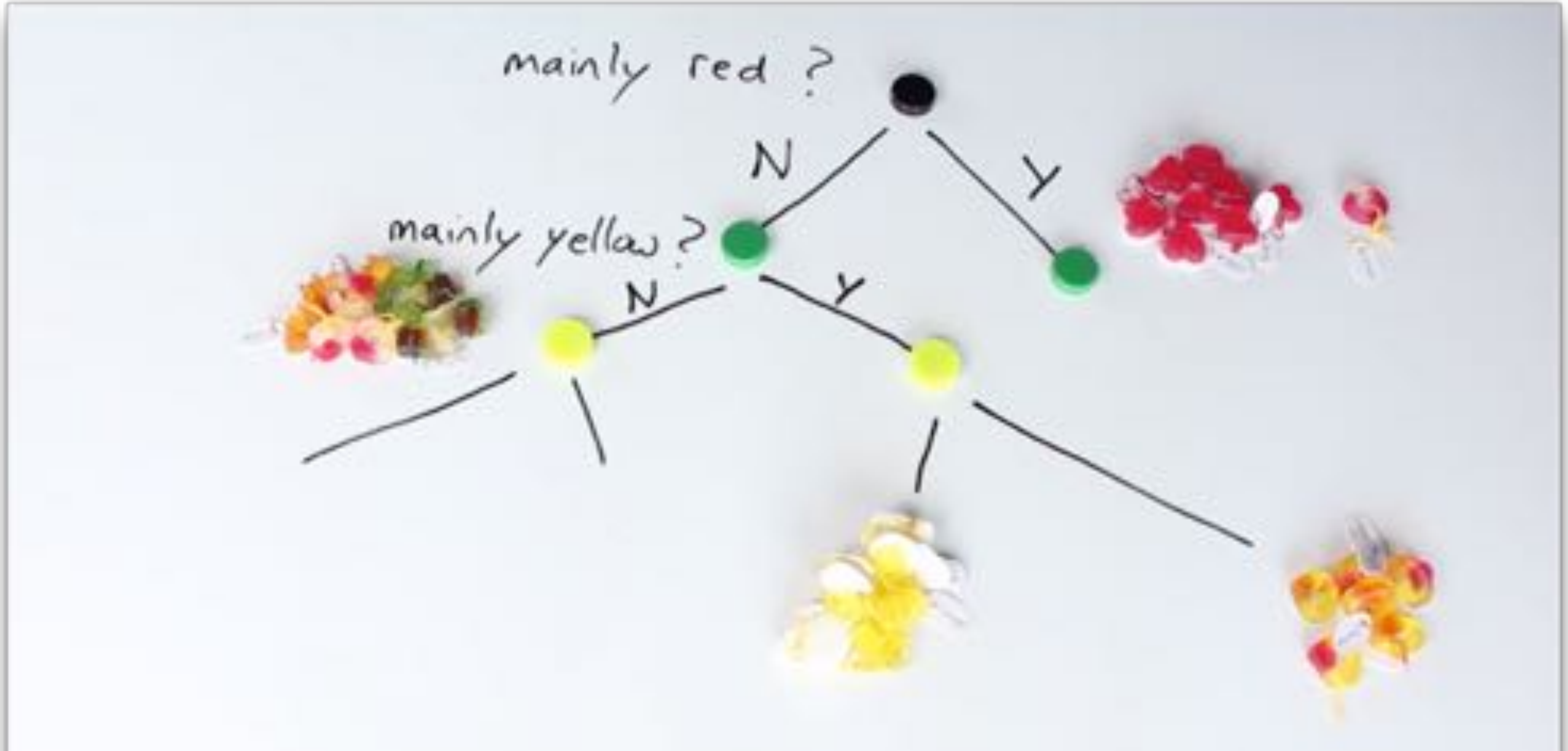
- Sequencing
  - **Hidden Markov Model**
  - Why? It's the simplest model we know that can generate sequences!

- Regression
  - **Regression tree** (i.e., a CART with continuously-valued predictee)
  - Why? Again, the simplest model we know, that can learn an arbitrary function
    - *the mapping from linguistic specification to speech spectrum is surely non-linear*



Regression

CART (Classification and Regression Tree) - see speech.zone



HMM for sequencing + **regression tree** for prediction

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

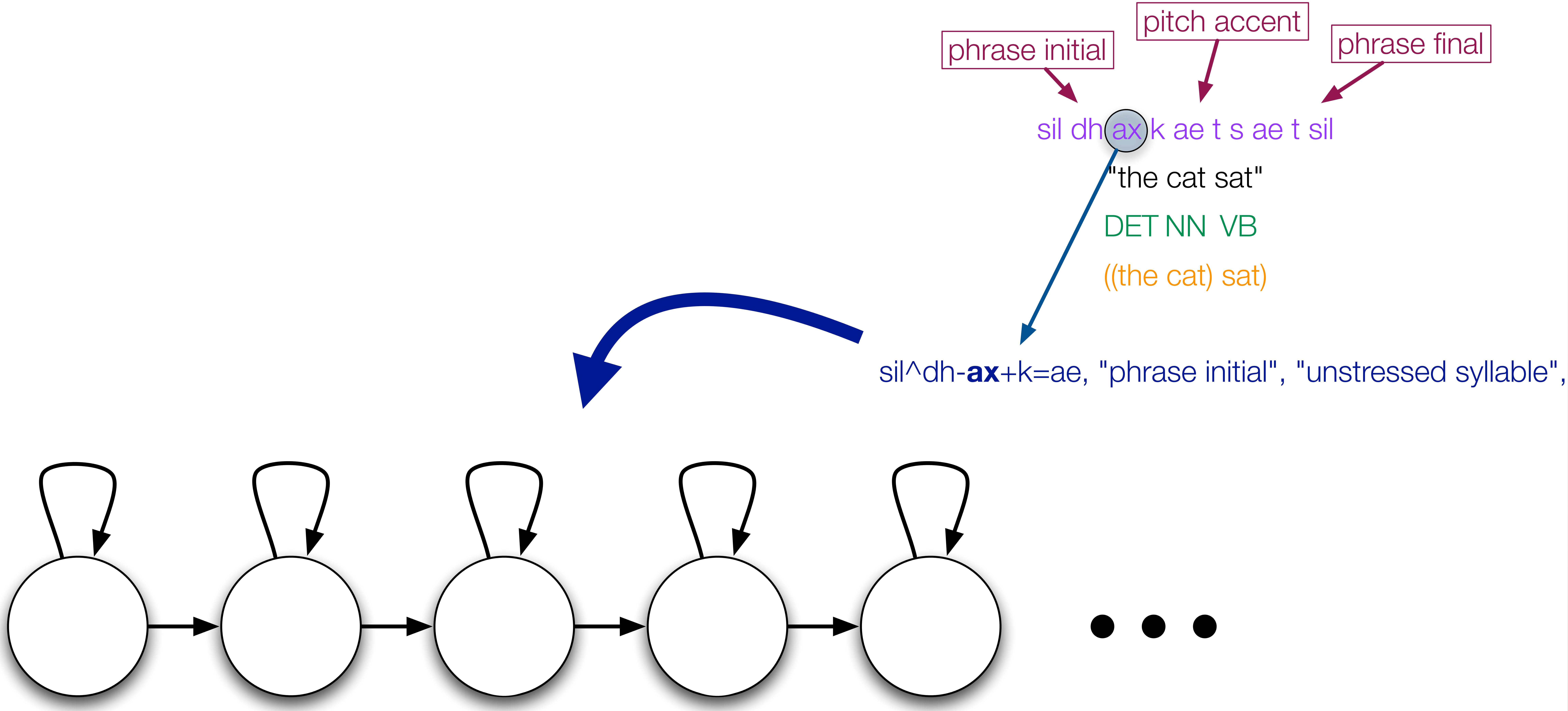
"the cat sat"

DET NN VB

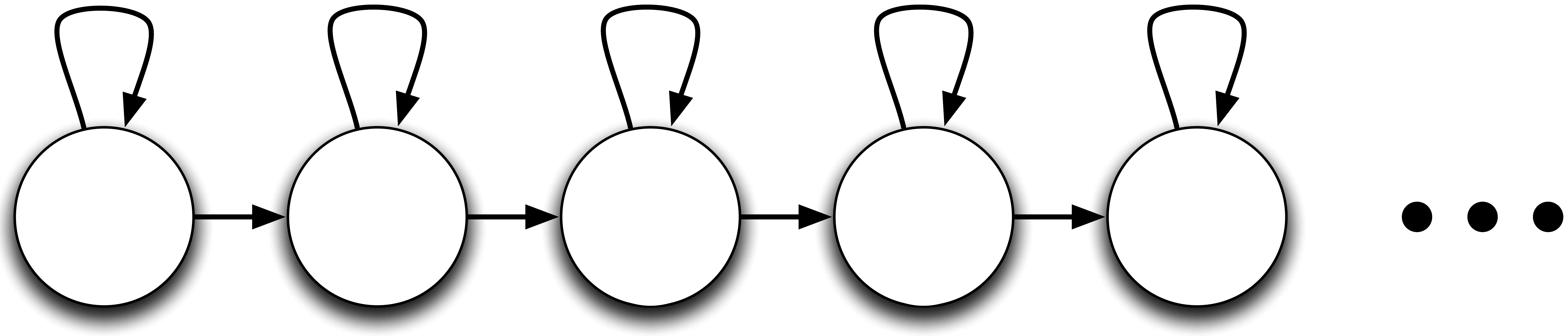
((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

HMM for sequencing + **regression tree** for prediction



HMM for sequencing + **regression tree** for prediction



sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

Two complementary explanations

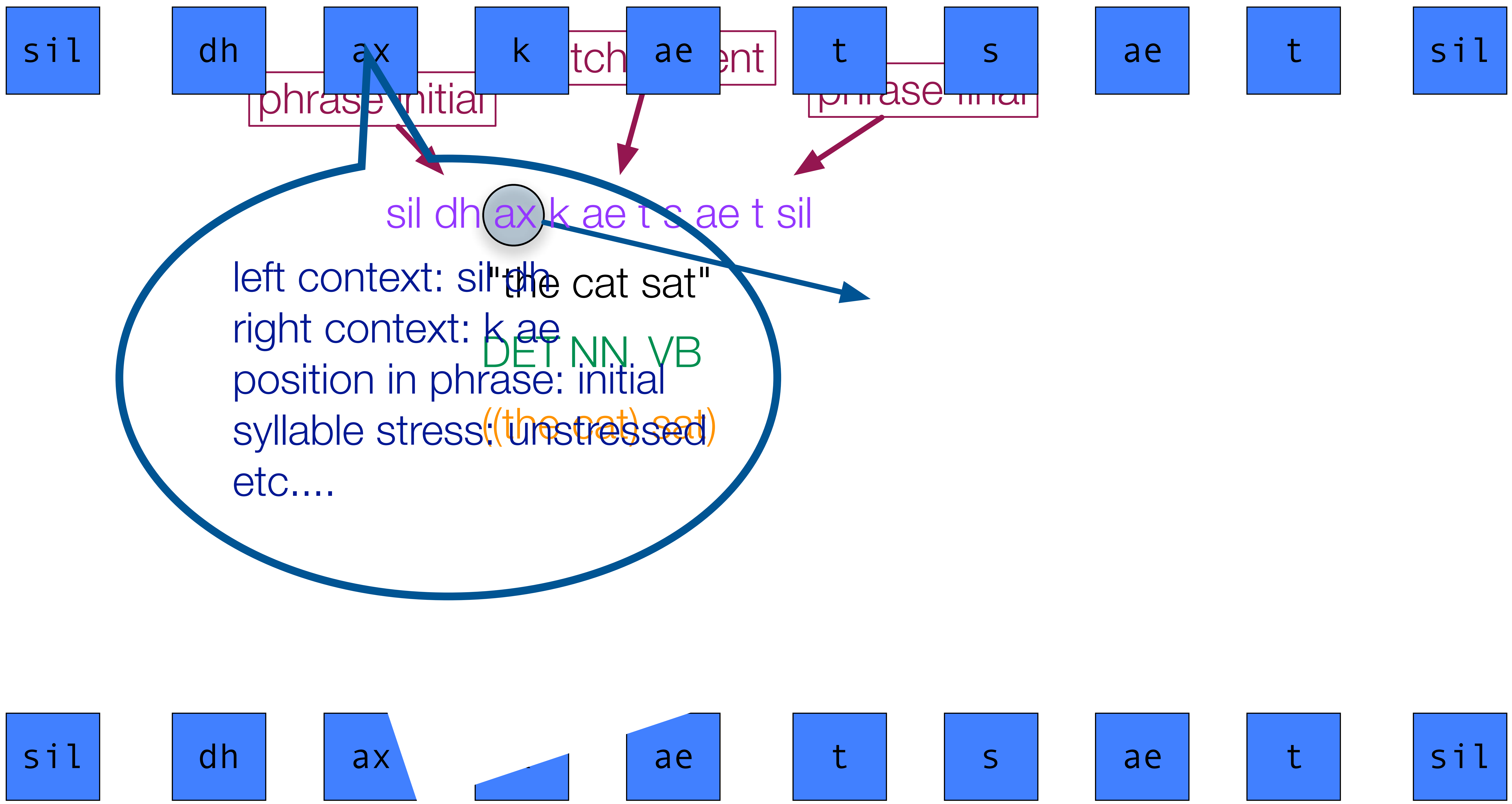
regression

- Describing synthesis as a regression task
  - **prediction** of continuous speech parameters from linguistic features

context-dependent modelling

- Practical implementation using context-dependent models
  - **create** *lots* of models: oops! for many, there is **no training data**
  - fix this by **sharing** parameters with existing models ("tying")

Reminder: constructing the target unit sequence (for unit selection)



From linguistic specification to sequence of models

"Author of the ..."

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$.....
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$.....
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$.....
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$.....
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$.....
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$.....
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$.....
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$.....
```

Context-dependent modelling

- We cannot be sure to have examples of every unit type in every possible context in the training data
- In reality, the context is so rich (it spans the whole sentence), that almost every single token in the training data is the only token of its type

- Two key problems to solve
  - train models for types that we have **too few** examples of (e.g., 1)
  - create models for types that we have **no examples** of

- Joint solution: parameter sharing amongst groups of similar models

Training models for types that we have too few examples of

- We *could* train a model on just a single example (= single token)
- But it will be very poorly estimated
  - unlikely to perform well

- **Pooling training data** across groups of types will increase amount of data available
- How to decide **which groups** of models should share data?
  - i.e., which groups of models will end up with the same parameters

Some contexts exert similar effects

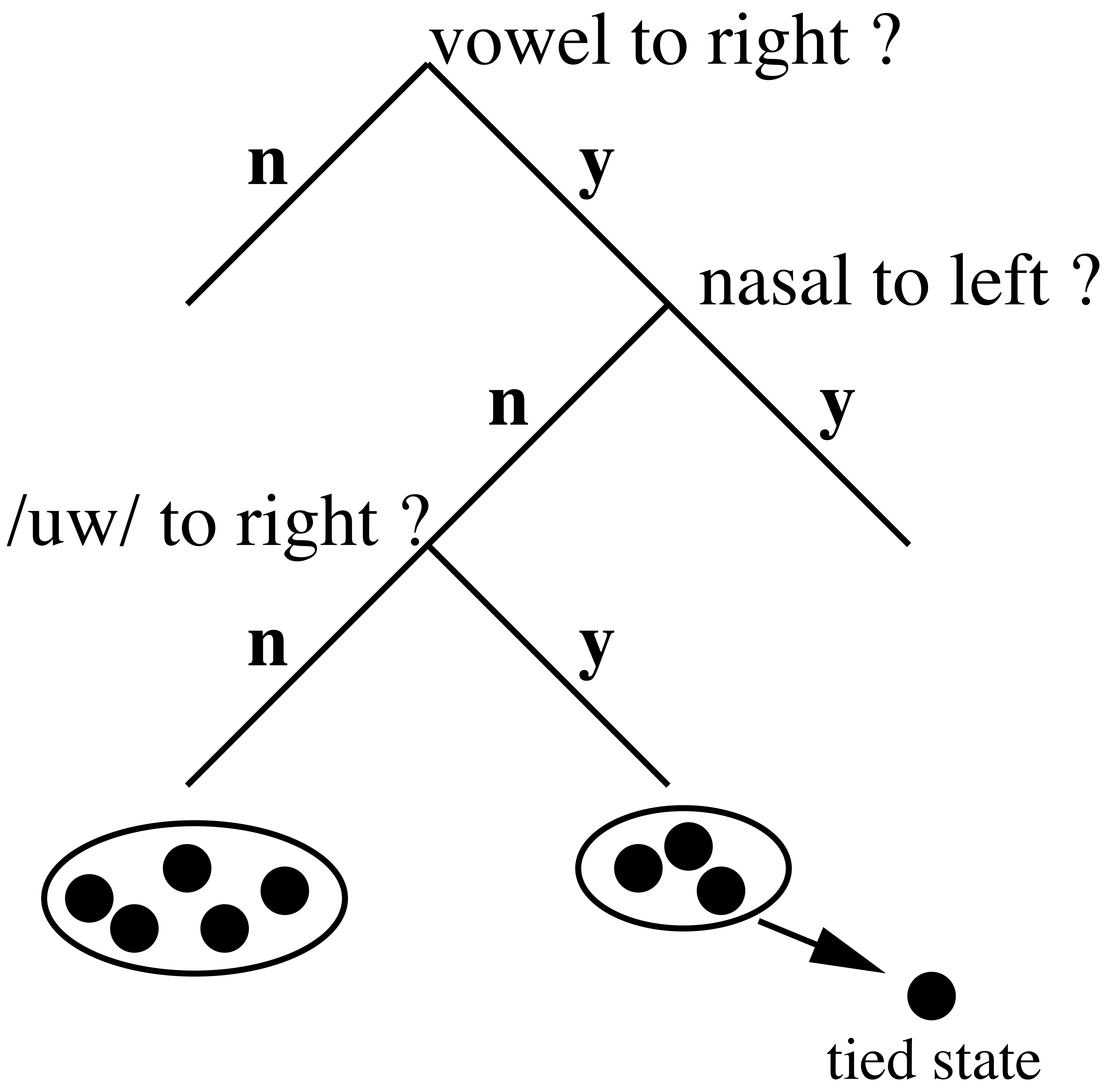
- Key insight
  - we can group *contexts* according to the effect that they have on the centre phoneme
  - for example
    - the [ae] in the contexts p-ae+t and b-ae+t may be very similar
  - how to group these contexts?
    - how to represent them so we can form useful groupings?

  - **use the phonetic features of the surrounding context**
    - place, manner, voicing, ...

Grouping contexts according to phonetic features

- Could try to write rules to express our knowledge of how co-articulation and other context effects work
  - *"all bilabial stops have a similar effect on the following vowel"*
  - *"all nasals have a similar effect on the preceding vowel"*
  - *... etc*
- Of course, it's better to learn this from the data, for 2 reasons
  - find those groupings that actually make **a difference to the acoustics**
  - **adjust the granularity** of the groups according to how much data we have
- But we still want to make use of our **phonetic knowledge**

Combining phonetic knowledge with data-driven learning

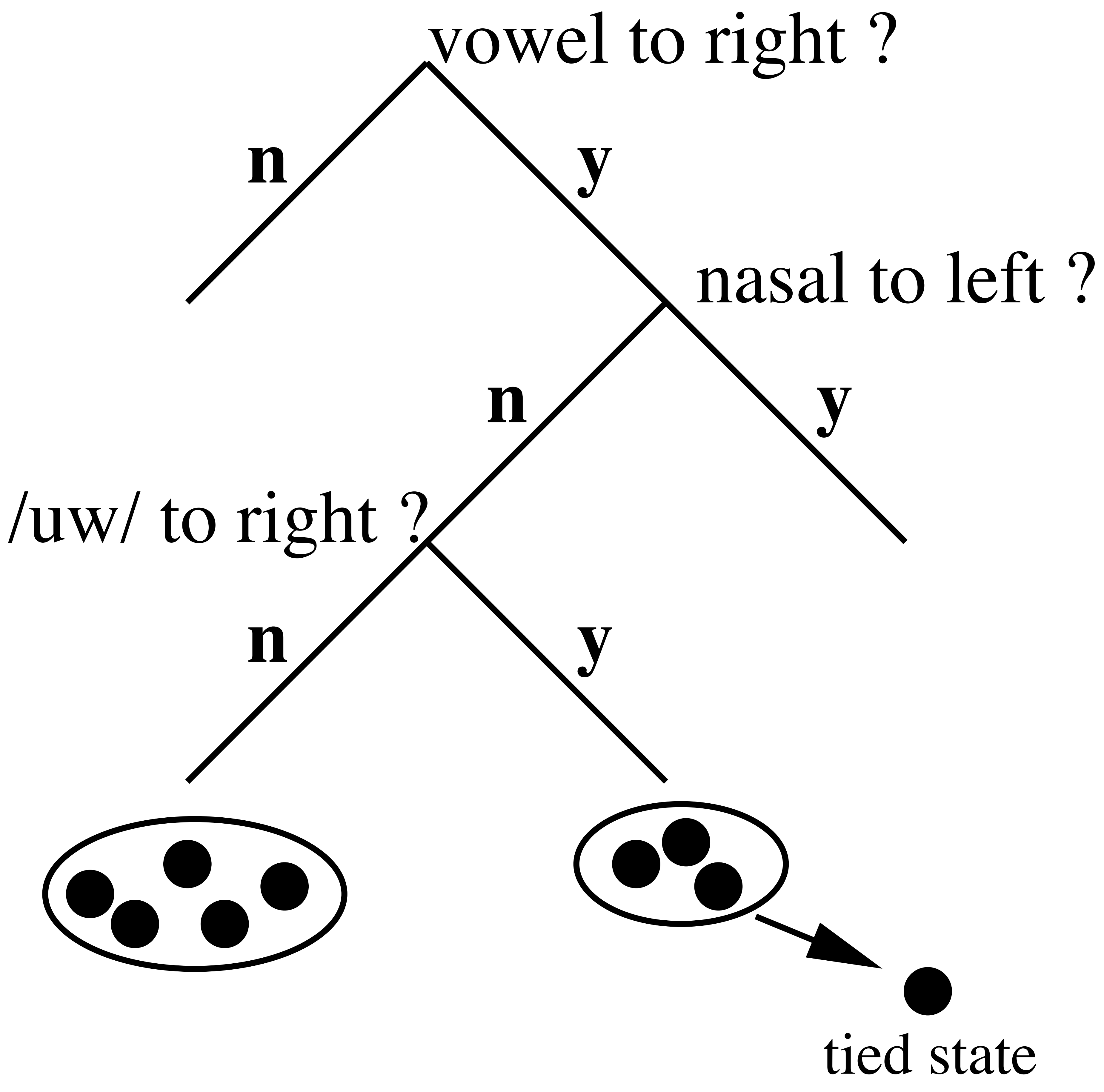


How to choose the best split

- Ideal measure
  - a) train a single model on data pooled across the unsplit set of contexts
  - b) train two models: one on each split of the data
  - compare the **likelihood increase** from a) to b)
- This is not feasible in practice - too computationally-expensive
  - cannot retrain models for every possible split, at every node in the tree
- Instead, use an **approximation** to the likelihood increase
  - this can be computed without actually retraining any models
  - only requires access to the state occupancy statistics and Gaussian parameters

What about models for unseen contexts?

- To find out which model to use for a particular context
  - just follow the tree from root to leaf, answering the questions
- Crucially, to do this we only need to know the **name** of the model, in order to answer those questions
- So it works for models which have training data, and also for models that don't




```
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$.....
```

Summary: linguistic processing, training, synthesis

- Linguistic processing
  - from text to linguistic features using the **front end** (same as in unit selection)
  - attach linguistic features to phonemes: **"flatten"** the linguistic structures

  - we then create one context-dependent HMM for **every unique combination** of linguistic features

Summary: linguistic processing, training, synthesis

- Training the HMMs
  - need **labelled** speech data, just as for ASR (supervised learning)
  - need models for all combinations of linguistic features, including those **unseen** in the training data
    - this is achieved by parameterising the models using a regression tree

Summary: linguistic processing, training, synthesis

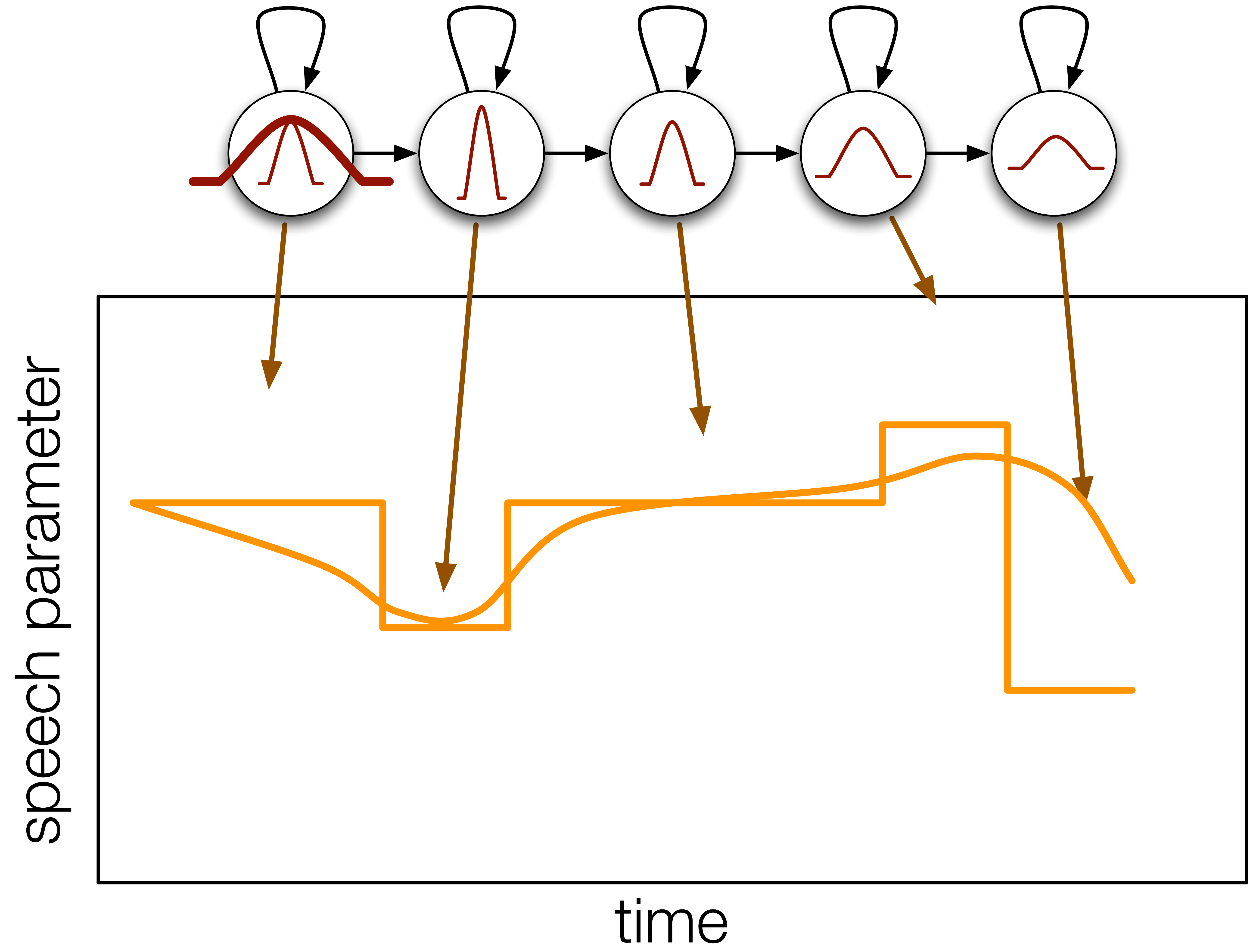
- Synthesising from the HMMs
  - use the front end to predict required **sequence** of context-dependent models
    - the regression tree provides the **parameters** for these models
  - use those models to **generate** speech parameters
  - use a **vocoder** to convert those to a waveform

Generating from the regression tree + Hidden Markov Model

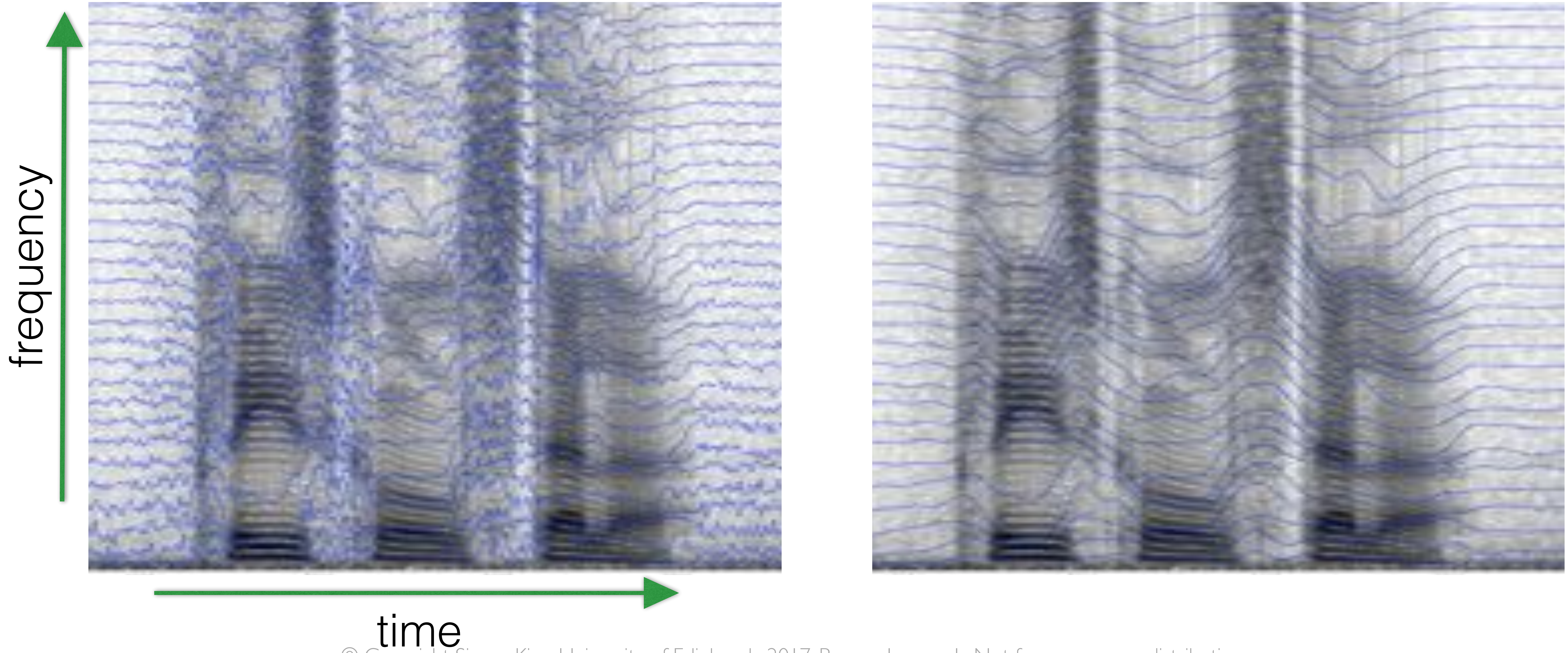
- This is straightforward, because the HMM is a generative model
- Follow the Maximum Likelihood principle
    - generate the **most likely** output
    - that will simply be the sequence of state **means**

- What about duration?
    - we need a model to predict this
    - let's just use another regression tree, predicting duration **per state**
        - <u>predictors</u>: linguistic context + state-position-within-phone
        - <u>predictee</u>: duration of the current state, in frames

Trajectory generation



LSPs extracted from waveform vs. generated by HMM




8000
7000
6000
5000
4000
3000
2000
1000
0
Hz
0
100
200
300
400
500
600
700
800
900
1000
Frame


0
0
0

"Over smoothing"

- Generated trajectories are temporally smoother than natural ones
  - **fine detail** is lost - this is actually not a problem (*it was probably just analysis error*)
  - **deviation from the mean** is reduced - this is a significant problem

- Standard solution: scale the standard deviation (or variance) back up to global natural level
  - *Global Variance (GV)*
  - *simple scaling*

"Over smoothing"

- Generated spectral envelope is smoother in <u>frequency domain</u> than natural one
  - formant (resonance) peaks are wider and less sharp - giving a '**muffled**' sound
  - reduced resonance reveals the '**buzzy**' nature of the artificial source signal

- Standard solution: enhance the spectral sharpness
  - *raise spectrum to a power greater than* 1
  - …*or one of many other solutions*

Orientation

- Our **first attempt** at statistical parametric speech synthesis
  - we used models that we are familiar with and understand well

- Gaussians are convenient
  - e.g., so we can borrow many useful techniques from ASR

- But regression trees are weak models

Orientation

- Our **first attempt** at statistical parametric speech synthesis
  - we used models that we are familiar with and understand well

This is perfectly sensible: we have **good algorithms** for training the models, for example.

- Gaussians are convenient
  - e.g., so we can borrow many useful techniques from ASR

e.g., model adaptation

- But regression trees are weak models

The key weakness of the method. We must replace the regression tree with something more powerful.

What next?

- **Better regression model**
  - a Neural Network
  - input & output features essentially the same as regression tree + HMM

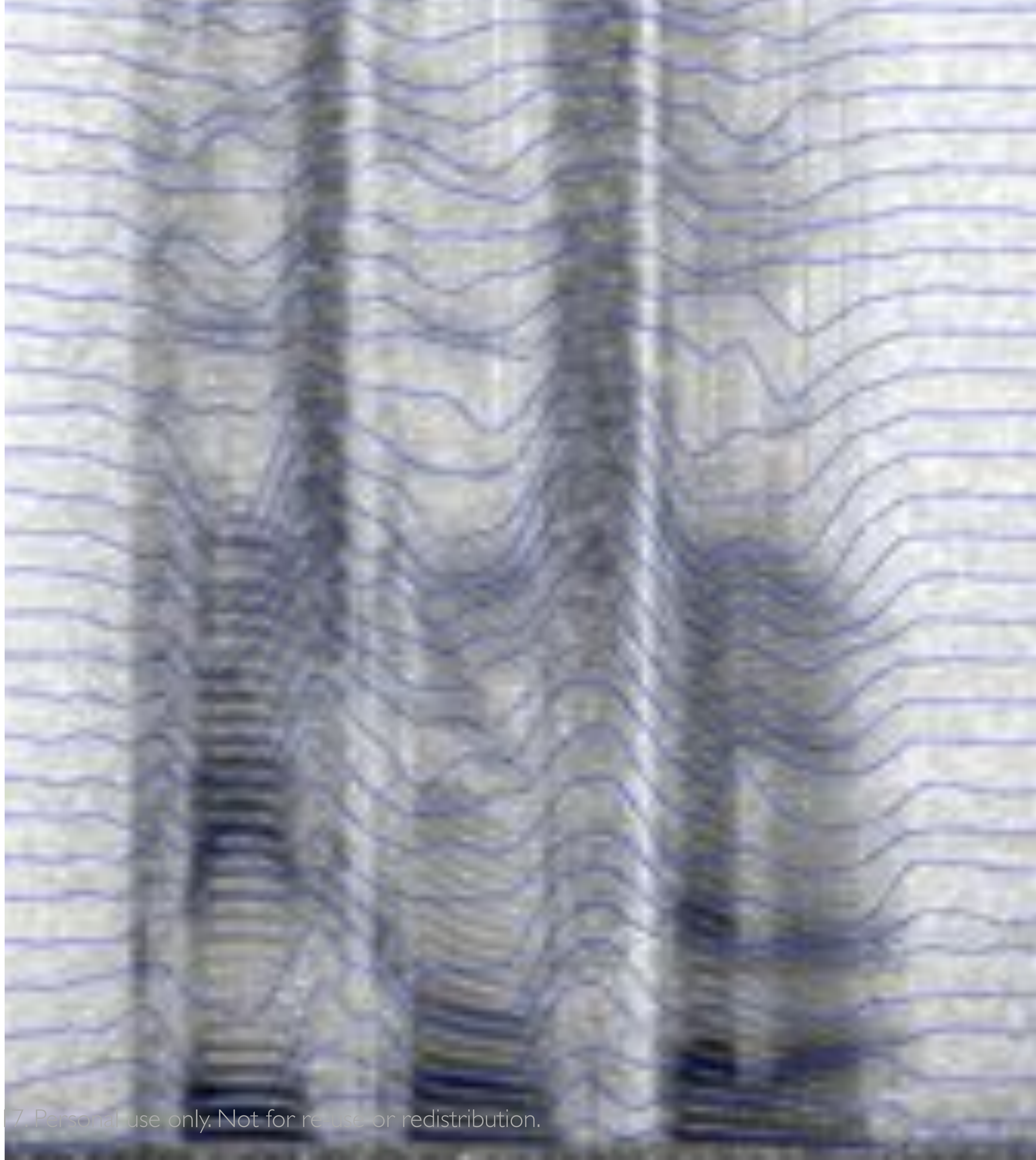
- Quality will still be limited by the **vocoder**

- Later, we will also address that problem
  - *hybrid synthesis (not in this course)*
  - direct waveform generation


PLAY
PAUSE/STILL

Statistical Parametric Speech Synthesis
- HMM-based (2 hours)
- <u>DNN-based (1 hour)</u>

Simon King
University of Edinburgh



Contents

- The big picture
  - text-to-speech, viewed as regression ✔
- Getting ready for regression
  - feature extraction from text ✔
  - feature extraction from speech ✔
- Doing regression
  - using a decision tree: so-called "HMM-based TTS" ✔
  - **using more powerful and general regression models: neural networks**

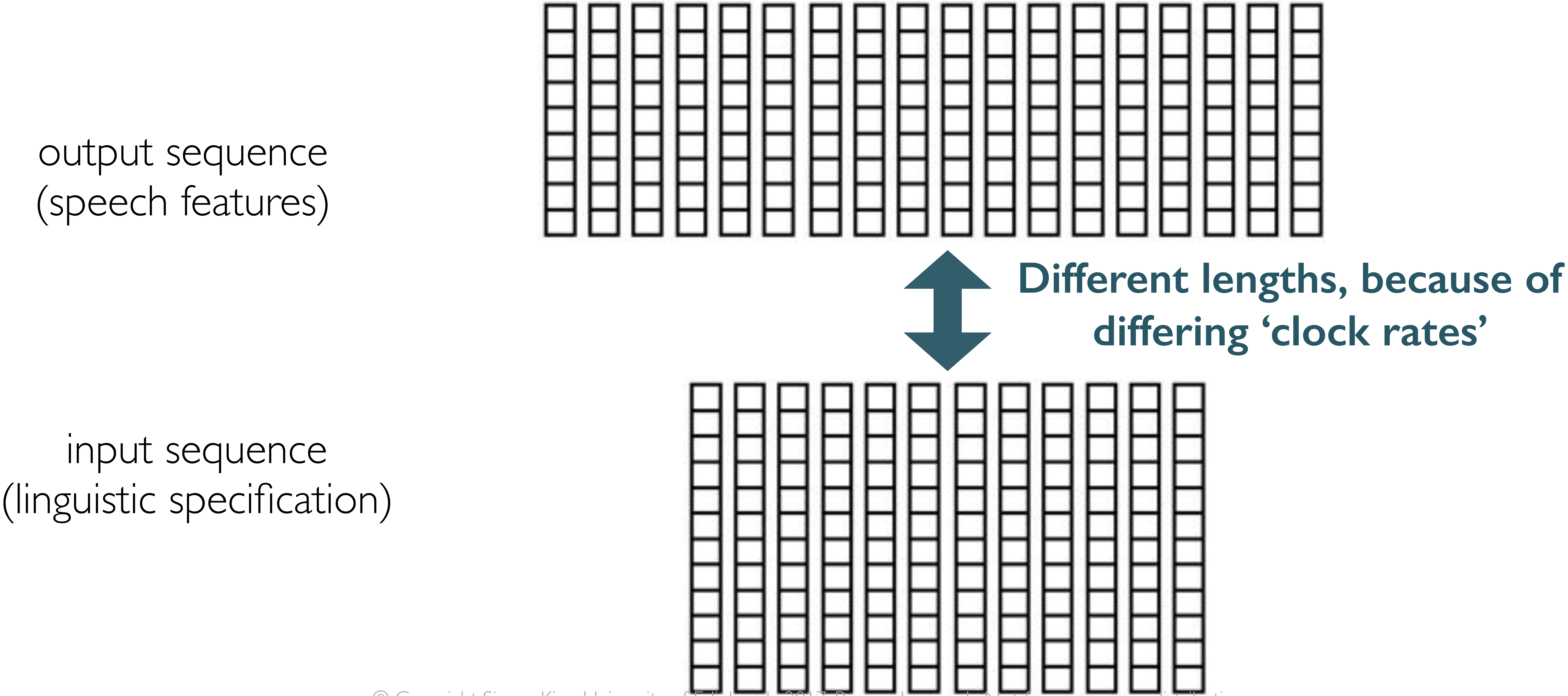
Speech synthesis using Neural Networks

- preparing the input features
- what is a Neural Network?
- generating speech with a Neural Network
- training a Neural Network

Speech synthesis using Neural Networks

- preparing the input features
- what is a Neural Network?
- generating speech with a Neural Network
- training a Neural Network

We've described the problem as **sequence-to-sequence regression**



Remember this problem? Now we really have to solve it!

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

input feature vector

Preparing the input features for Neural Network speech synthesis
1) flatten the linguistic structure, to create a linear sequence (as for HMMs)

phrase initial

pitch accent

phrase final

sil dh ax k ae t s ae t sil

"the cat sat"

DET NN VB

((the cat) sat)

sil^dh-**ax**+k=ae, "phrase initial", "unstressed syllable", ...

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

Preparing the input features for Neural Network speech synthesis
2) encode and upsample

linguistic timescale → fixed framerate

predict durations

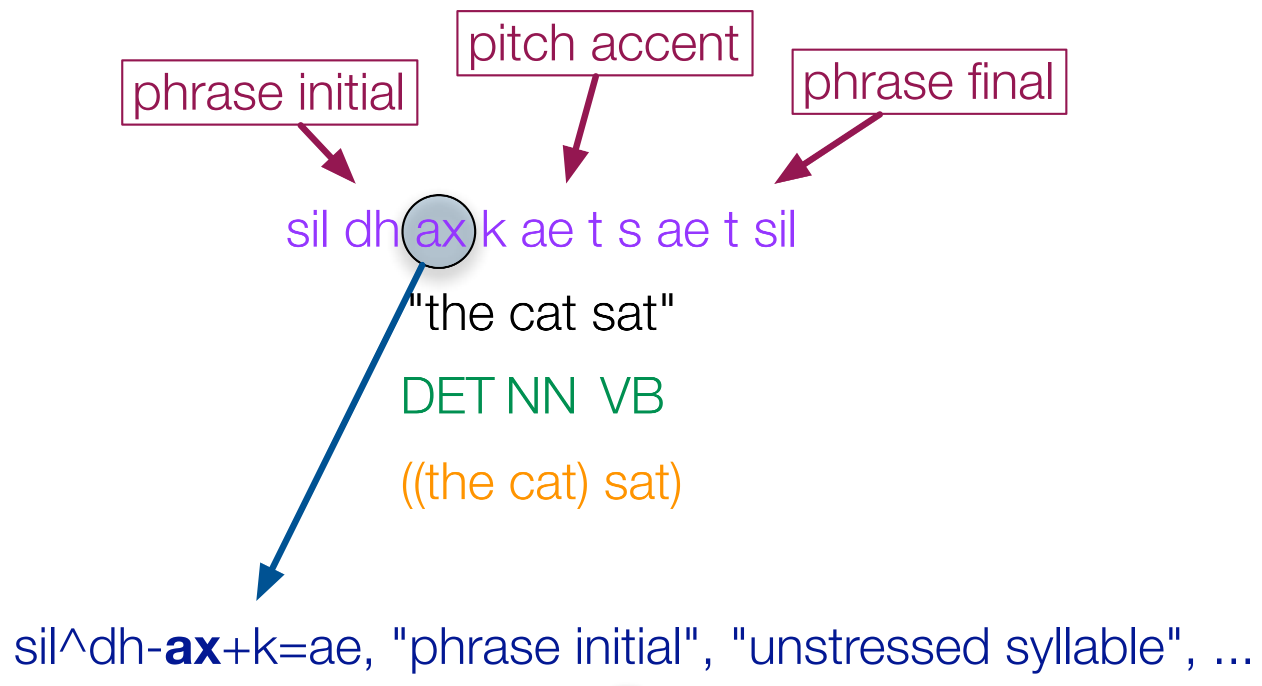
```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

How to construct the sequence of input features

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

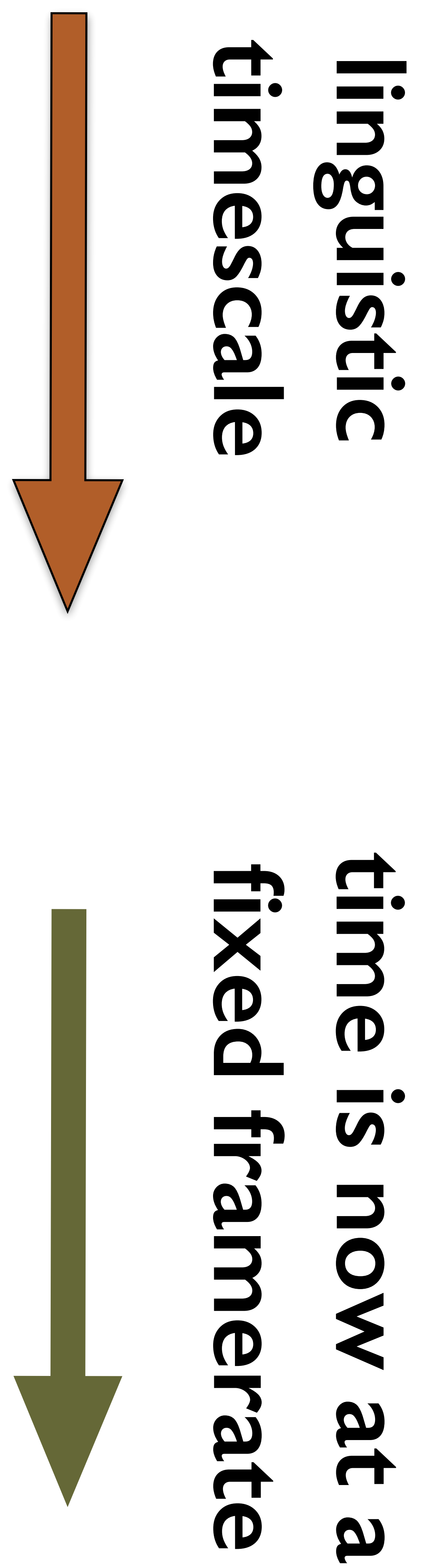
Preparing the input




```
sil~sil-sil+dh=ax@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-dh+ax=k@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~dh-ax+k=ae@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
dh~ax-k+ae=t@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
ax~k-ae+t=s@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
k~ae-t+s=ae@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ae~t-s+ae=t@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
t~s-ae+t=sil@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

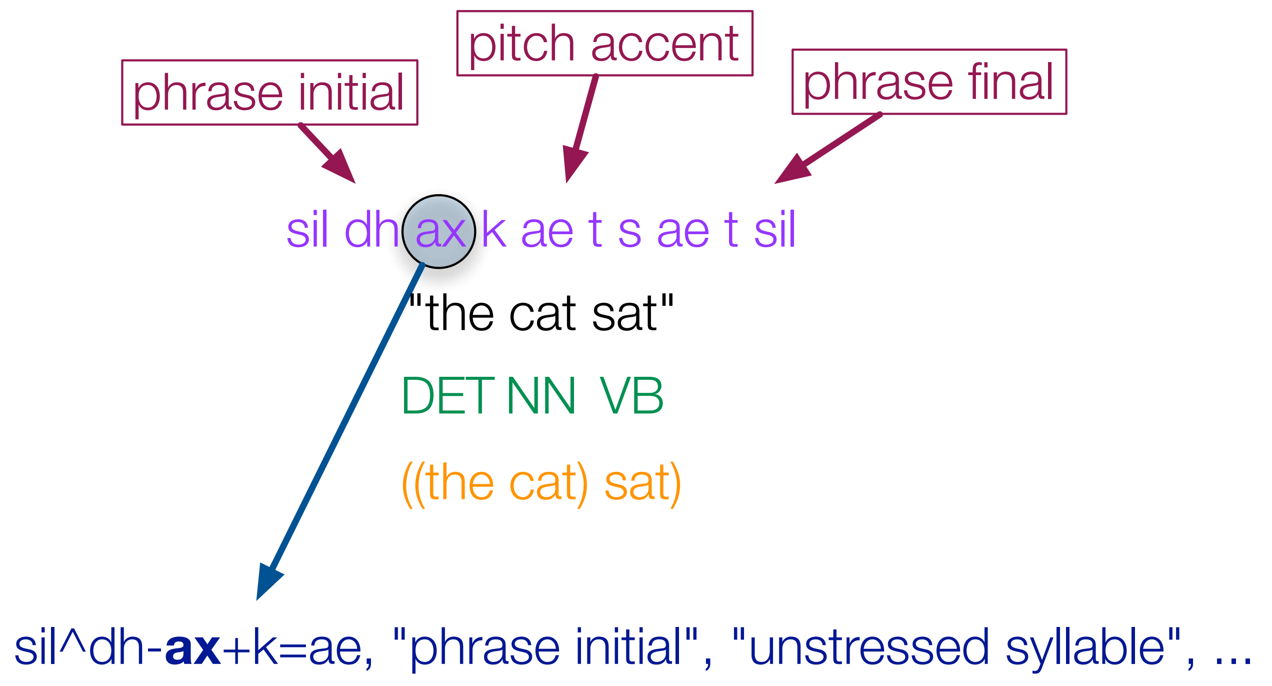
```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

- Run the front end
  - obtain linguistic specification



Preparing the input:
flatten linguistic specification

linguistic timescale: phones



Preparing the input:
a sequence of context-dependent phones

linguistic timescale: phones

“Please call . . .”

```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

quinphone

positional features
(e.g., position of phone in syllable)

POS features

This is the sequence of model names that we would use in HMM-based speech synthesis

Preparing the input:
predict durations at the subphone level

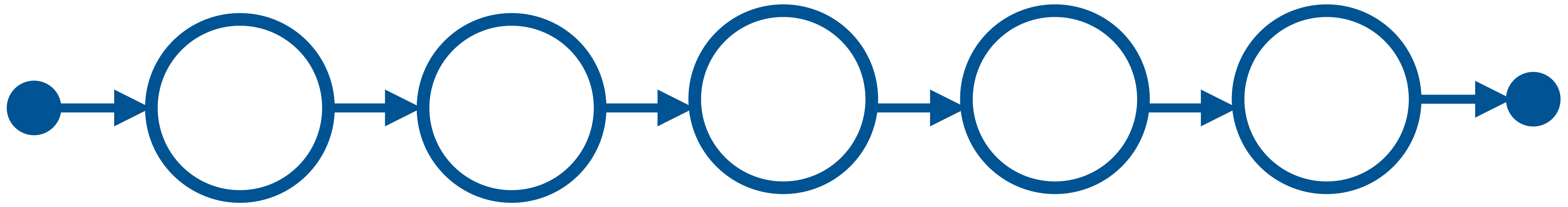
linguistic timescale: subphones

```
"Please call . . ."

#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

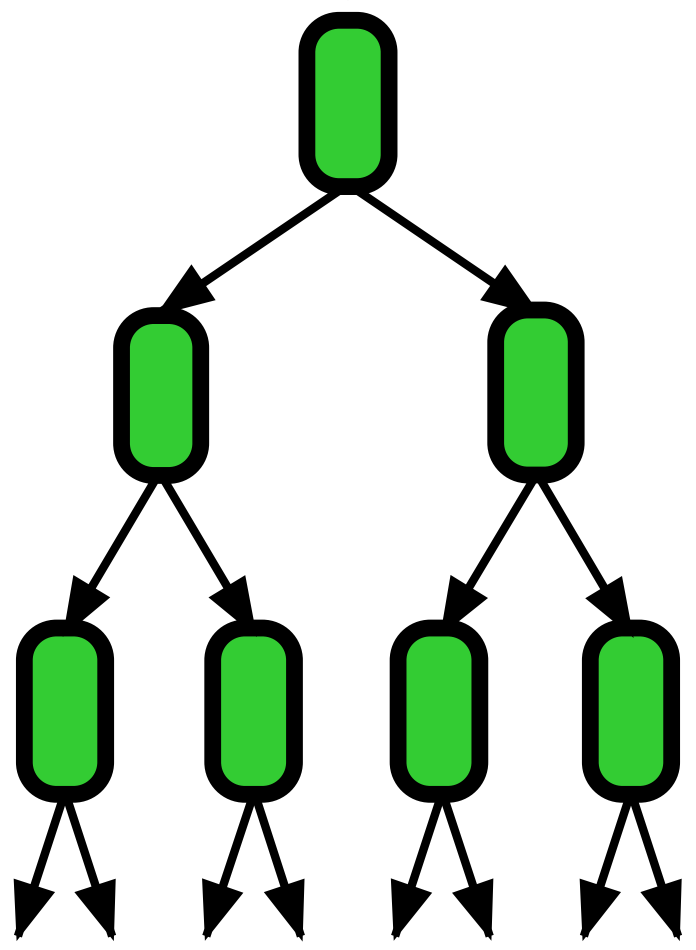
What is the "subphone"?

- All early DNN systems employ HMMs as a sub-phonetic "clock"
  - duration is then modelled at the **state** (i.e, subphone) level

`#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .`



duration (in frames) 2 1 3 1 3



regression tree duration model

Preparing the input: predict durations at the subphone level

linguistic timescale: subphones

"Please call . . ."

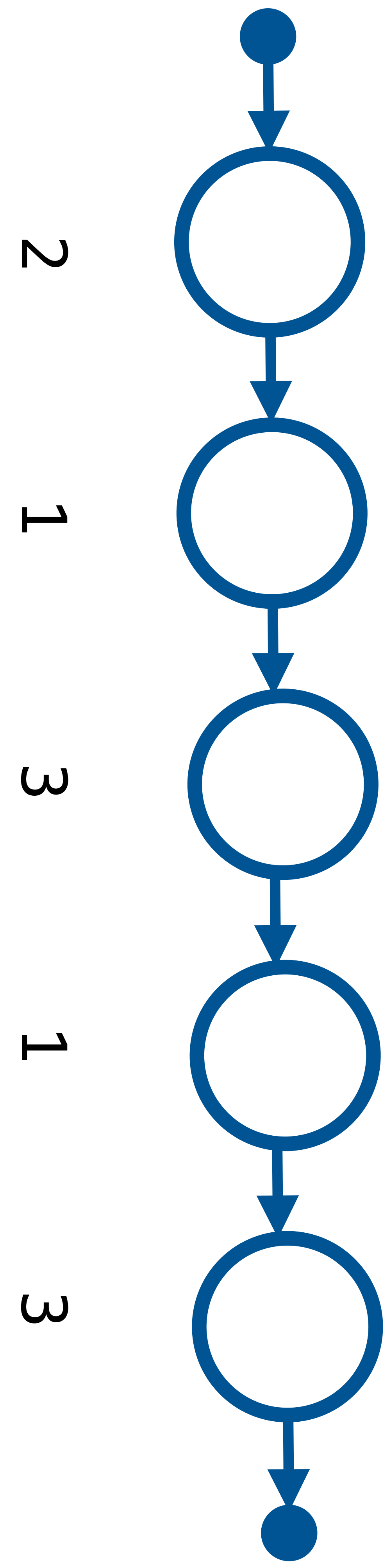
```
#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
                 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1_3/A/1_1_4/B/1-1-3:1-1&2-3# . . .
z~k-O+lw=s:2_2/A/1_1_4/B/1-1-3:1-1&2-3# . . .
```

Preparing the input: convert each state of each context-dependent phone to a vector of binary features

```
"Please call . . ."

#~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&1-4# . . .
 3900000 4000000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&
 4000000 4050000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&
 4050000 4200000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&
 4200000 4250000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&
 4250000 4400000 #~p-l+i=z:2_3/A/0_0_0/B/1-1-4:1-1&
p~l-i+z=k:3_2/A/0_0_0/B/1-1-4:1-1&1-4# . . .
l~i-z+k=O:4_1/A/0_0_0/B/1-1-4:1-1&1-4# . . .
i~z-k+O=lw:1 3/A/1 1 4/B/1-1-3:1-1&2-3# . . .
z~k-O-
```

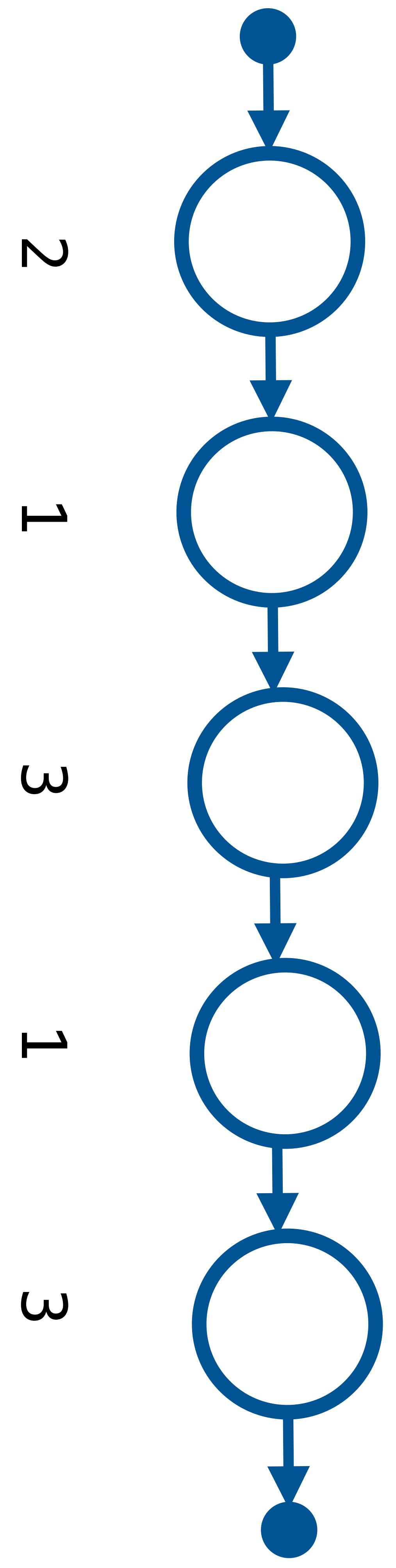
```
QS "C-OI" {-OI+}
QS "C-i" {-i+}
QS "C-aU" {-aU+}
QS "C-aI" {-aI+}
QS "C-a" {-a+}
QS "C-Q" {-Q+}
QS "C-@@" {-@@+}
QS "C-I@" {-I@+}
QS "C-U@" {-U@+}
QS "C-E@" {-E@+}
QS "C-E" {-E+}
QS "C-A" {-A+}
```

Position-within-phone and *position-within-state* features



`000001000001000100000 . . .`

Position-within-phone = state counter



00000100000100010000 . . . 2

00000100000100010000 . . . 3

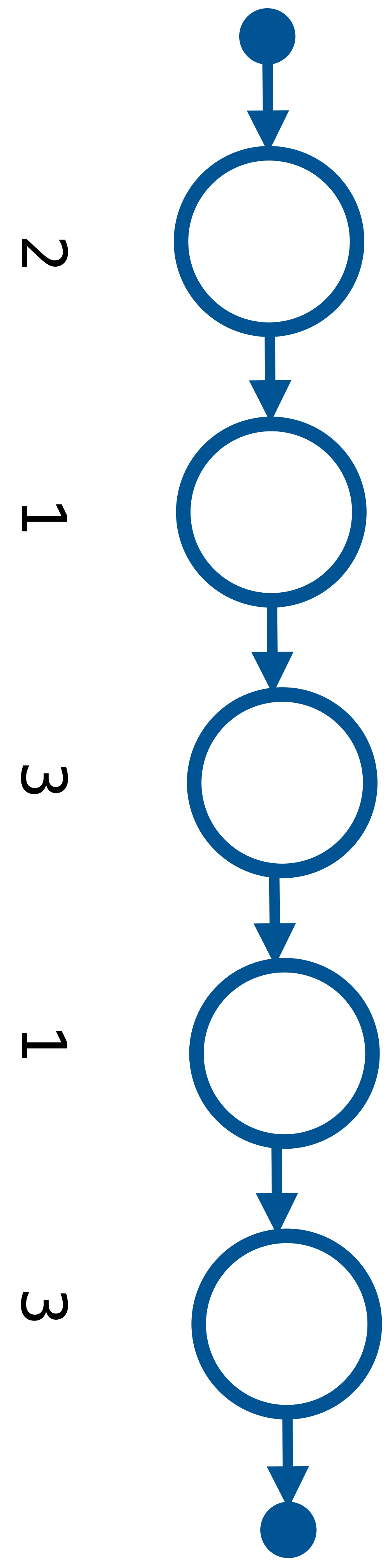
00000100000100010000 . . . 4

00000100000100010000 . . . 5

00000100000100010000 . . . 6

time is now at a fixed framerate

Position-within-state feature




```
000001000001000100000 . . . 2 0.50
000001000001000100000 . . . 2 1.00

000001000001000100000 . . . 2 0.50
000001000001000100000 . . . 2 1.00
000001000001000100000 . . . 3 1.00
000001000001000100000 . . . 4 0.33
000001000001000100000 . . . 4 0.66
000001000001000100000 . . . 4 1.00
000001000001000100000 . . . 5 1.00
000001000001000100000 . . . 6 0.33
000001000001000100000 . . . 6 0.66
000001000001000100000 . . . 6 1.00

000001000001000100000 . . . 6 0.33
000001000001000100000 . . . 6 0.66
000001000001000100000 . . . 6 1.00
```

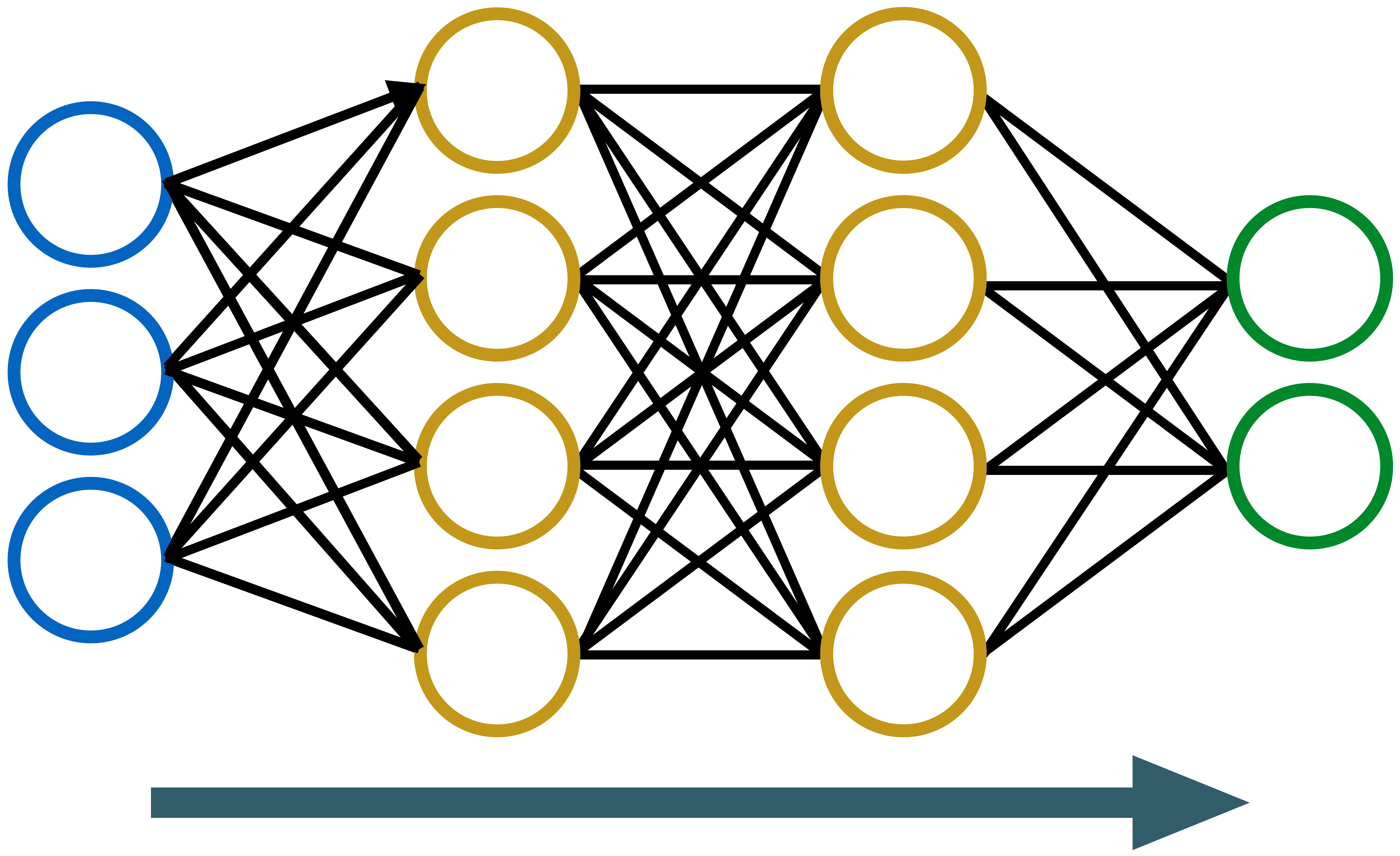
```
[l] in the context
#~p-l+i=z:2_3/. . .

with a duration of
10 frames (50ms)
```

real example of prepared features

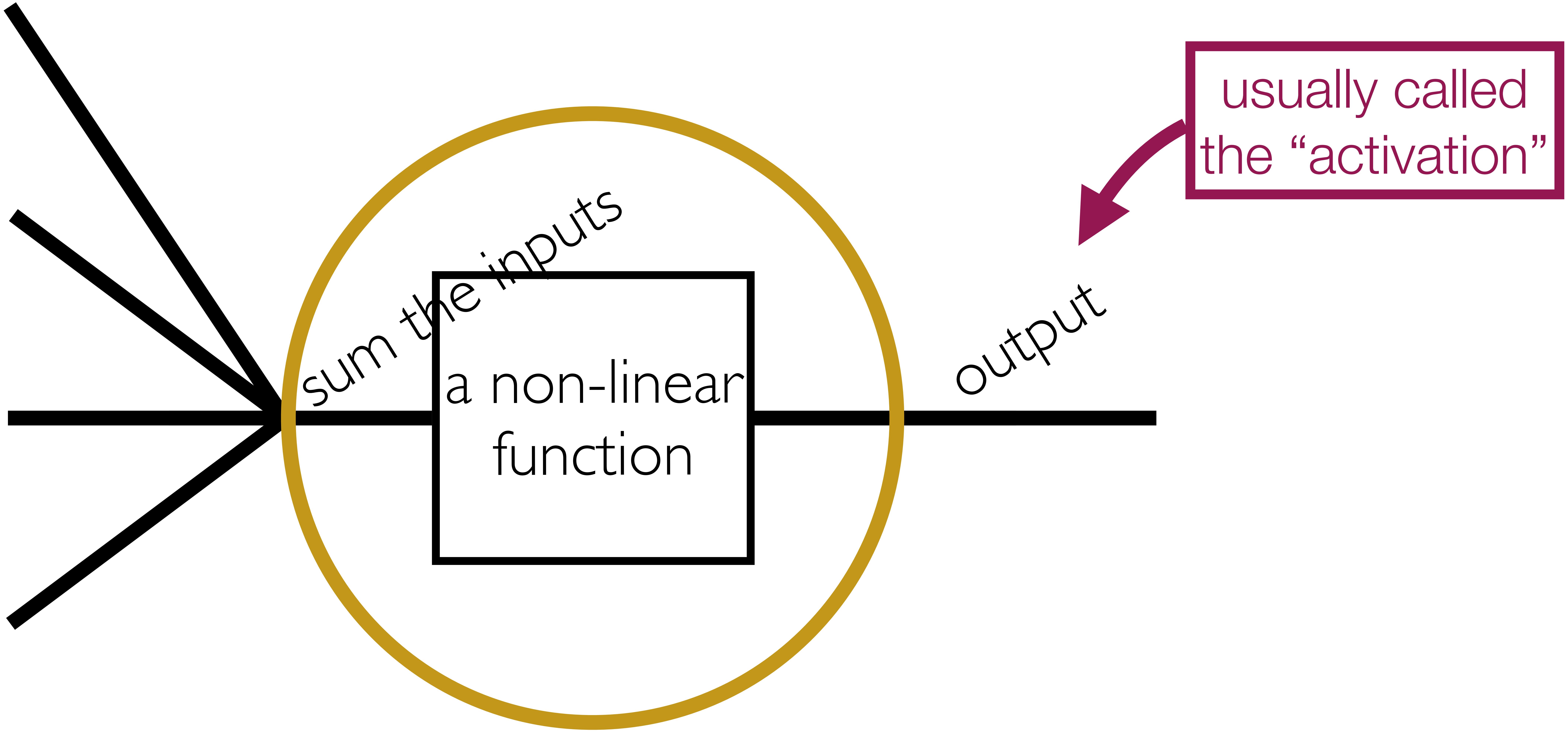
Speech synthesis using Neural Networks

- preparing the input features
- <u>what is a Neural Network?</u>
- generating speech with a Neural Network
- training a Neural Network

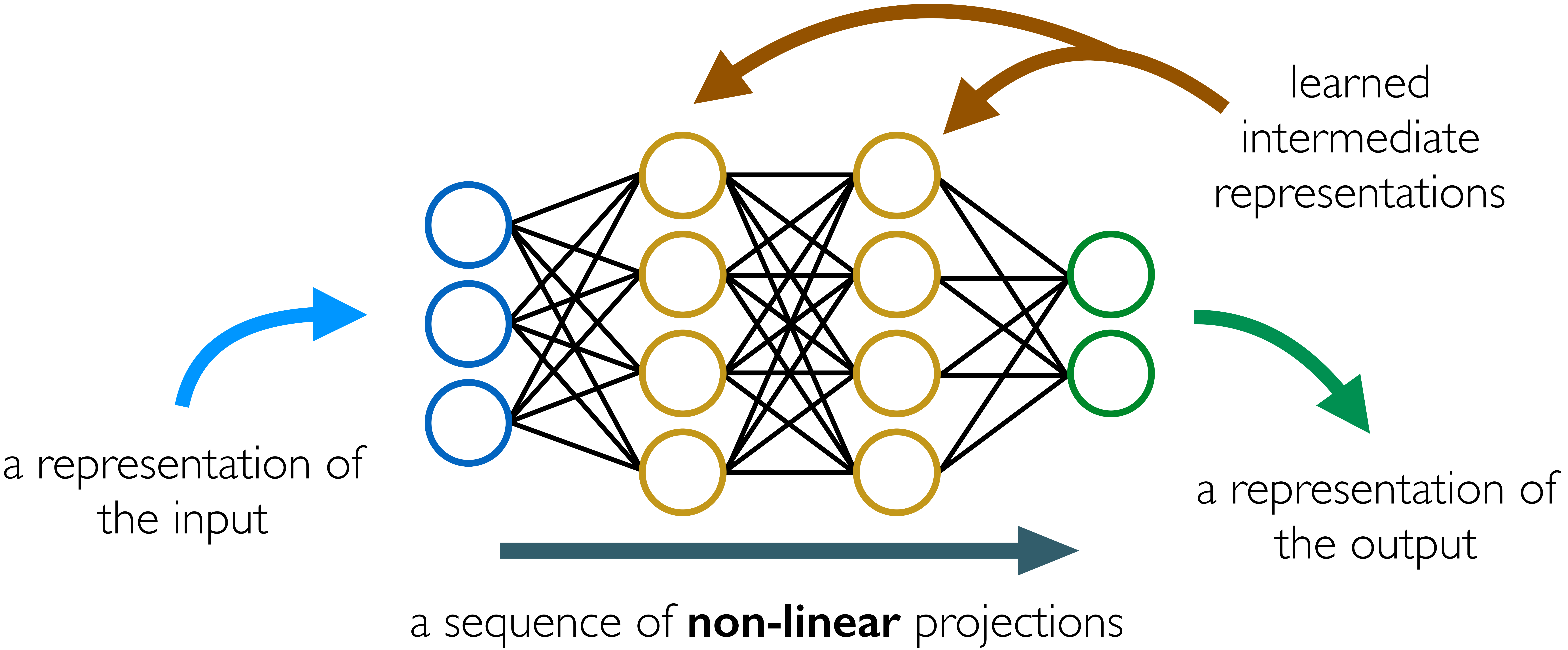
A simple "feed forward" neural network



What is a unit, and what does it do?



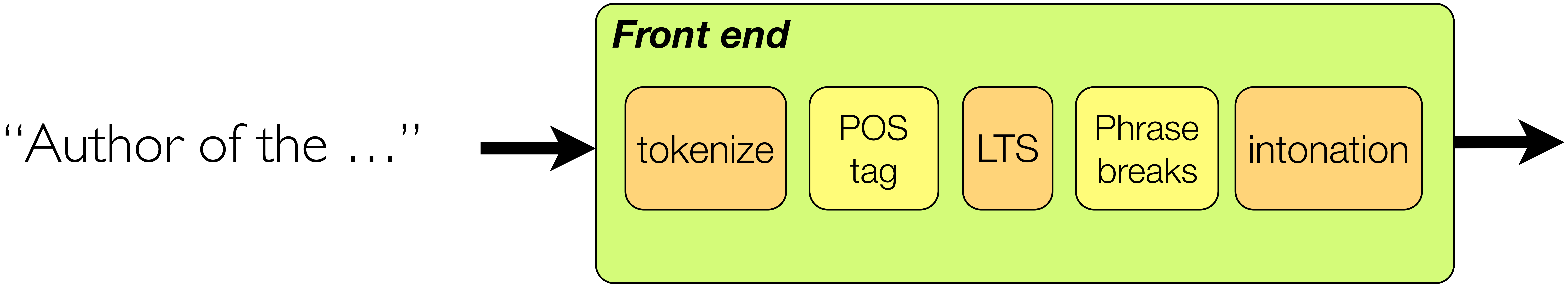
What are all those layers for?



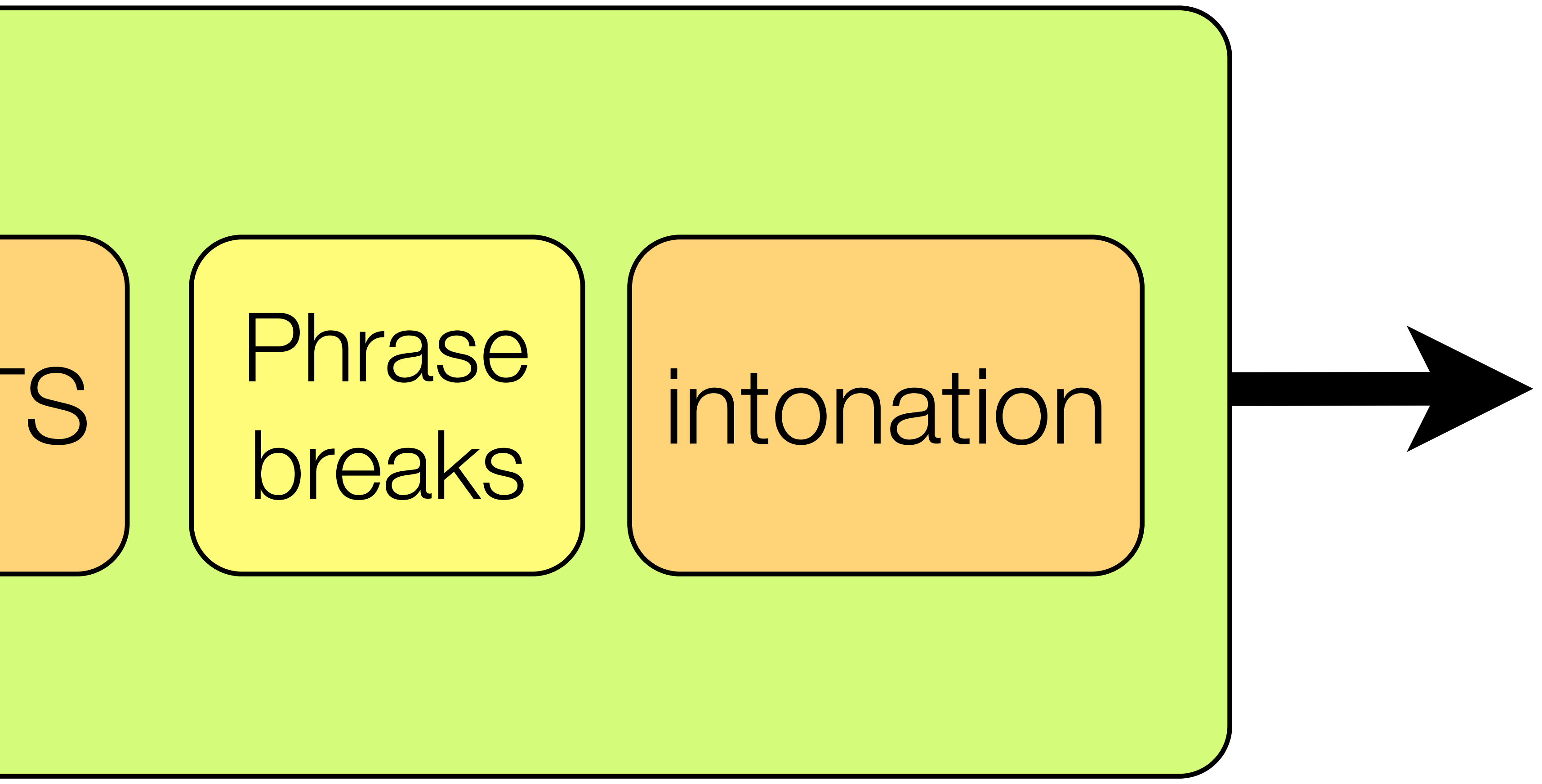
Speech synthesis using Neural Networks

- preparing the input features
- what is a Neural Network?
- <u>generating speech with a Neural Network</u>
- training a Neural Network

Putting it all together: text-to-speech with a neural network



Putting it all together: text-to-speech with a neural network



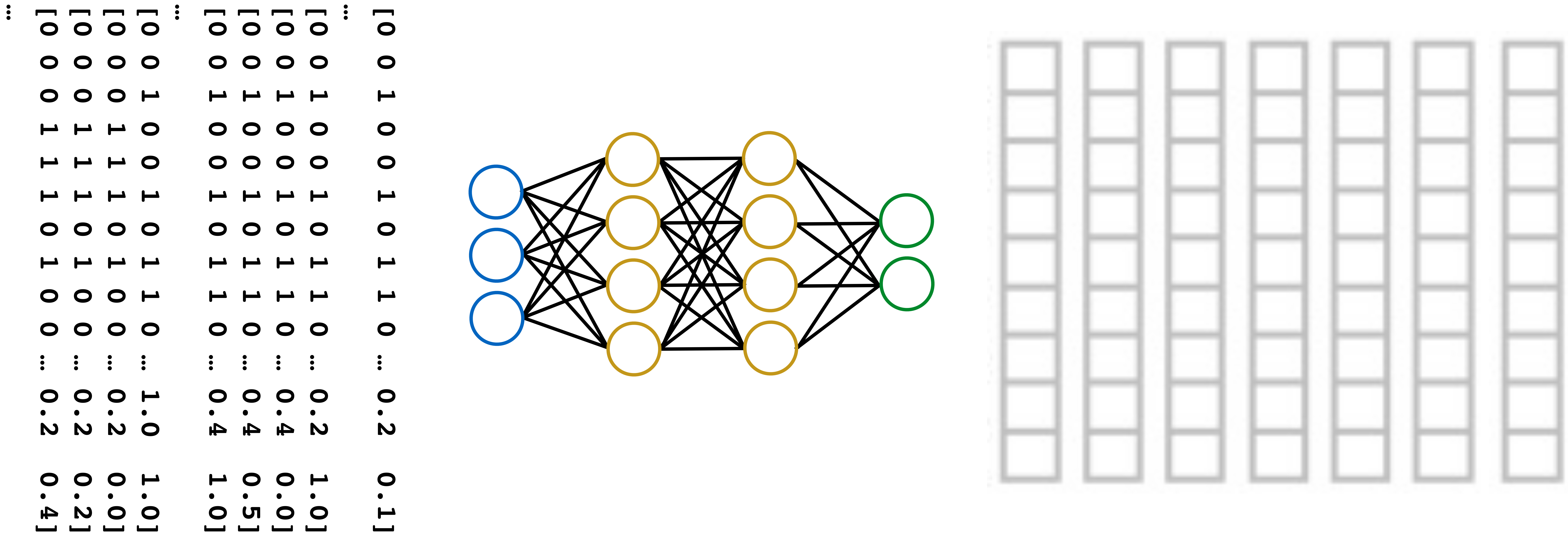

```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

Putting it all together: text-to-speech with a neural network

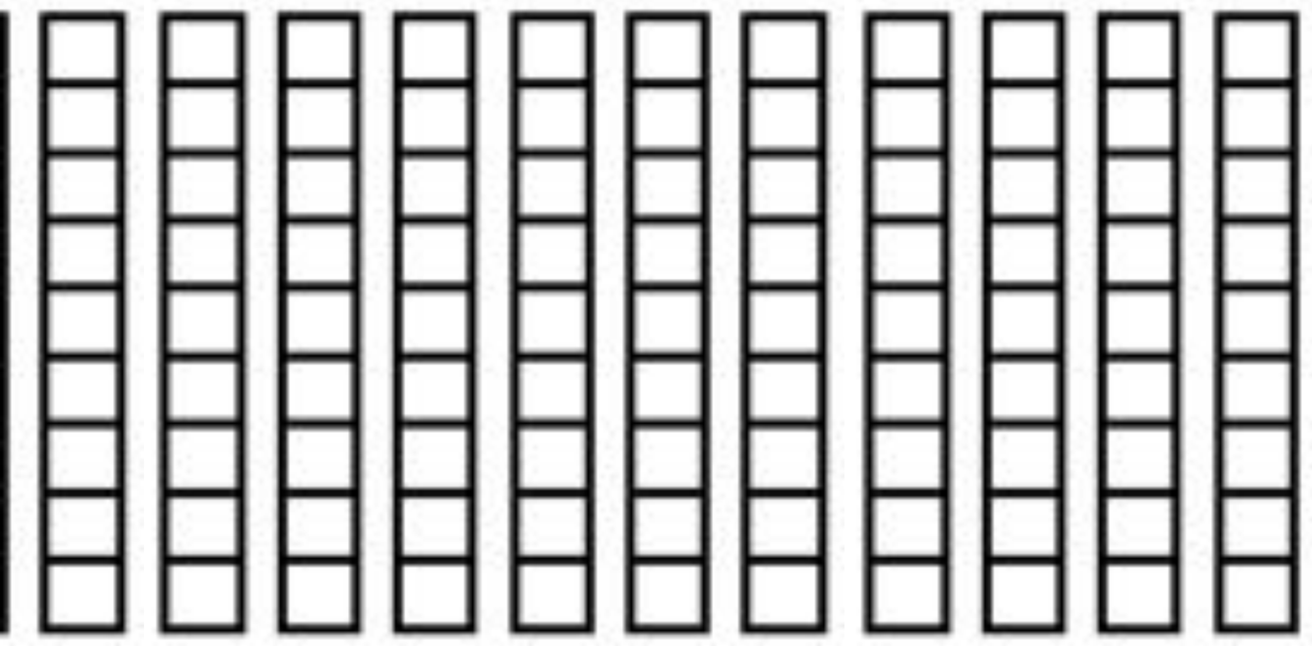
```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
ah~v-dh+ax=d@1_2/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

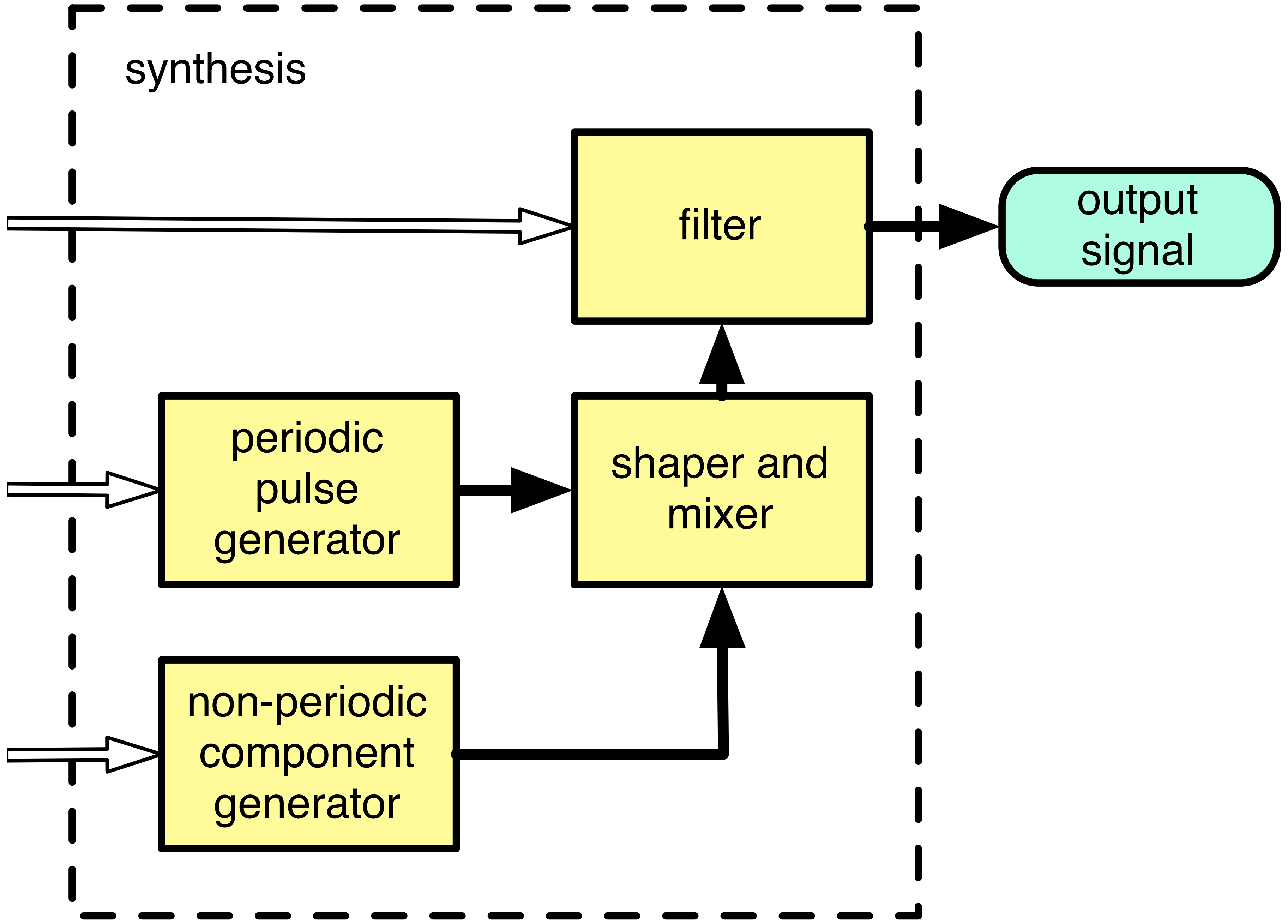
```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]
…
```

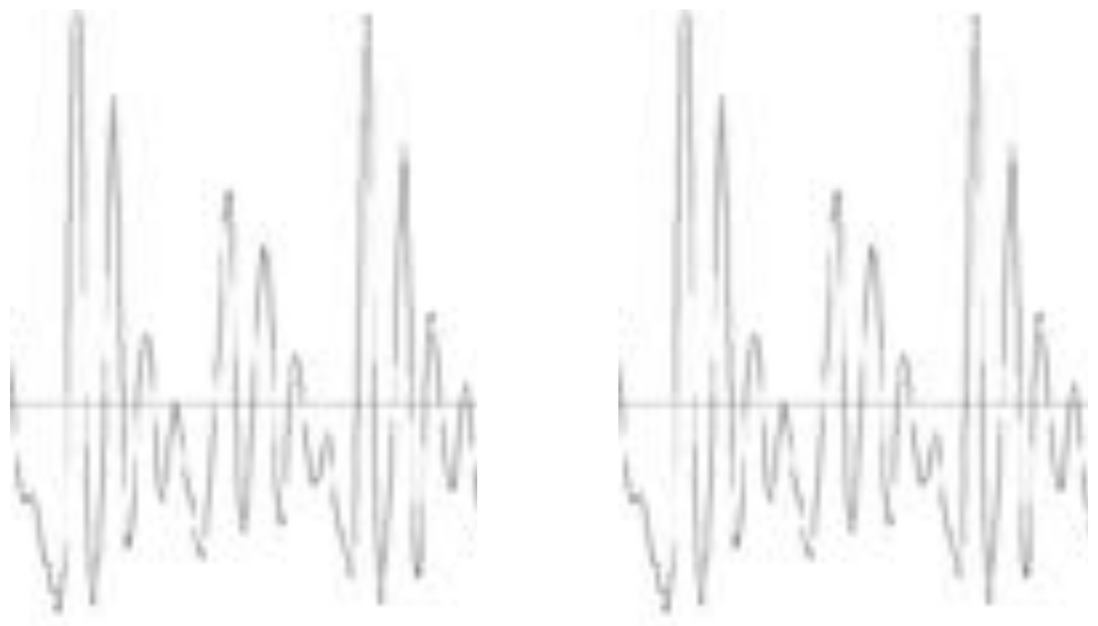
Putting it all together: text-to-speech with a neural network



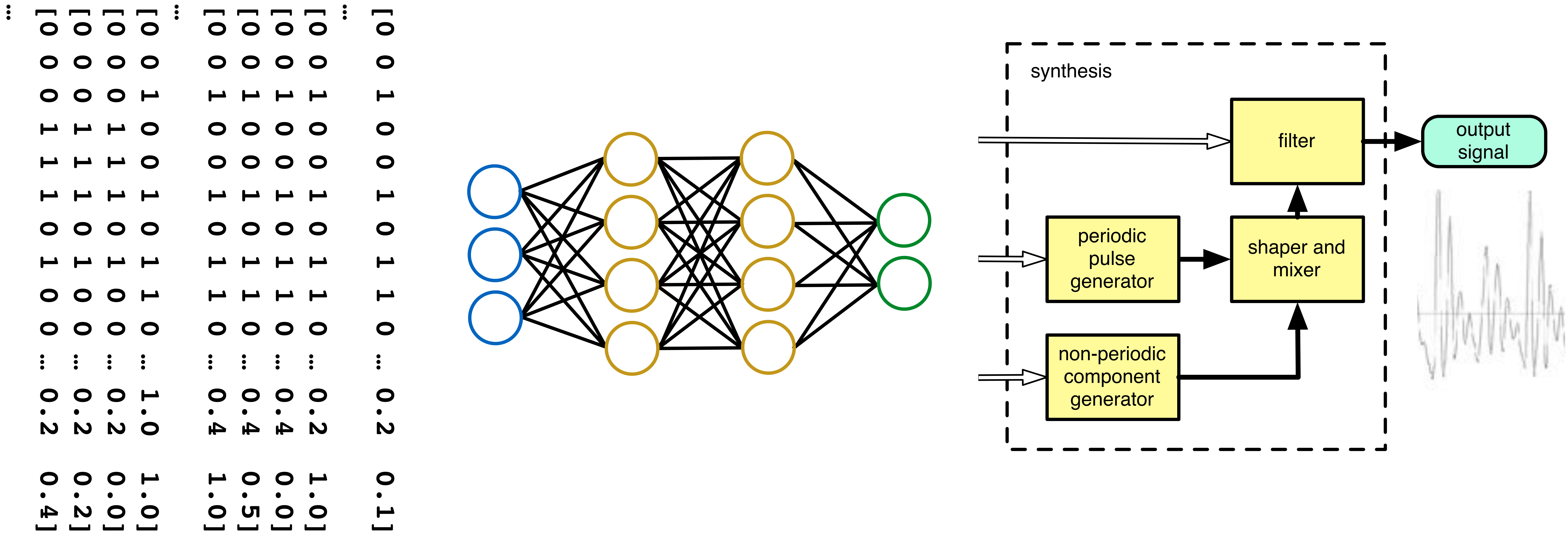
Putting it all together: text-to-speech with a neural network







Putting it all together: text-to-speech with a neural network



Speech synthesis using Neural Networks

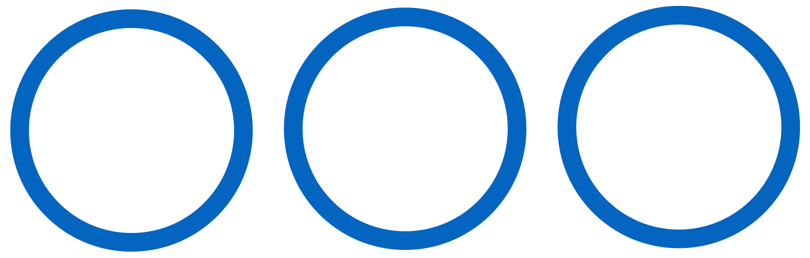
- preparing the input features
- what is a Neural Network?
- generating speech with a Neural Network
- <u>training a Neural Network</u>

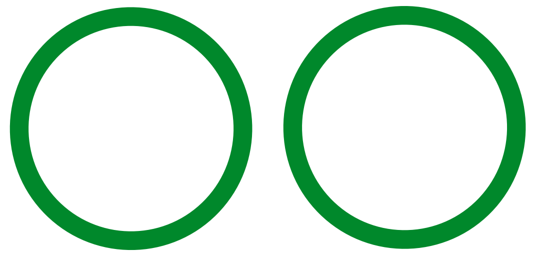
Preparing the inputs and outputs for training

- Inputs
  - linguistic features
  - plus positional features (‘counters’)
  - re-write as vectors
    - [0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 ….. 0.2 0.1]
- Outputs
  - same speech features (vocoder parameters) used in HMM synthesis
- Form input/output pairs, one pair per frame (e.g., every 5 msec)
  - how to get the alignment?

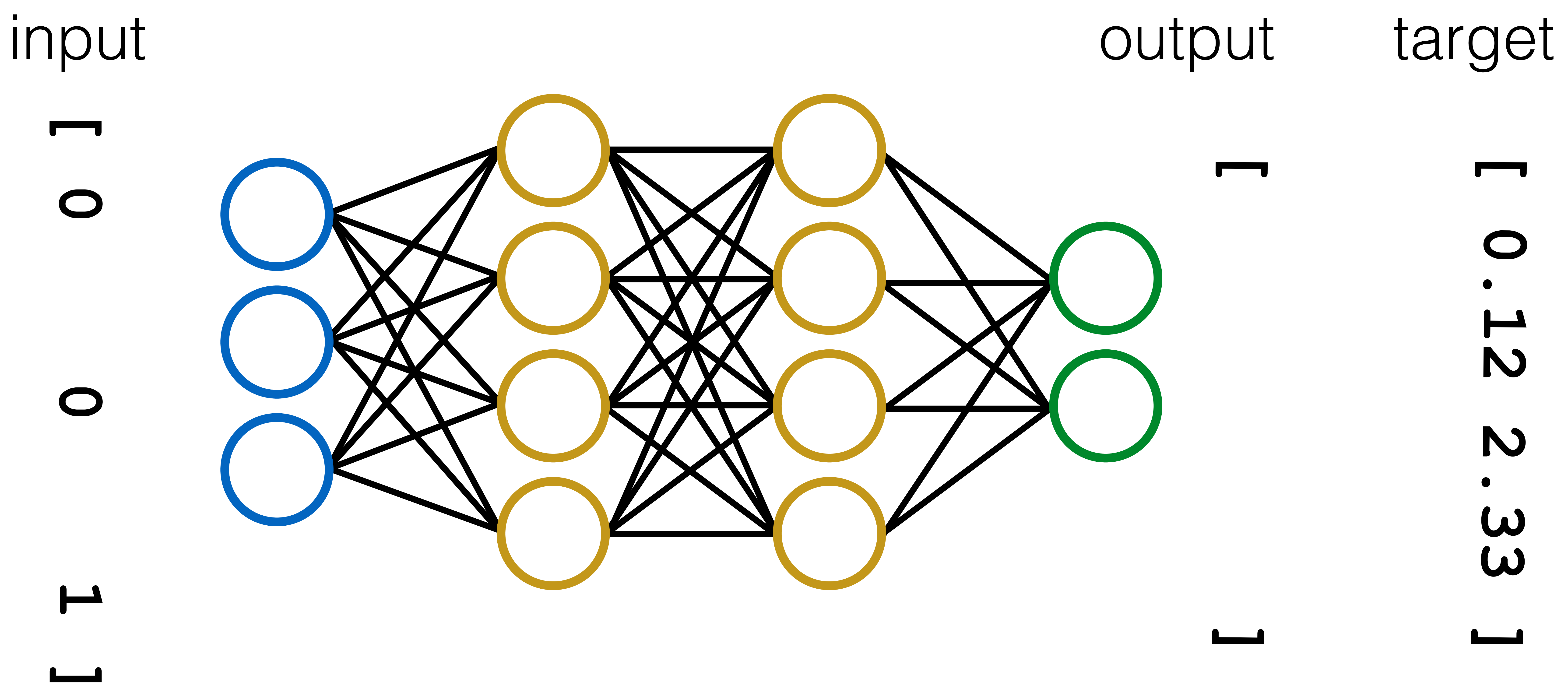
Training a neural network: pairs of input/output vectors

```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]     [0.12 2.33 2.01 0.32 6.33 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]     [0.43 2.11 1.99 0.39 4.83 … ]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]     [1.11 2.01 1.87 0.36 2.14 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]     [1.52 1.82 1.89 0.34 1.04 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]     [1.79 1.74 2.21 0.33 0.65 … ]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]     [1.65 1.58 2.68 0.31 0.73 … ]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]     [1.55 1.03 3.44 0.30 1.07 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]     [1.92 0.99 3.89 0.29 1.45 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]     [2.38 1.13 4.02 0.28 1.98 … ]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.4]     [2.65 1.98 3.94 0.29 2.16 … ]
…
```





Training a neural network: back-propagation




PAUSE/STILL

Orientation

- Simple neural networks
  - feed-forward architecture

- Constructing the input features
  - converting categorical features to binary
  - mapping linguistic timescale to fixed frame rate using the duration model

Orientation

- Simple neural networks
  - feed-forward architecture

a straightforward replacement for the regression tree

- Constructing the input features
  - converting categorical features to binary
  - mapping linguistic timescale to fixed frame rate using the duration model

Early work borrowed a duration model from an HMM system. Later work uses a better duration model.

In both cases, the 'clock' is a separate mechanism to the main regression (acoustic) model.

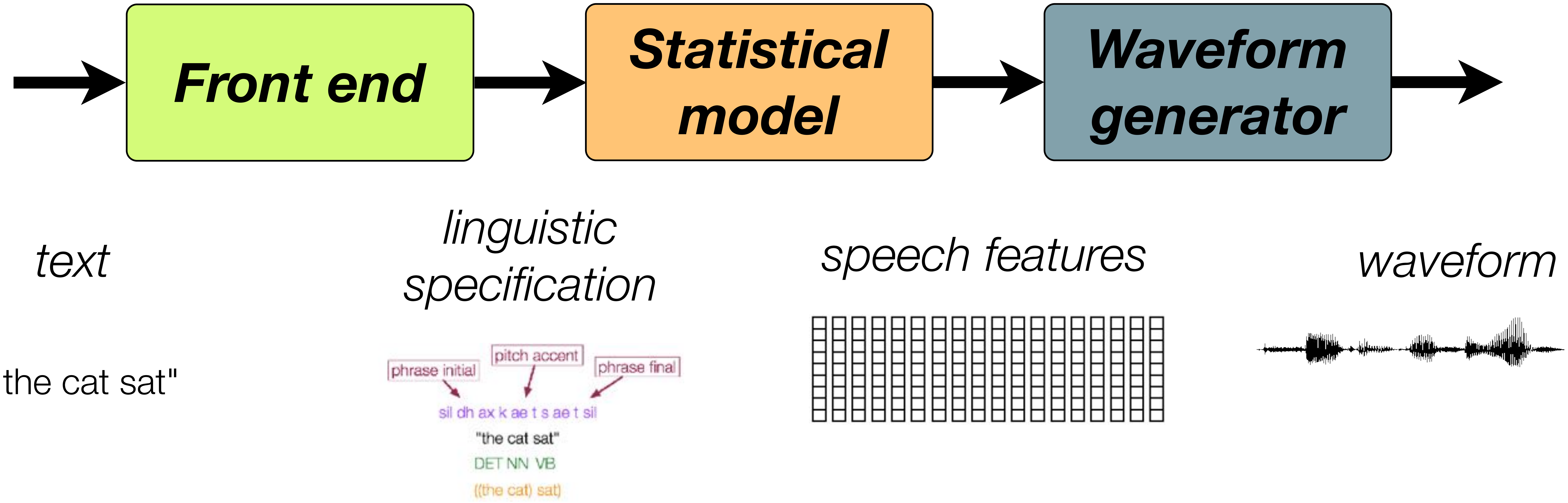
What next?

- **<u>Even better</u> regression models?**
  - different Neural Network architectures
    - recurrent, sequence-to-sequence, etc
- **Avoiding vocoding**
  - generating a spectrogram
  - direct waveform generation
  - other possibilities
- **Avoiding the front end**
  - 'raw text' input

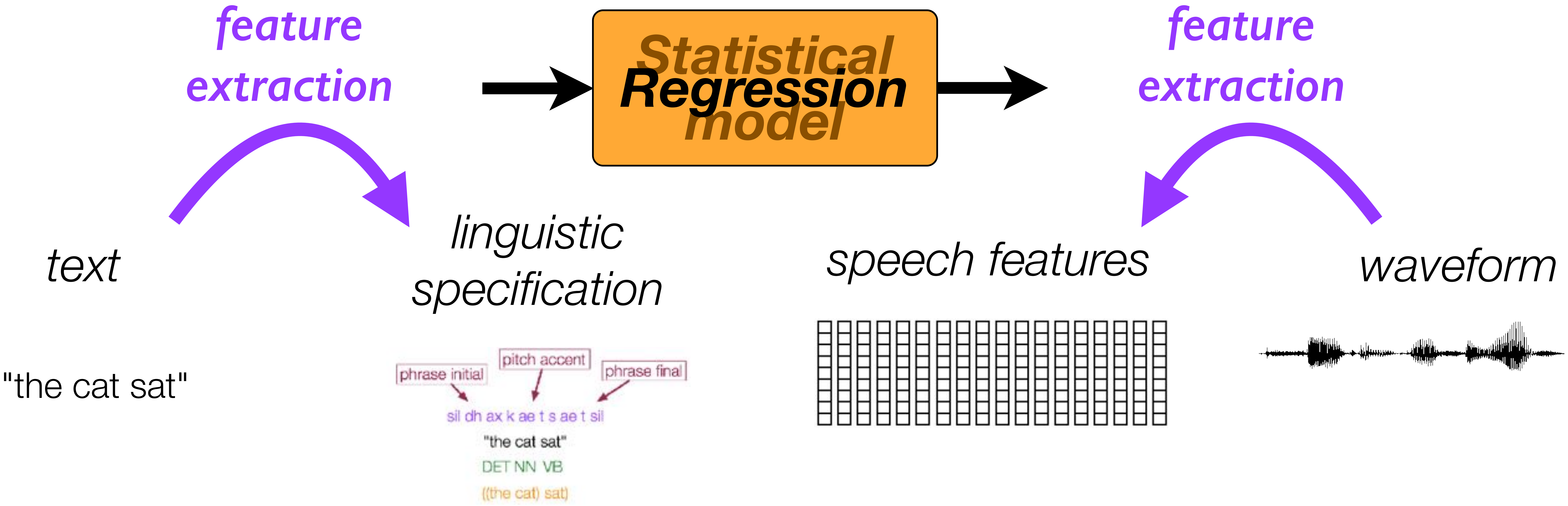
Alternative and/or advanced Neural Network techniques

- network architectures
- avoiding vocoding
- avoiding the front end

The classic three-stage pipeline of statistical parametric speech synthesis



The classic three-stage pipeline of statistical parametric speech synthesis

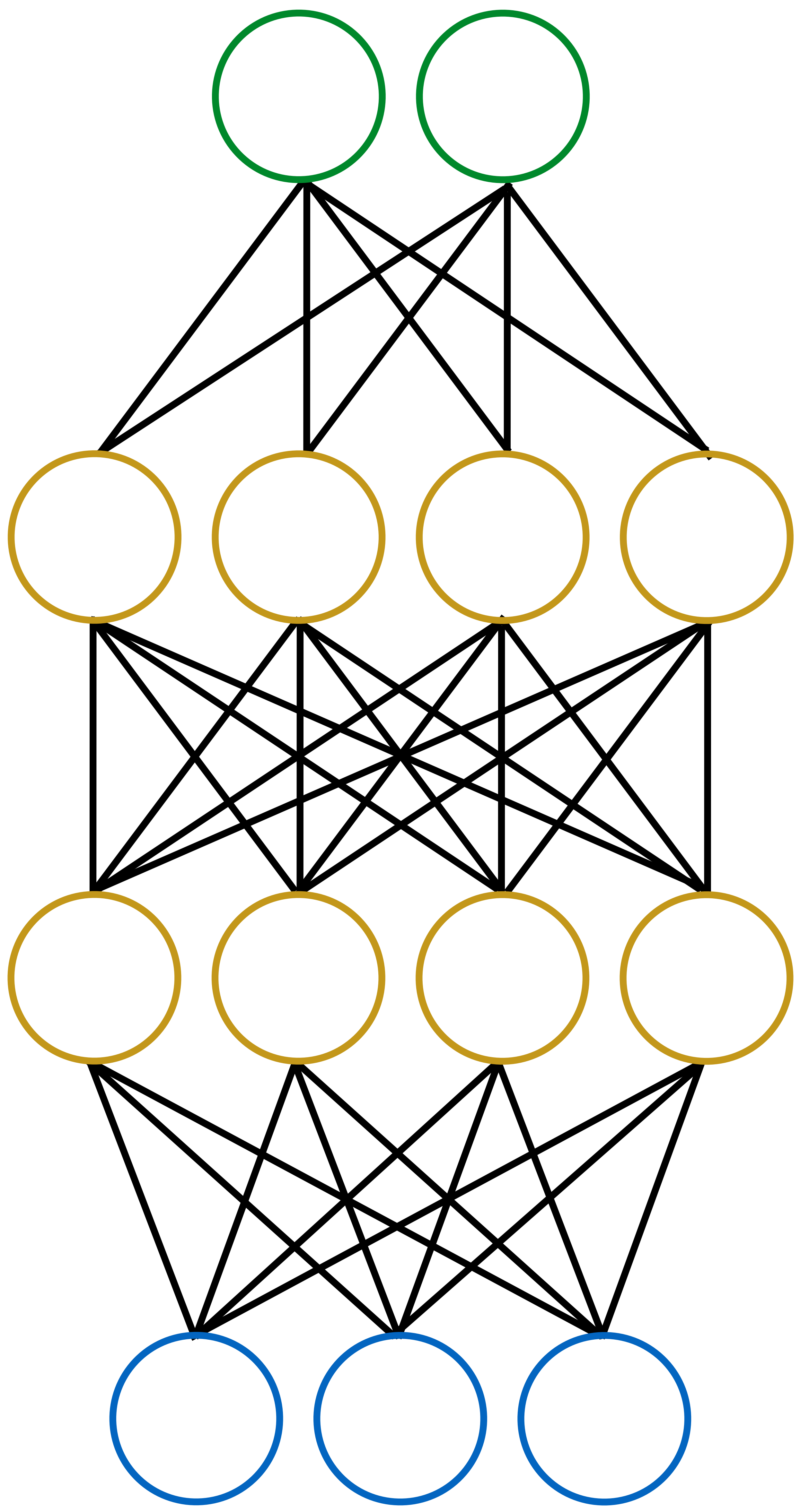


Alternative and/or advanced Neural Network techniques

- <u>network architectures</u>
- avoiding vocoding
- avoiding the front end

Feed-forward

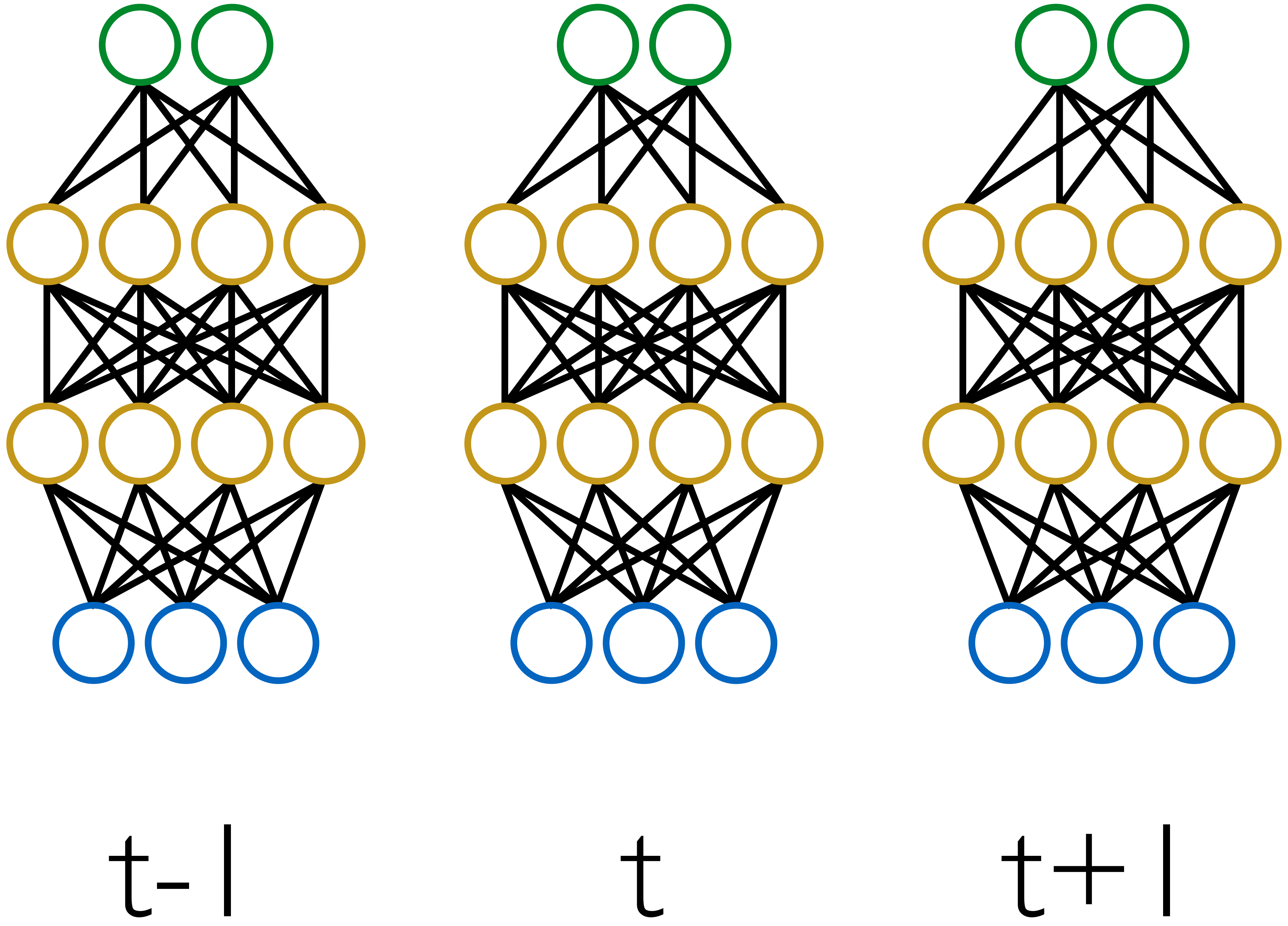
- Conceptually straightforward
- For each input frame
  - perform regression to corresponding output features
- To provide wider input context, could simply stack several frames together
  - although, remember that the linguistic features already span several timescales



Recurrent

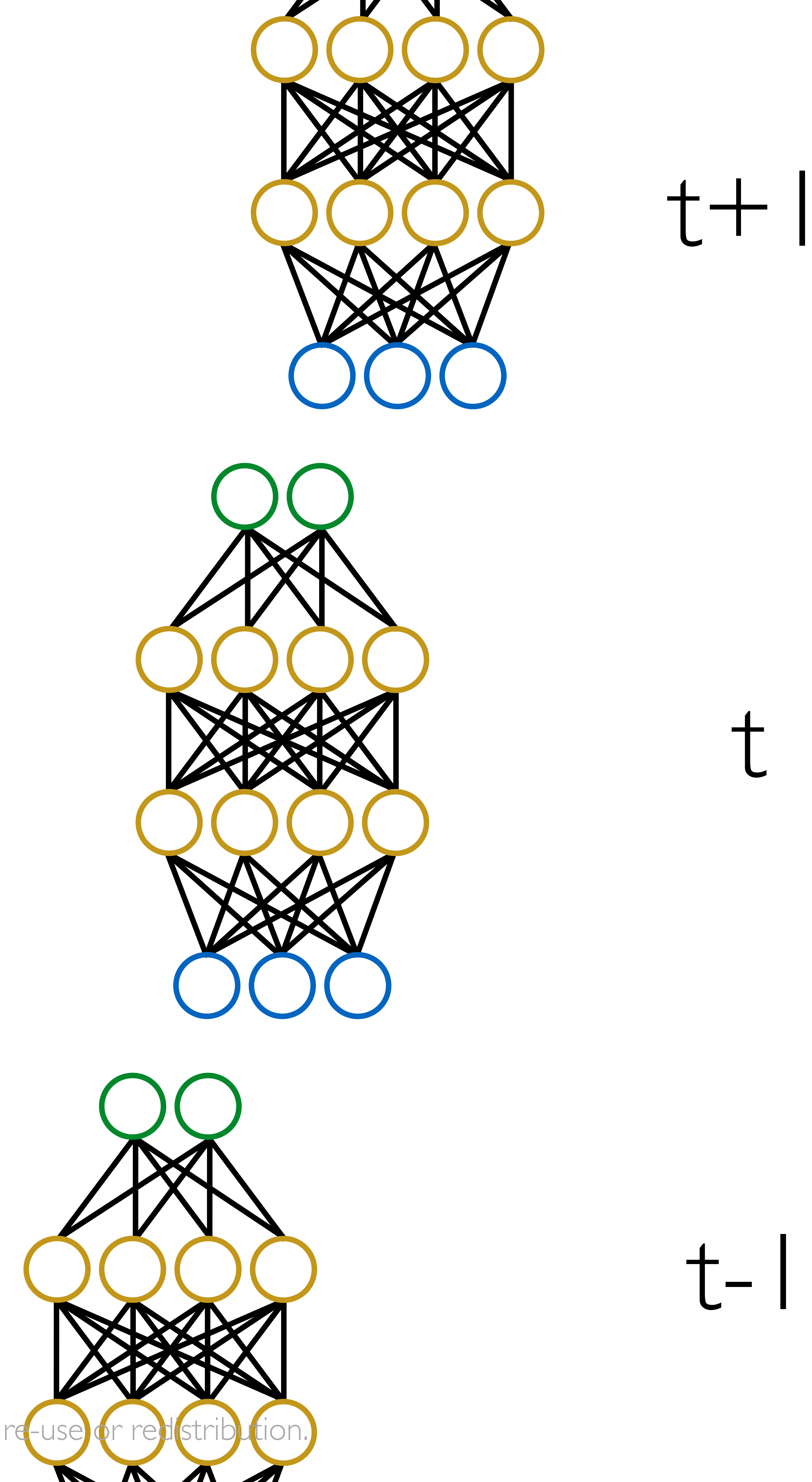
- Pass some of the outputs (or hidden layer activations) forwards in time, typically to the next time step
- A kind of **memory**
- Provides "infinite" left context

- Could also pass information backwards in time



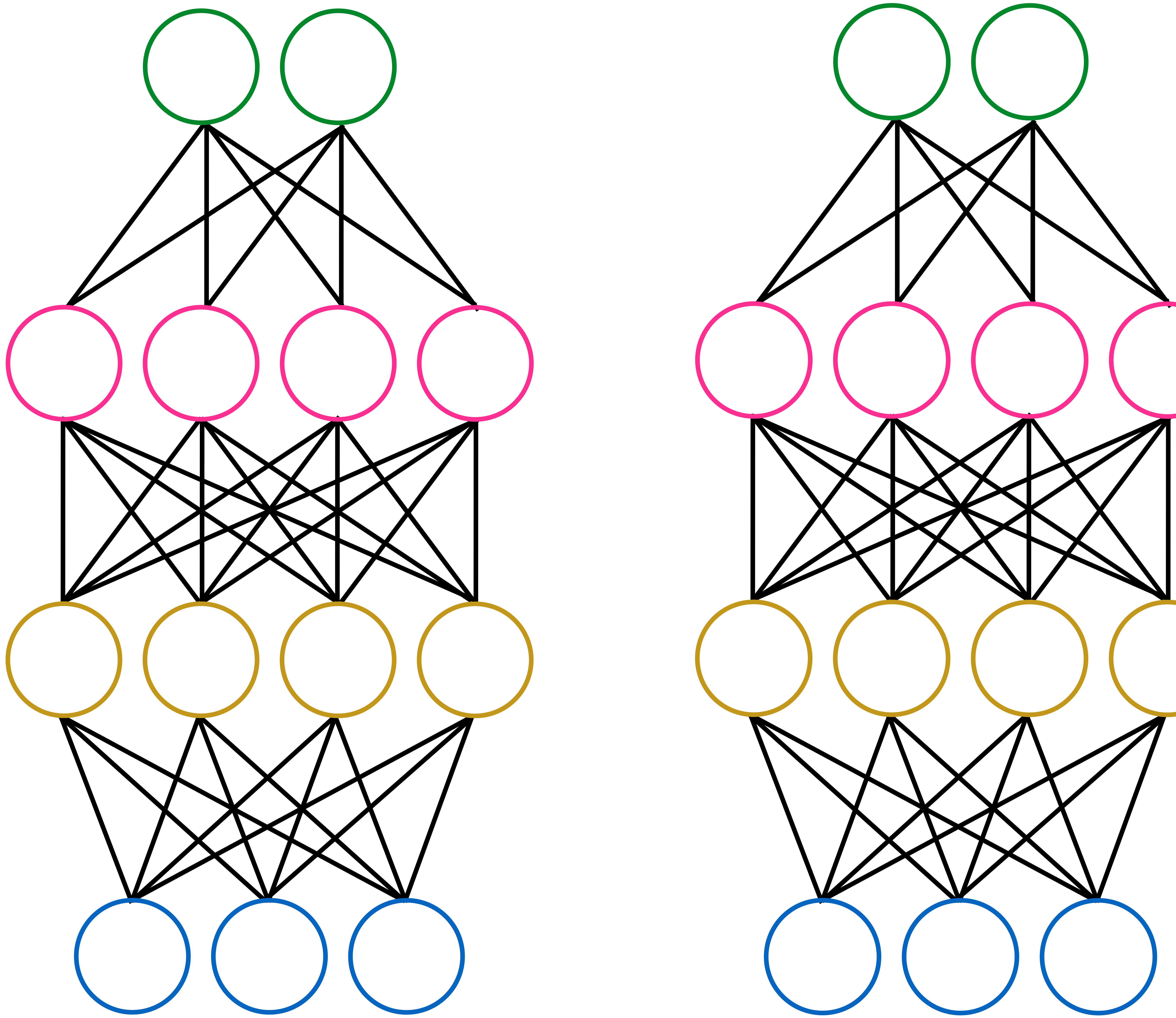
Recurrent

- Simple recurrence is equivalent to a **very deep network**
- To train this network, we have to backpropagate the derivative of the the errors (the **gradient**) through all of the layers
  - "backpropagation through time"
- Suffers from the "**vanishing gradient**" problem, for long sequences



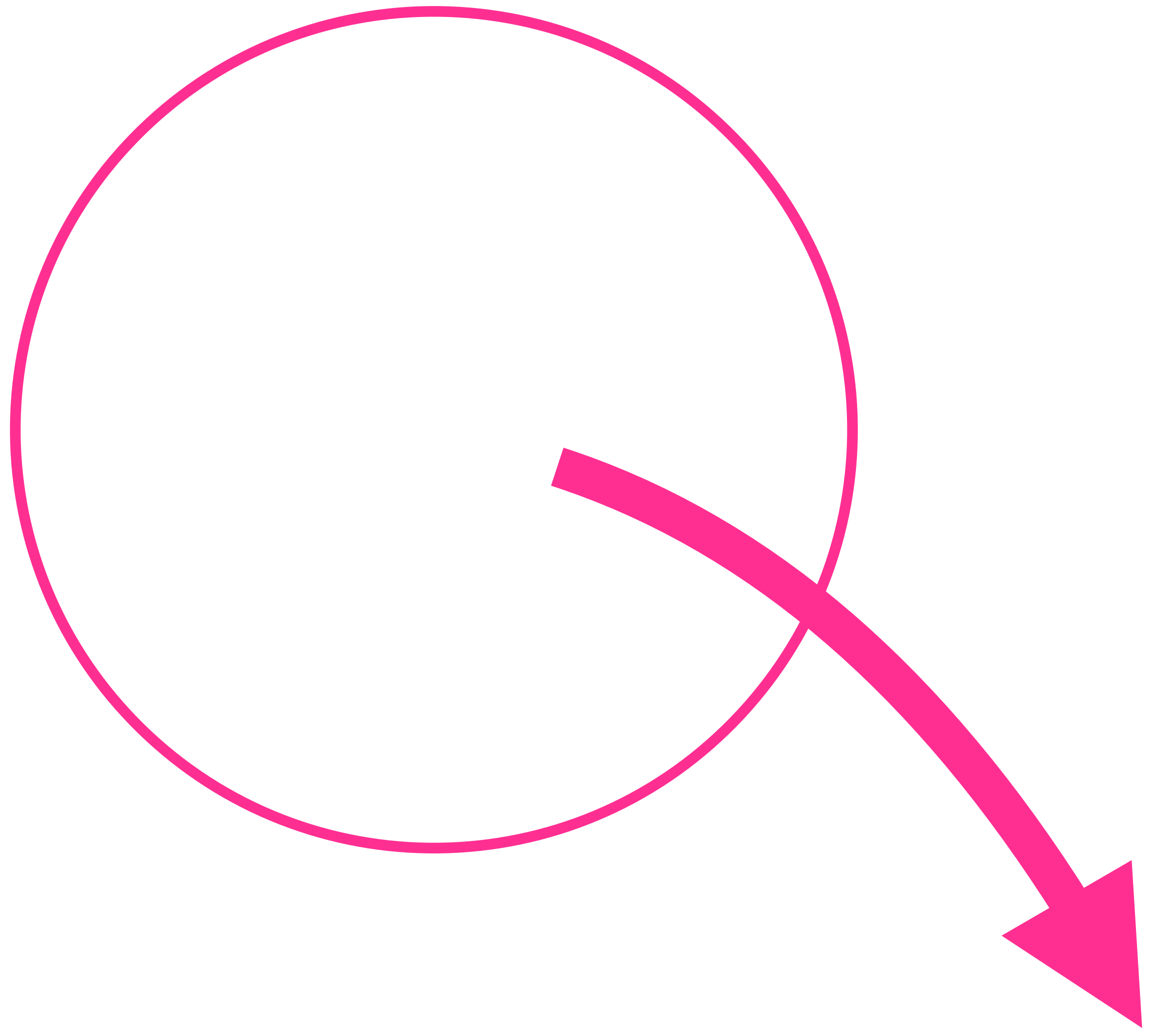
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- <u>Conceptually</u>
  - Learns when to remember
  - Remembers information 'perfectly' for some number of time steps
  - Learns when to forget



Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- <u>Conceptually</u>
  - Learns when to remember
  - Remembers for several time steps
  - Learns when to forget



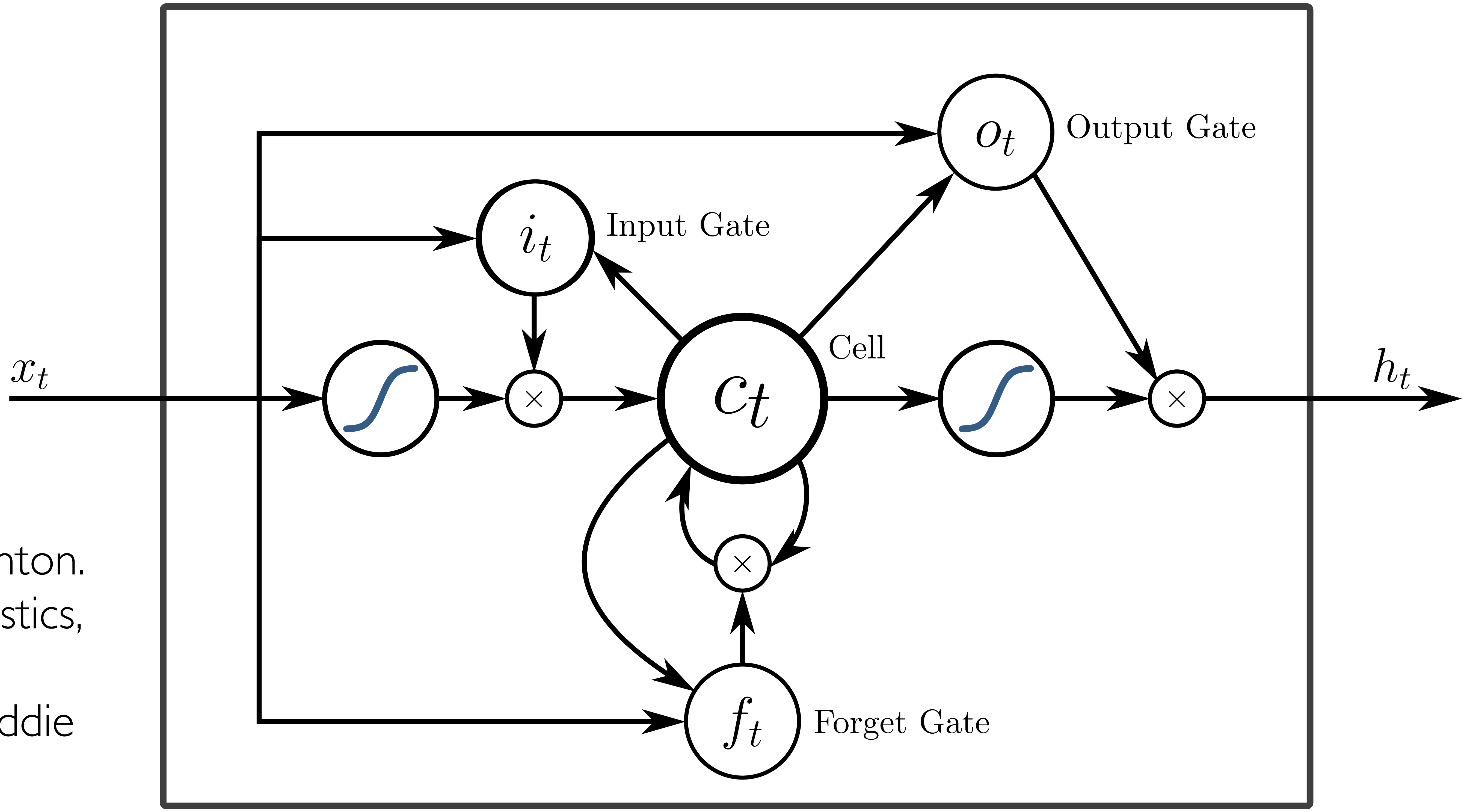


Figure from Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6645–6649. IEEE, 2013, redrawn as SVG by Eddie Antonio Santos

Orientation

- Feed-forward architecture
  - no memory
- "Simple" recurrent neural networks
  - vanishing gradient problem
- LSTM unit solves vanishing gradient problem
- **But**
  - inputs and outputs at same frame rate
  - need an external 'clock' or alignment mechanism to 'upsample' the inputs

Sequence-to-sequence

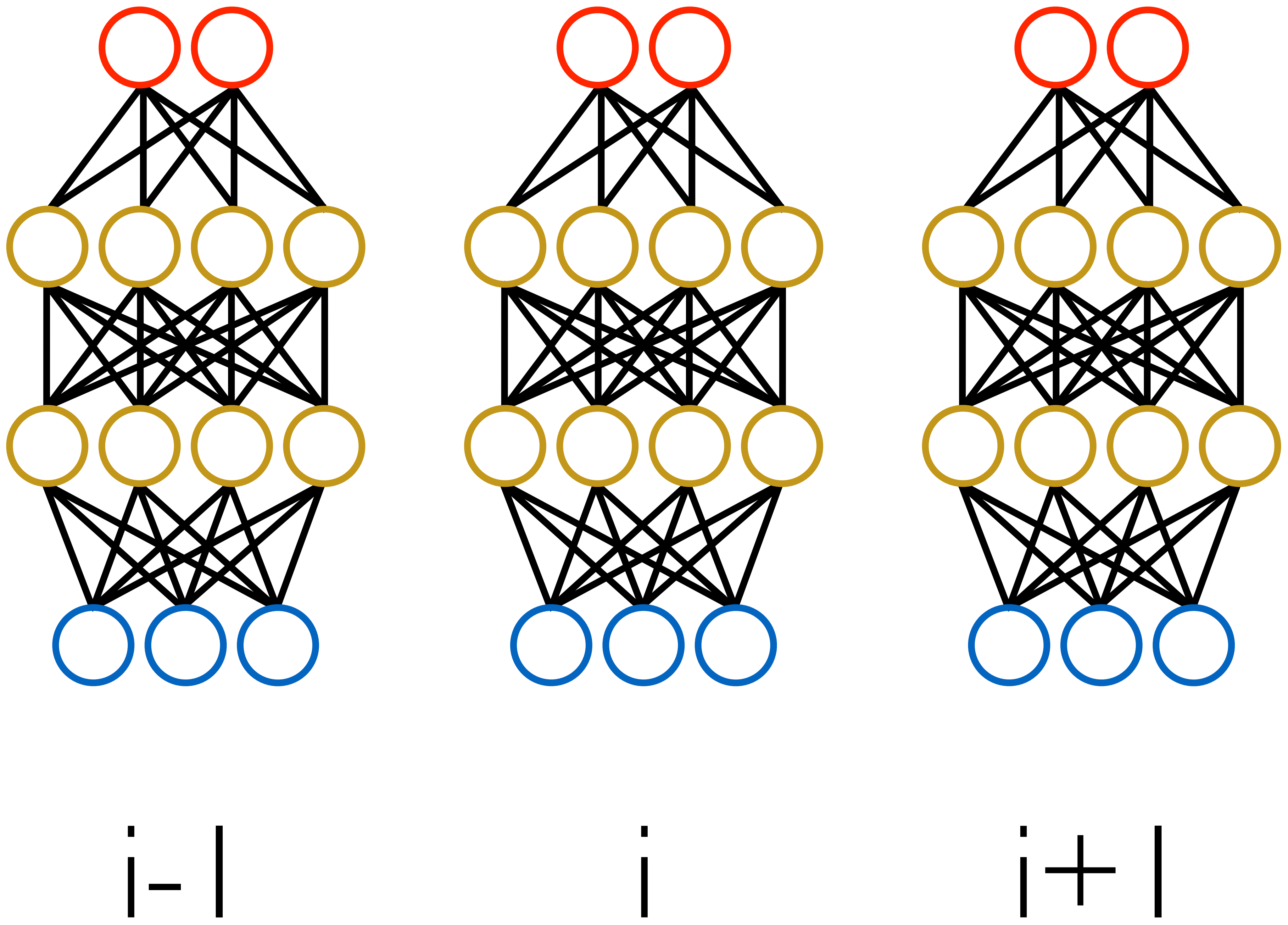
- Next step is to integrate the alignment mechanism into the network itself
- Now, length of input sequence may be different to length of output sequence

- For example
  - input: sequence of context-dependent phones
  - output: acoustic frames (for the vocoder)

- <u>Conceptually</u>
  - **read** in the entire input sequence; **memorise** it using a **fixed-length representation**
  - given that representation, **write** the output sequence

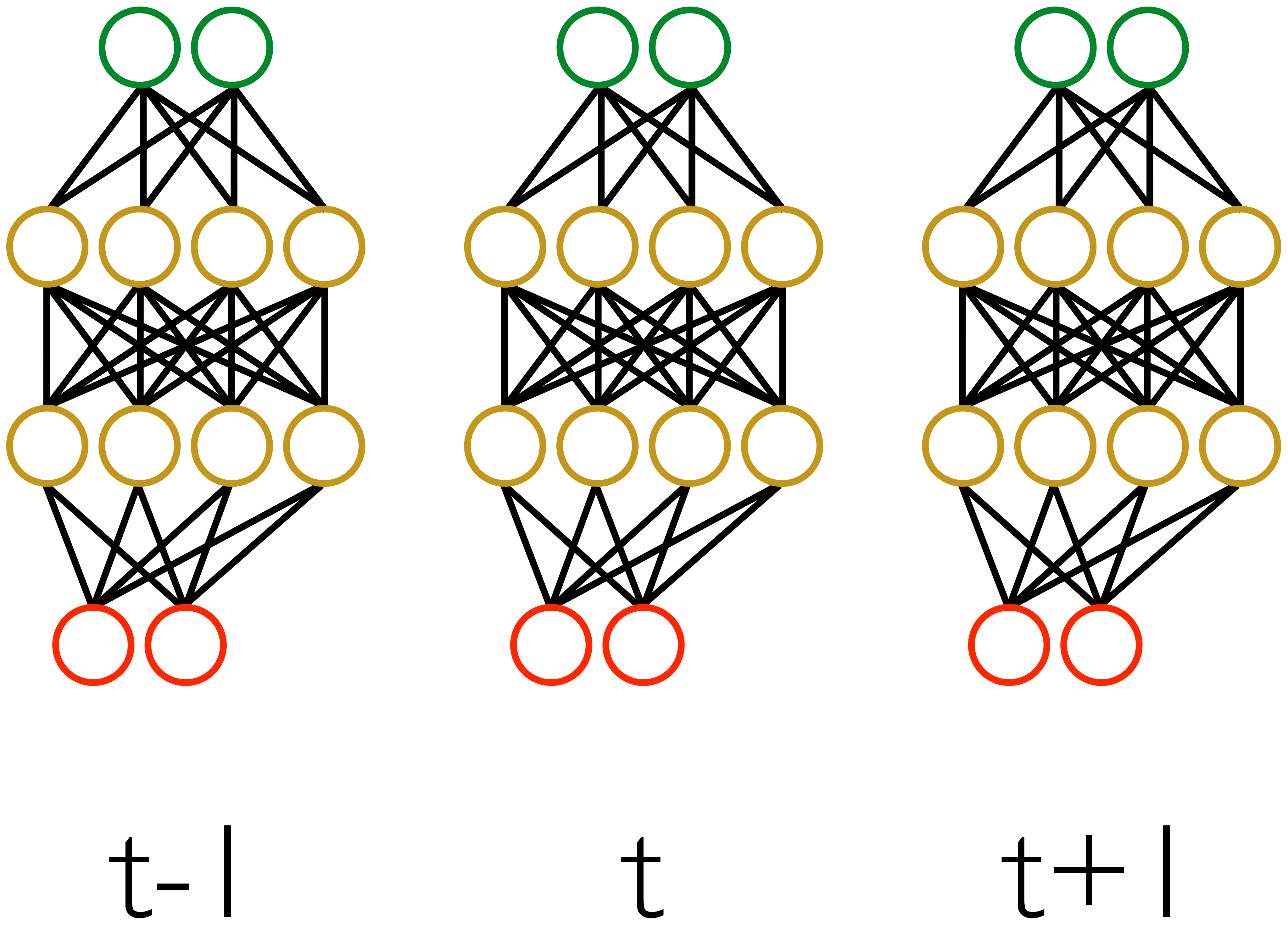
Sequence-to-sequence (just <u>conceptually</u>)

- The **encoder**
- A recurrent network that "reads" the entire input sequence and "summarises" or "memorises" it using a fixed-length representation



Sequence-to-sequence (just <u>conceptually</u>)

- The **decoder**
- A recurrent network that takes that fixed-length representation as its initial state, then generates the entire output sequence



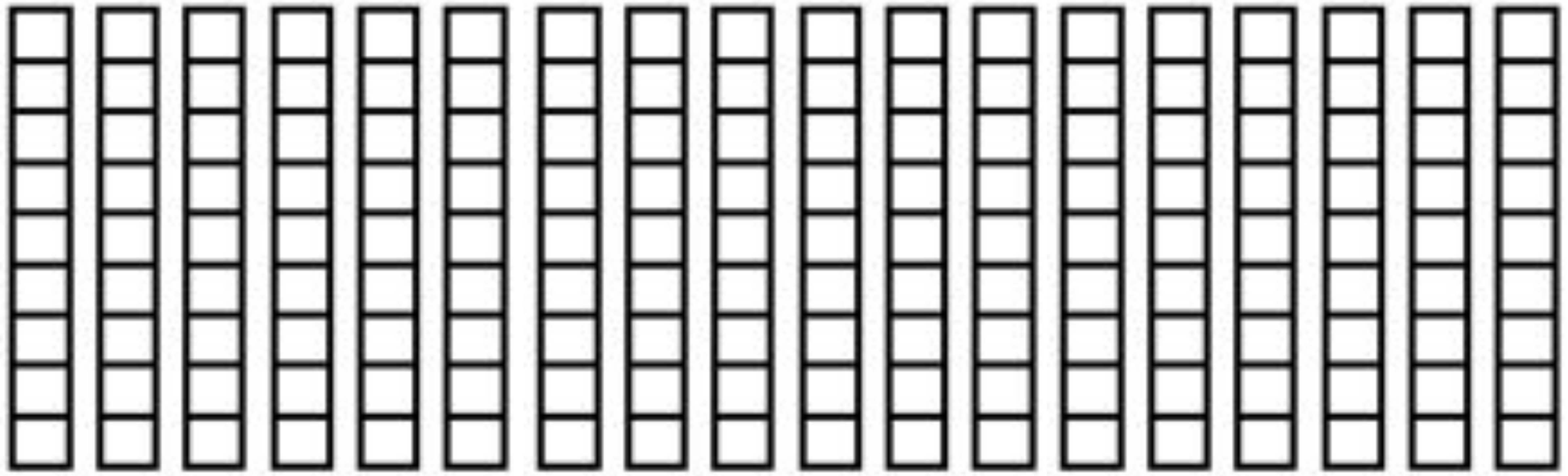
Alignment in sequence-to-sequence models: adding "attention"

- Basic model, as presented, has **no alignment** between input and output
- Get better performance by adding **"attention"** to the input sequence, in the decoder

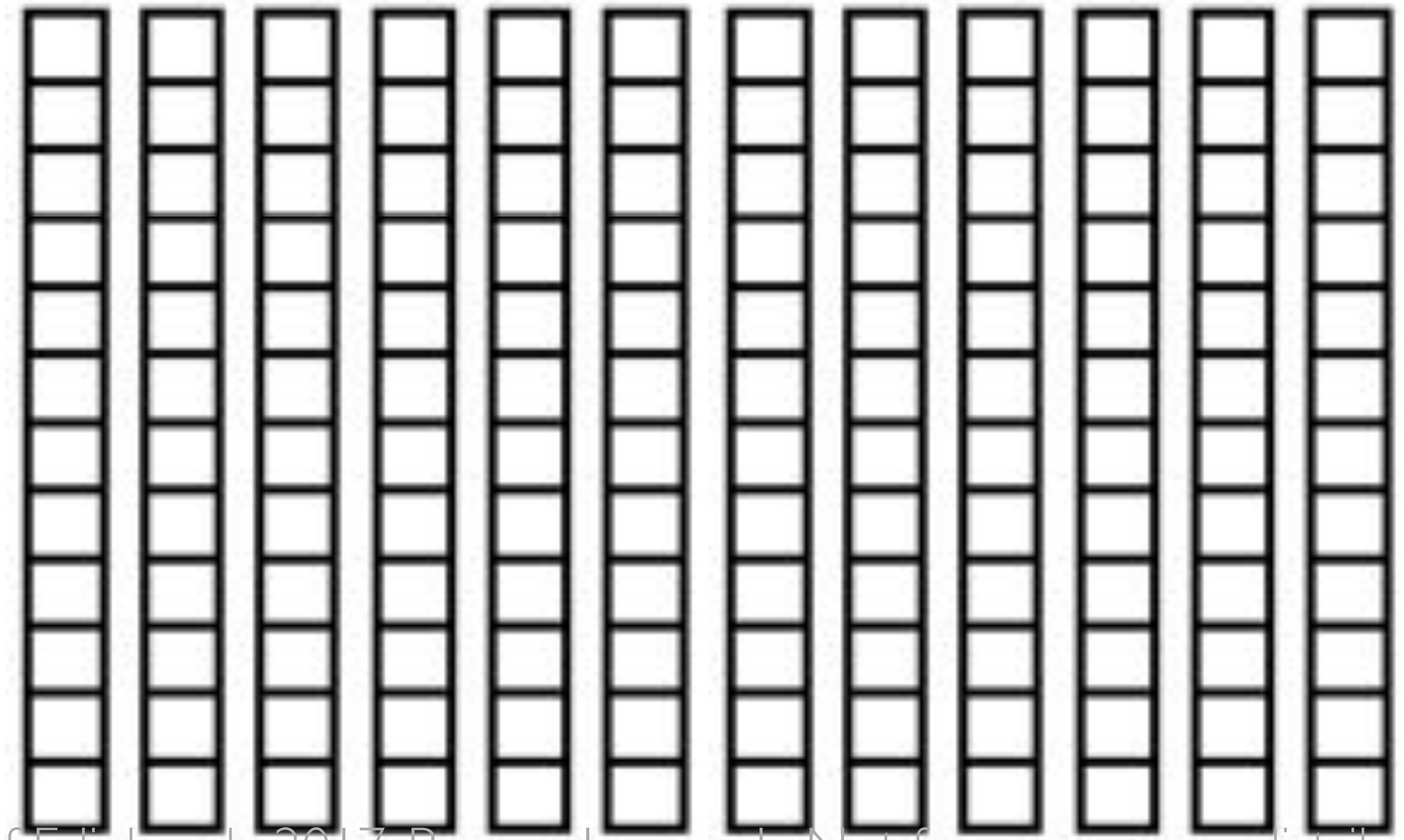
See also Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In Proc ICLR 2015

Alignment in sequence-to-sequence models: adding “attention”

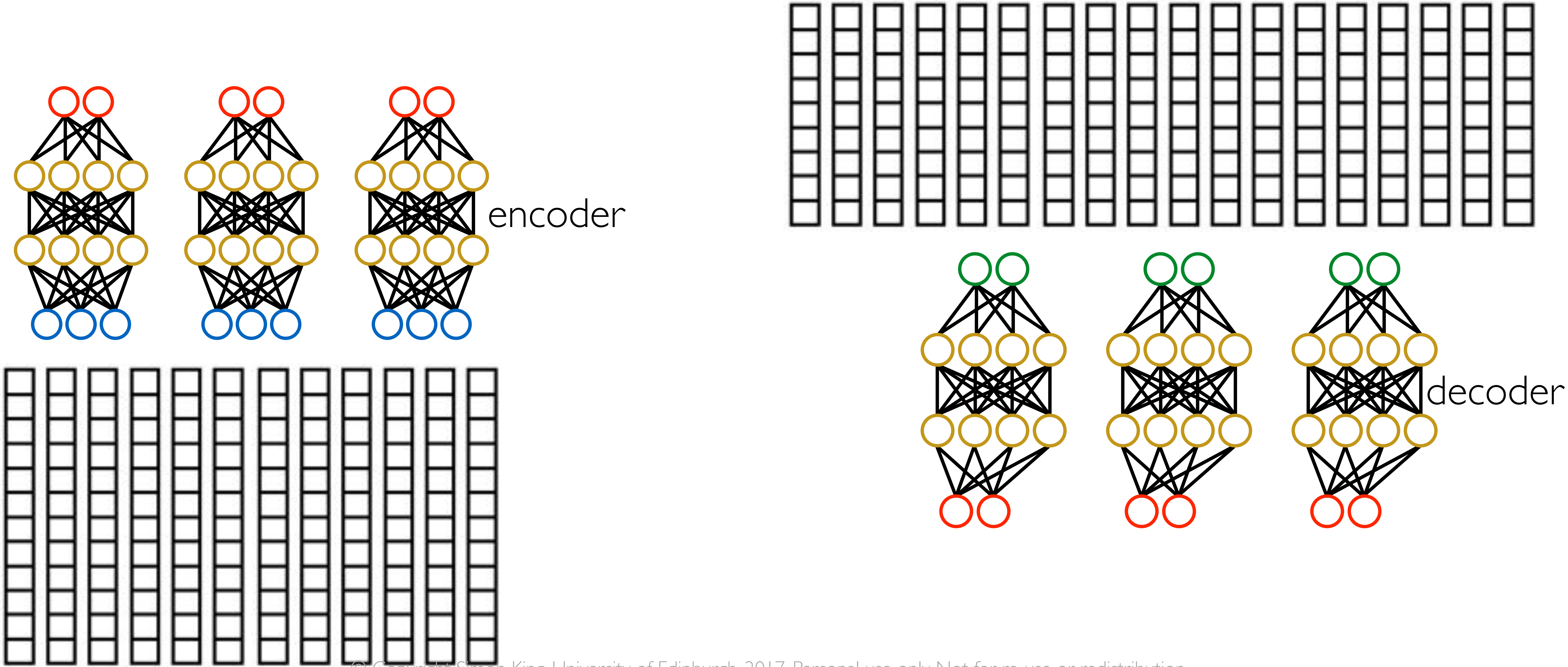
output sequence
(speech features)



input sequence
(linguistic specification)



Alignment in sequence-to-sequence models: adding "attention"



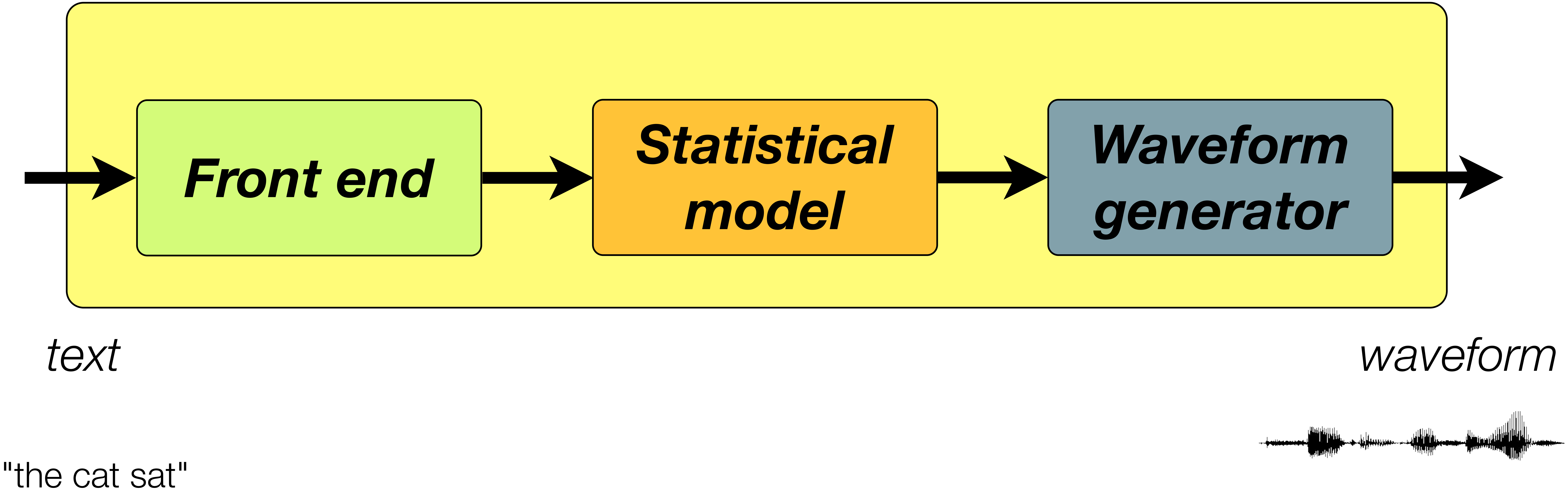
Alignment in sequence-to-sequence models: ASR-style acoustic features

- **Trying to do ASR with typical TTS vocoder features does not work very well**
  - J. Dines, J. Yamagishi and S. King, "Measuring the Gap Between HMM-Based ASR and TTS," in IEEE Journal of Selected Topics in Signal Processing, vol. 4, no. 6, pp. 1046-1058, Dec. 2010. doi: 10.1109/JSTSP.2010.2079315
- So, we would expect to get better performance by using ASR-style acoustic features (just for the alignment part of the model)
  - e.g. Mel-cepstrum or log Mel filterbank
- This is exactly what people are finding (e.g., the Tacotron)

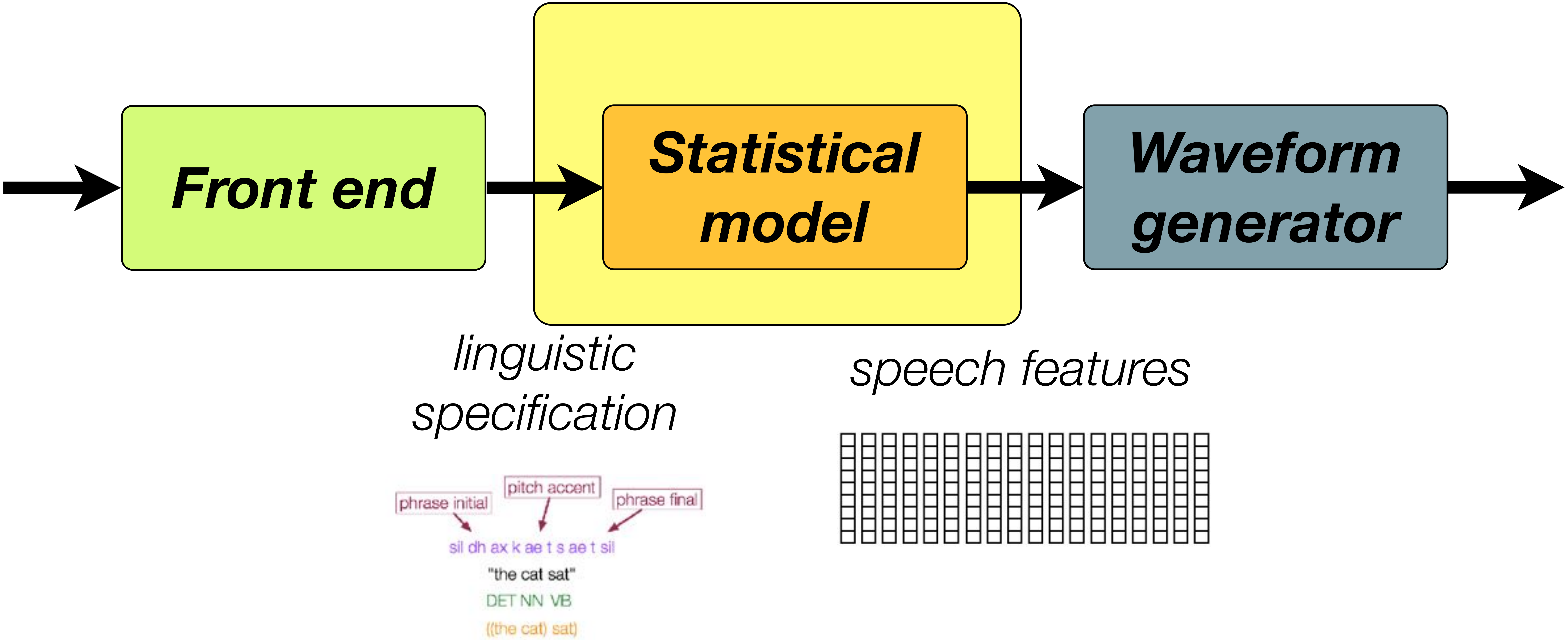
Alternative and/or advanced Neural Network techniques

- network architectures
- avoiding vocoding
- avoiding the front end

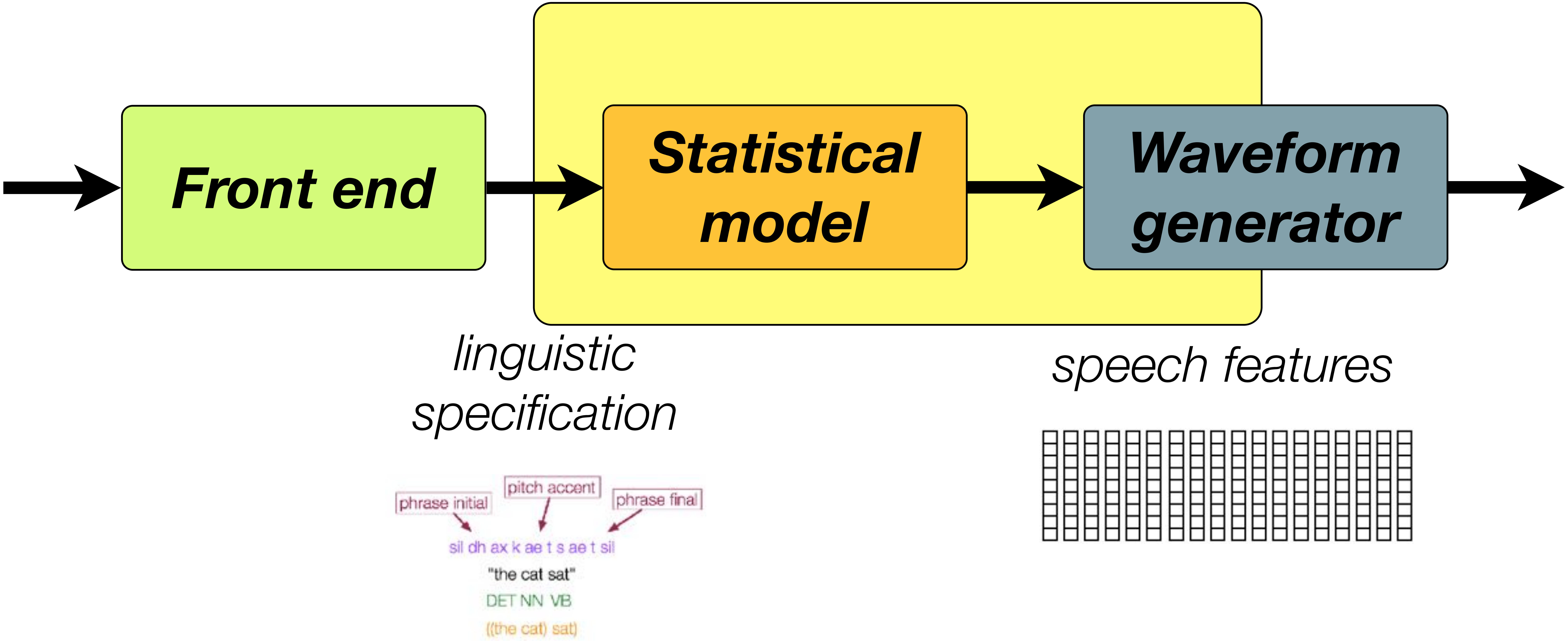
The end-to-end problem



"Regression only"



Generate lower-level speech features (e.g. power spectrum)



Why exactly were we using a vocoder anyway?

- Separate source and filter
- Filter can be very compactly parameterised (e.g., Mel cepstrum)
- For waveform reconstruction, we **do not need to provide phase**
  - The periodic source signal (e.g., pulse train) has phase structure
  - Make some simplifying assumption about the filter's phase

Predict spectrum: magnitude *only*

- e.g., "Tacotron" (Wang & 13 other authors, Interspeech 2017)
- Generate a spectrogram (i.e., sequence of **magnitude** spectra)
- Do not predict **phase**
  - Therefore, to create a waveform, phase has to be "recovered"
  - e.g., Griffin-Lim algorithm, or one of several variants on that
- Post-processing is required to reduce highly-audible phase-related artefacts in the waveform inferred using Griffin-Lim

Predict spectrum: magnitude *and* phase

to appear in Proc Interspeech 2017

Direct Modelling of Magnitude and Phase Spectra for Statistical Parametric Speech Synthesis

Felipe Espic, Cassia Valentini-Botinhao, and Simon King

The Centre for Speech Technology Research (CSTR), University of Edinburgh, UK

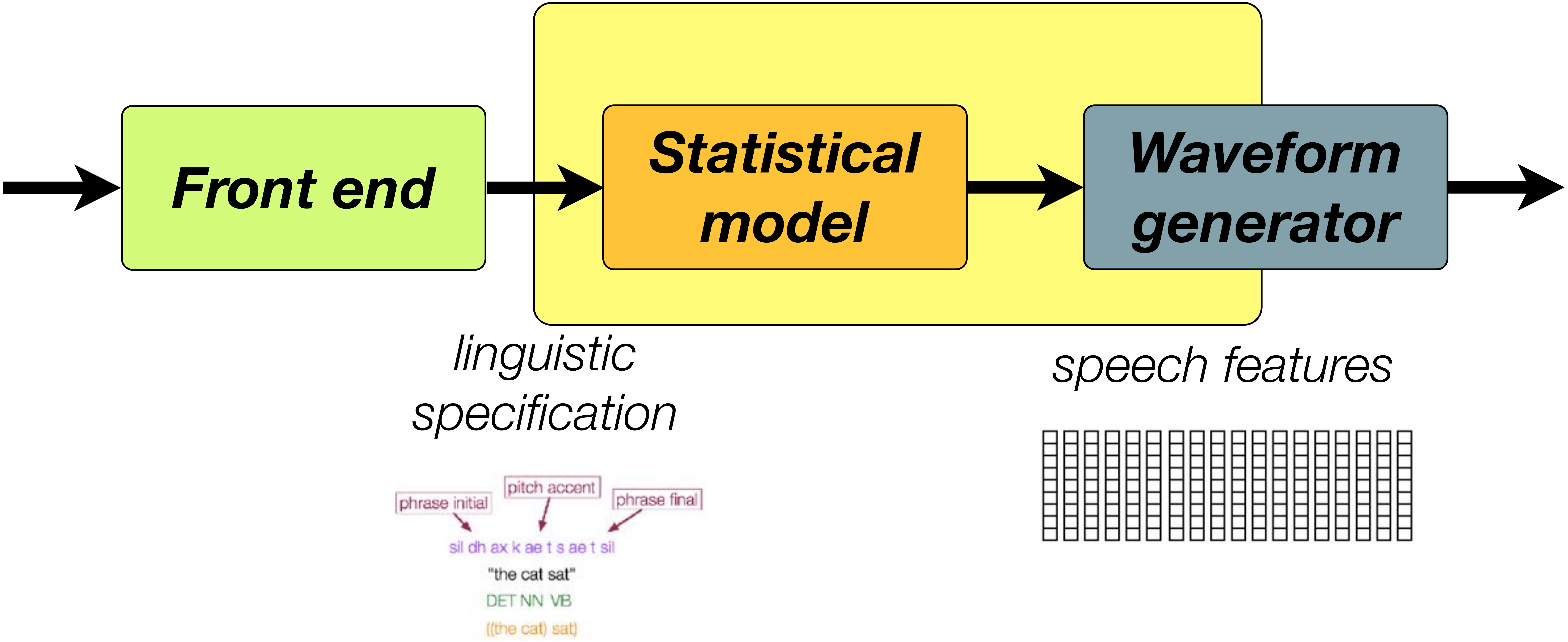
felipe.espic@ed.ac.uk, cvbotinh@inf.ed.ac.uk, Simon.King@ed.ac.uk

Abstract

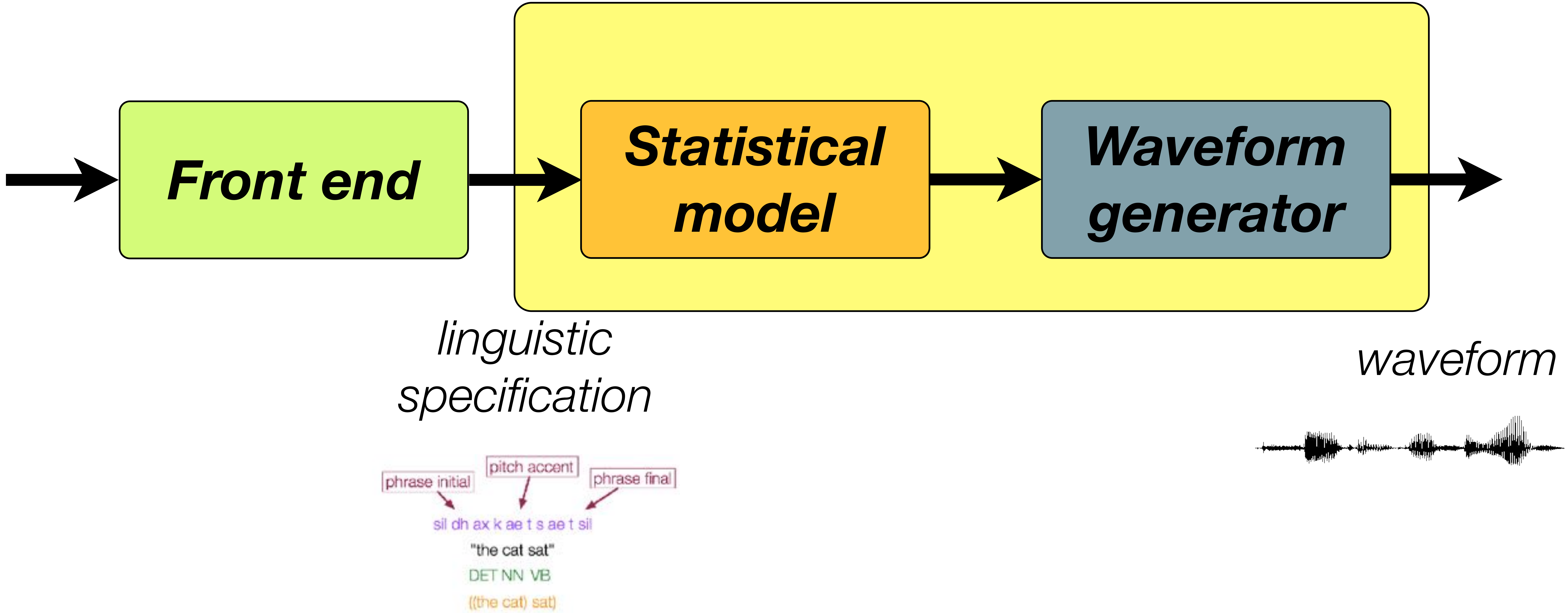
We propose a simple new representation for the FFT spectrum tailored to statistical parametric speech synthesis. It consists of four feature streams

Recently, we proposed a new waveform generation method for text-to-speech (TTS) in which synthetic speech is generated by modifying the fundamental frequency and spectral envelope of a natural speech sample to match values predicted by a model

Generate lower-level speech features (e.g. power spectrum)



Generate the waveform itself



Wavenet

- *"Researchers usually avoid modelling raw audio because it ticks so quickly: typically 16,000 samples per second or more, with important structure at many time-scales."*
  - No, that's not the **main reason** that most approaches do not deal directly with sampled (digital) speech waveforms.

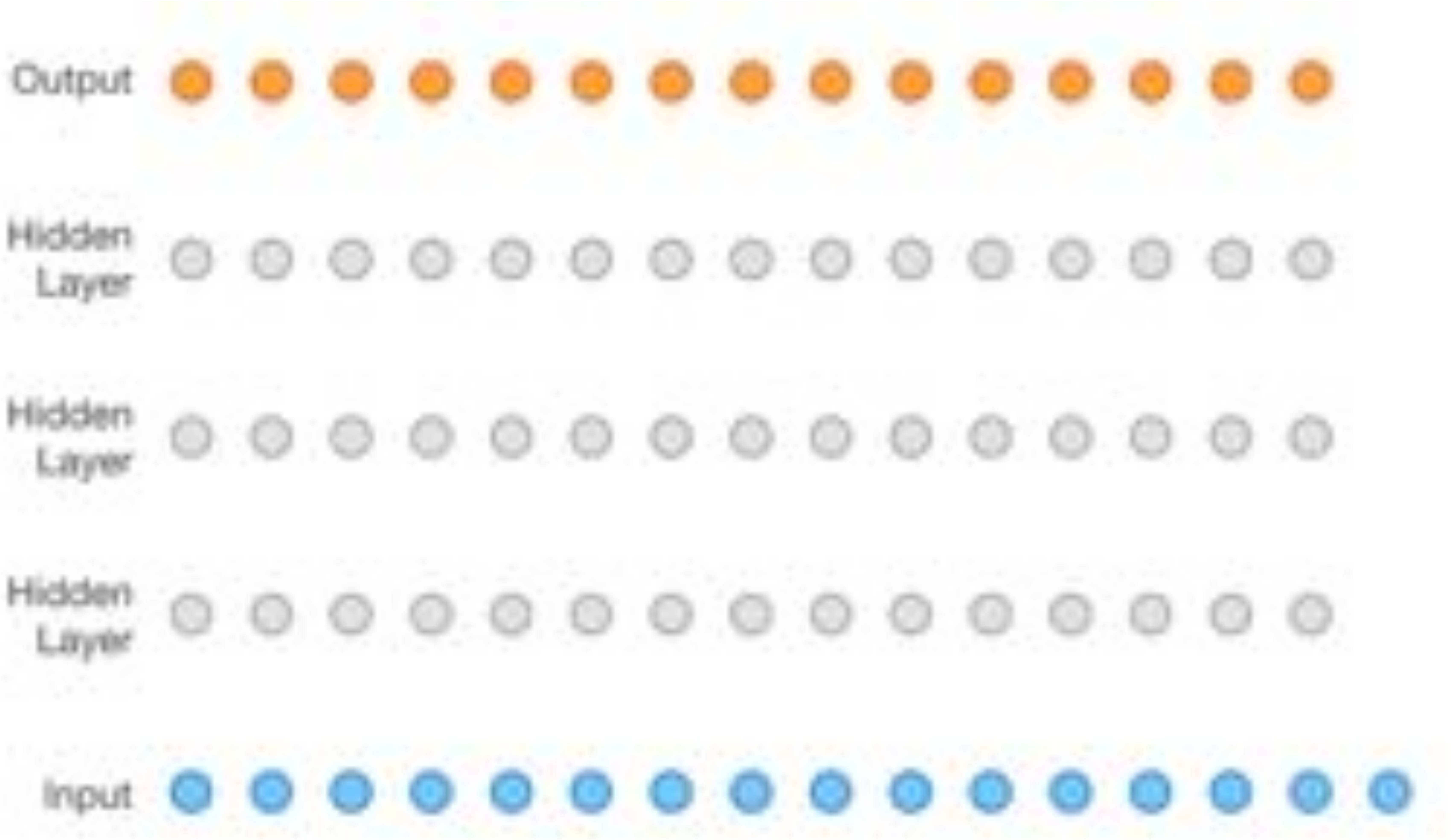
- *"Building a completely autoregressive model, in which the prediction for every one of those samples is influenced by all previous ones (in statistics-speak, each predictive distribution is conditioned on all previous observations), is clearly a challenging task."*
  - **Autoregressive** models with a **fixed** order are widespread, and have been in use since the 1960s : linear predictive coding (LPC).

Quotes are from https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Wavenet

- *"WaveNet…has none of the [common] assumptions [about speech signals]. It incorporates almost no prior knowledge about audio signals"*
  - Is it such a great idea to **disregard** almost everything we (think we) know about speech signals?
  - Discuss (later) !

Quotes are from arXiv:1609.03499 (not peer reviewed)


Output
Hidden Layer
Hidden Layer
Hidden Layer
Input

Quantisation (which introduces *quantisation noise*)

time

Alternative and/or advanced Neural Network techniques

- network architectures
- avoiding vocoding
- <u>avoiding the front end</u>

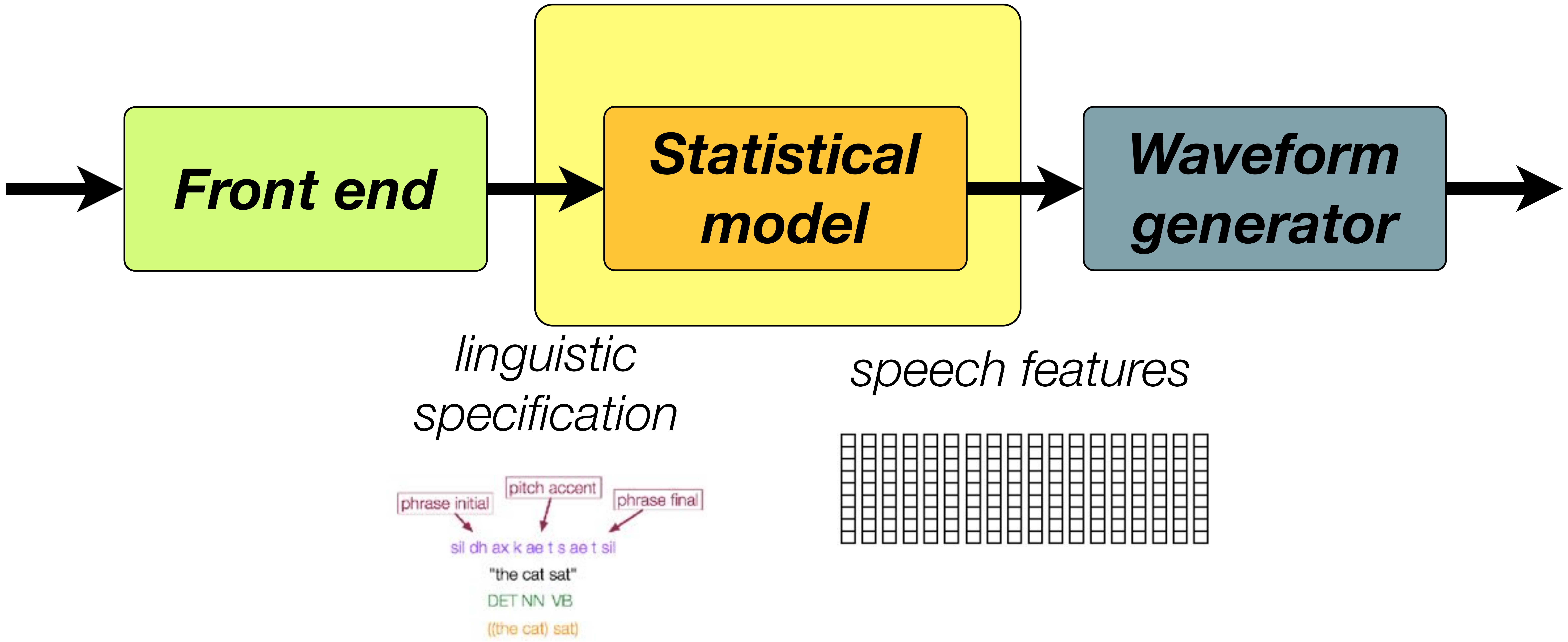
The end-to-end problem

text

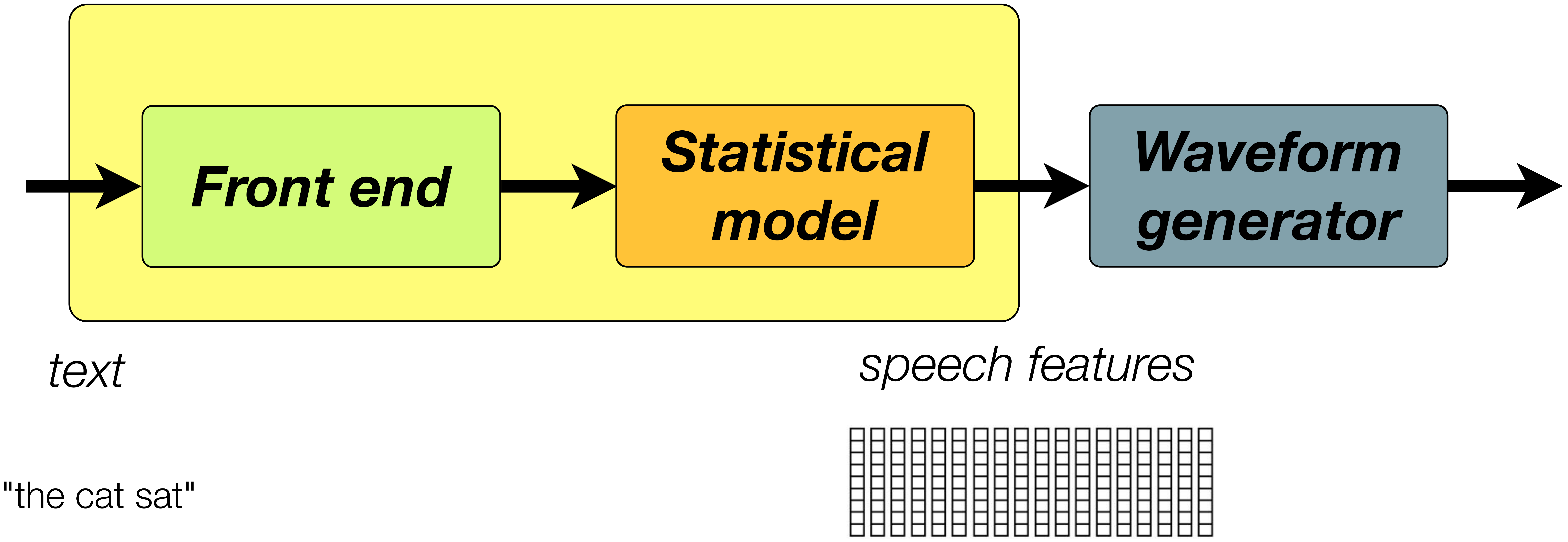
waveform

"the cat sat"

“Regression only”



"Avoiding the front end"



Raw text input ?

- e.g., "Tacotron" (Wang & 13 other authors, Interspeech 2017)

- *"Modern text-to-speech (TTS) pipelines are complex (Taylor, 2009)"*
  - True - but the Tacotron is hardly "simple" or "easy to build"

- *"[the front-end] components are based on extensive domain expertise"*
  - Definitely a problem for low-resource languages, but do we really want to disregard all available domain expertise in high-resource languages?

Quotes are from arXiv:1703.10135v2
(presumed to be pre-submission version of Interspeech paper)

Raw text input ?

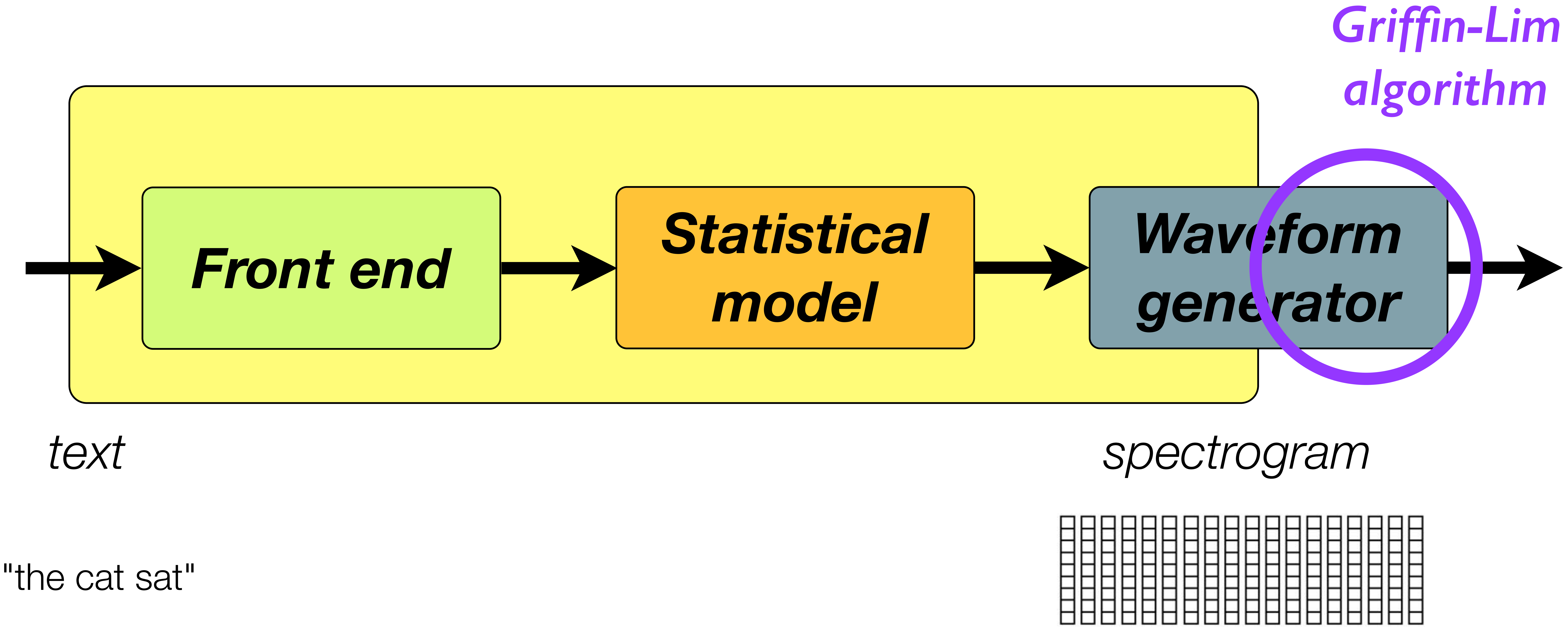
- e.g., "Tacotron" (Wang & 13 other authors, Interspeech 2017)

- *"errors from each component may compound"*
  - Agreed

- *"The complexity … leads to substantial engineering efforts when building a new system"*
  - How many people did it take to build the Tacotron !?

Quotes are from arXiv:1703.10135v2
(presumed to be pre-submission version of Interspeech paper)

The Tacotron



What next?

- Expect many, many papers on DNN synthesis at Interspeech
- Especially
  - "end-to-end"
  - "avoiding vocoding"

- Front-end issues probably harder to address with Deep Learning
  - but isolated parts of the problem certainly can be (e.g., LTS / G2P)