Welcome!

Simon King

Simon King

- Prof. of Speech Processing
- Director of CSTR
- Co-author of Festival

- CSTR website: **`www.cstr.ed.ac.uk`**

- Teaching website: **`speech.zone`**

Oliver Watts

- Research Fellow in CSTR
- Author of the Ossian framework for building TTS front ends
- Ossian website: `simple4all.org`

Srikanth Ronanki

- PhD student in CSTR
- Maintainer & co-author of Merlin
- Website: `srikanthr.in`

Felipe Espic

- PhD student in CSTR
- Expert in signal processing, especially speech analysis and waveform generation
- Website: `felipeespic.com`

Zhizheng Wu

- Creator of Merlin
- Now with Apple
- Website: `zhizheng.org`

Tutorial coverage

- PART 1 - Text-to-speech, in a nutshell

- PART 2 - Building a system using
  - Ossian for the front end
  - Merlin for the acoustic model
  - WORLD vocoder

- PART 3 - Extensions that are (or could easily be) achievable with Merlin

References

- This tutorial covers the **ideas** of many people, and not just those of the presenters
- To keep the slides clear and simple, **citations are not included**
- Instead, there is a brief **reading list** at the end, arranged by topic

Agenda

	Topic	Presenter
PART 1	**From text to speech**	**Simon King**
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

From text to speech

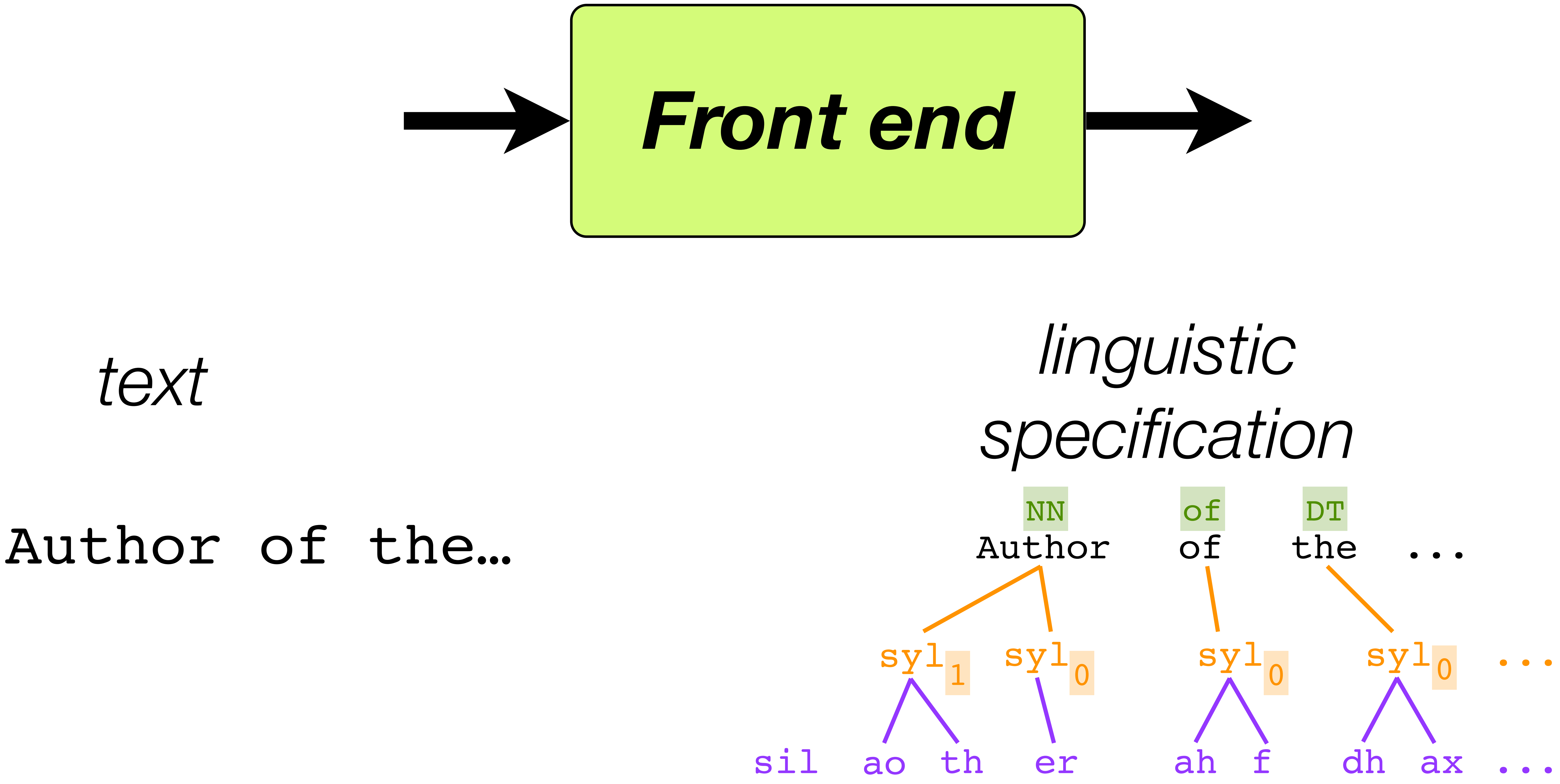
Simon King

The classic two-stage pipeline of unit selection

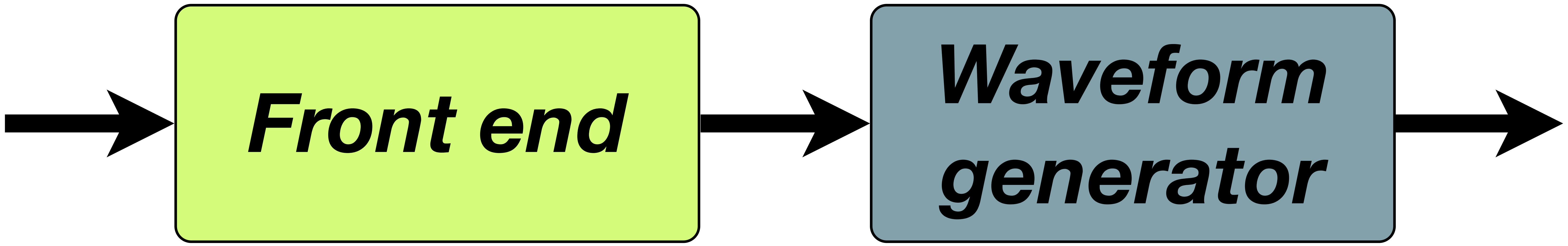
text

```
Author of the…
```

The classic two-stage pipeline of unit selection



The classic two-stage pipeline of unit selection



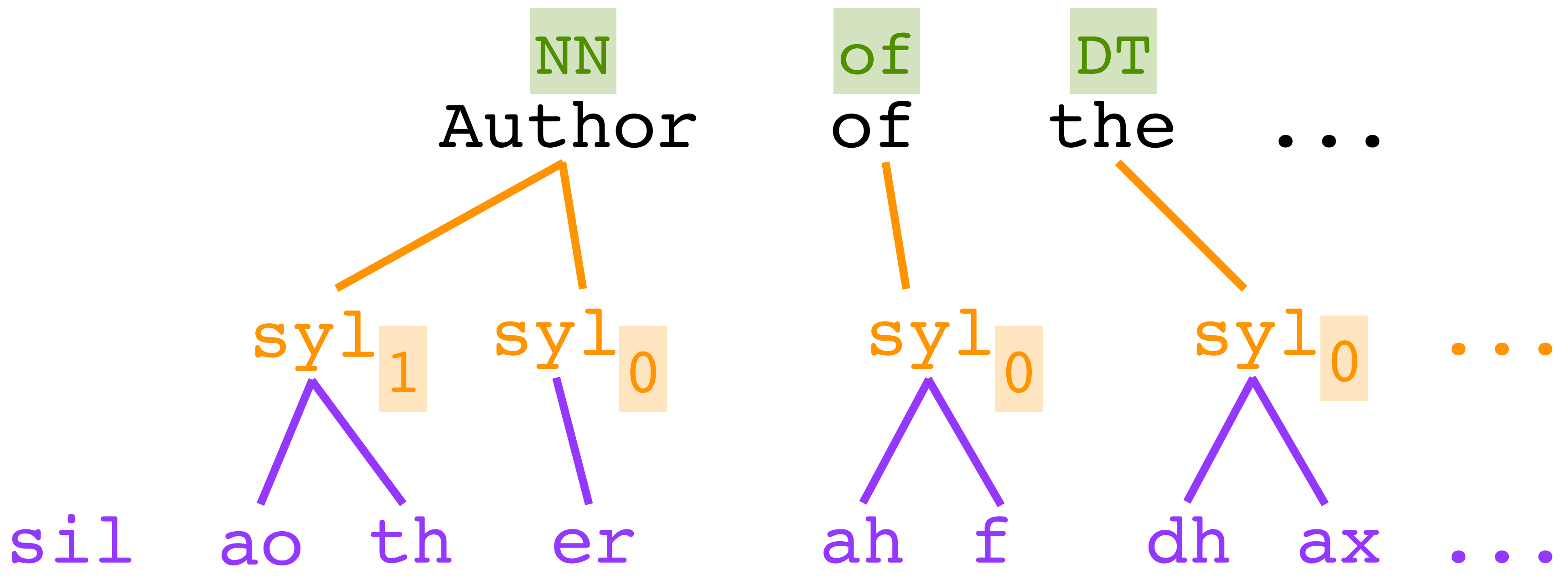
Front end

Waveform generator

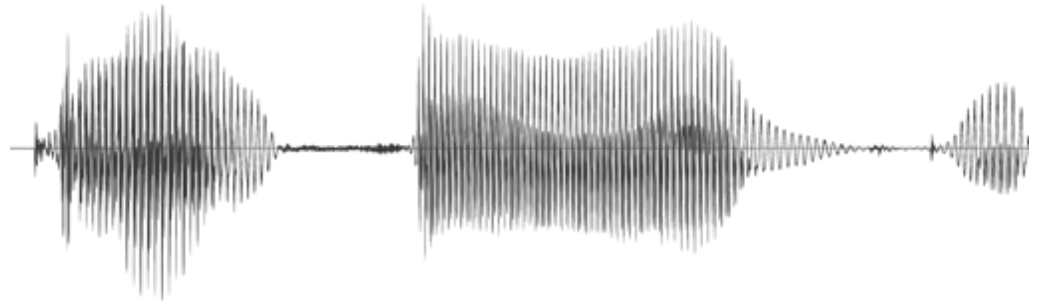
text

`Author of the…`

linguistic specification



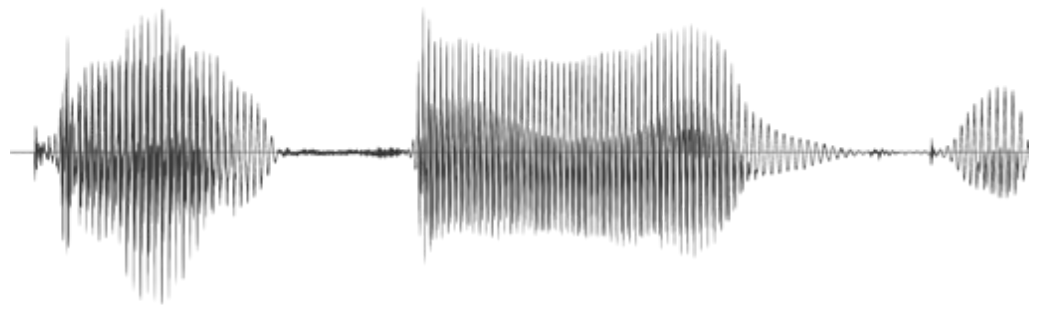
waveform



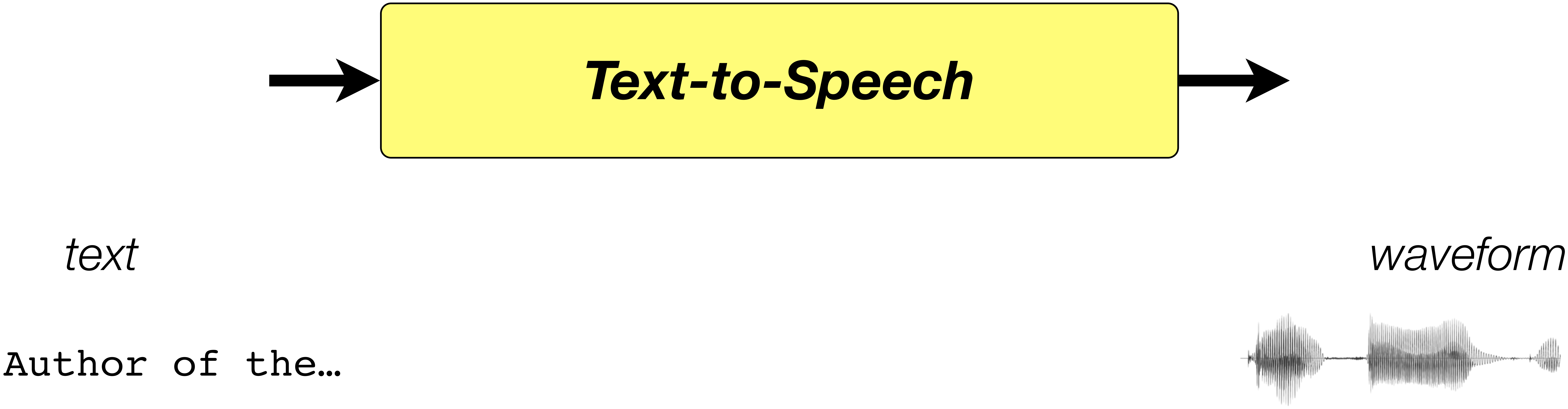
The end-to-end problem we want to solve

text

`Author of the…`



The end-to-end problem we want to solve



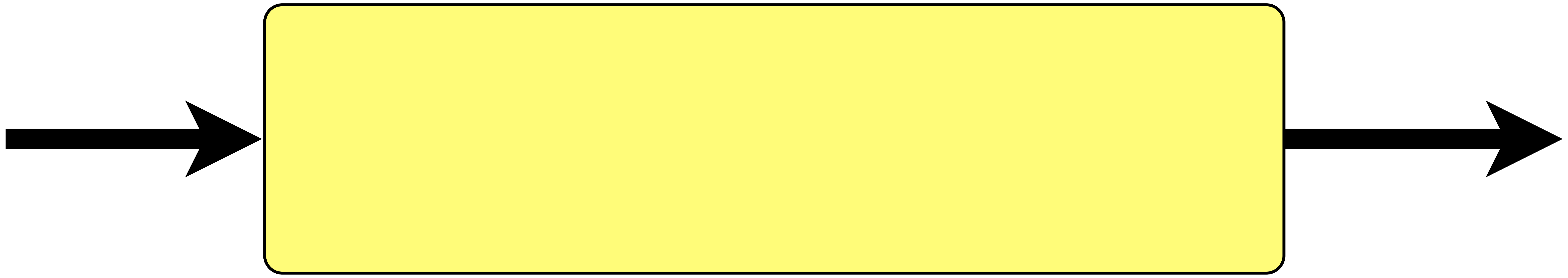
Text-to-Speech

text

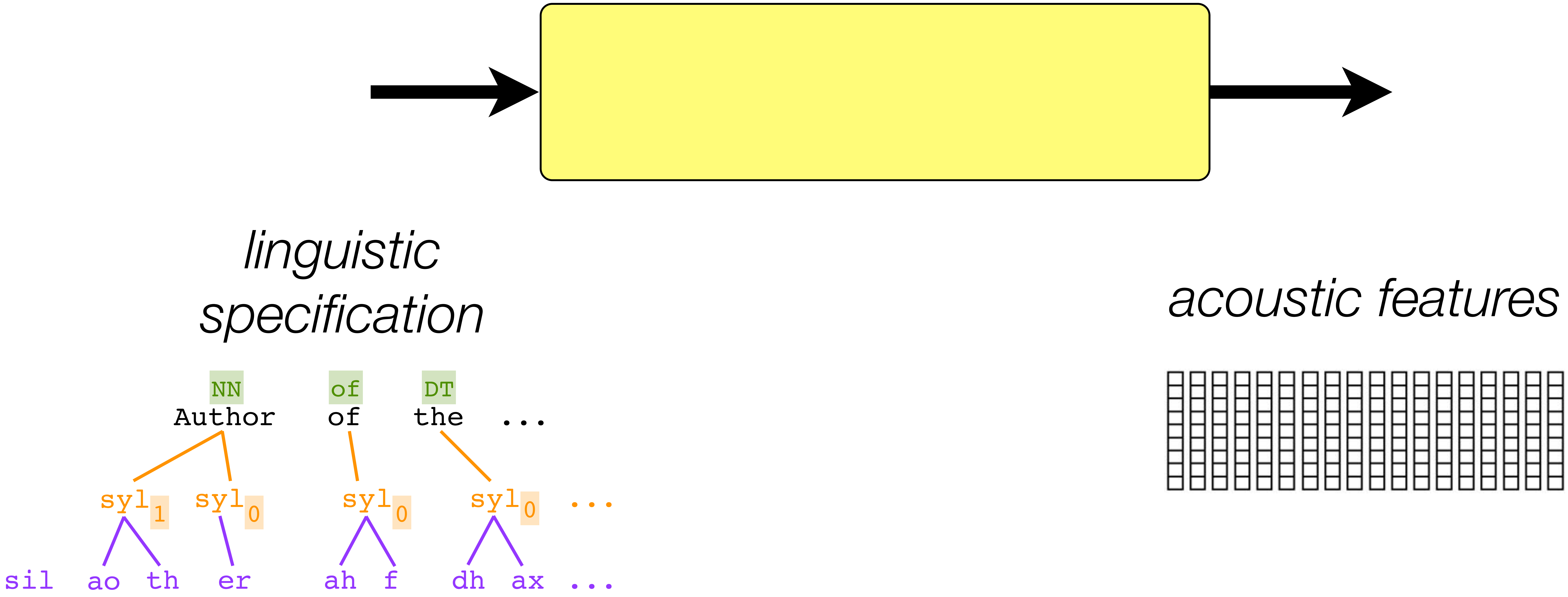
waveform

`Author of the…`

A problem we can actually solve with machine learning



A problem we can actually solve with machine learning



linguistic specification

acoustic features

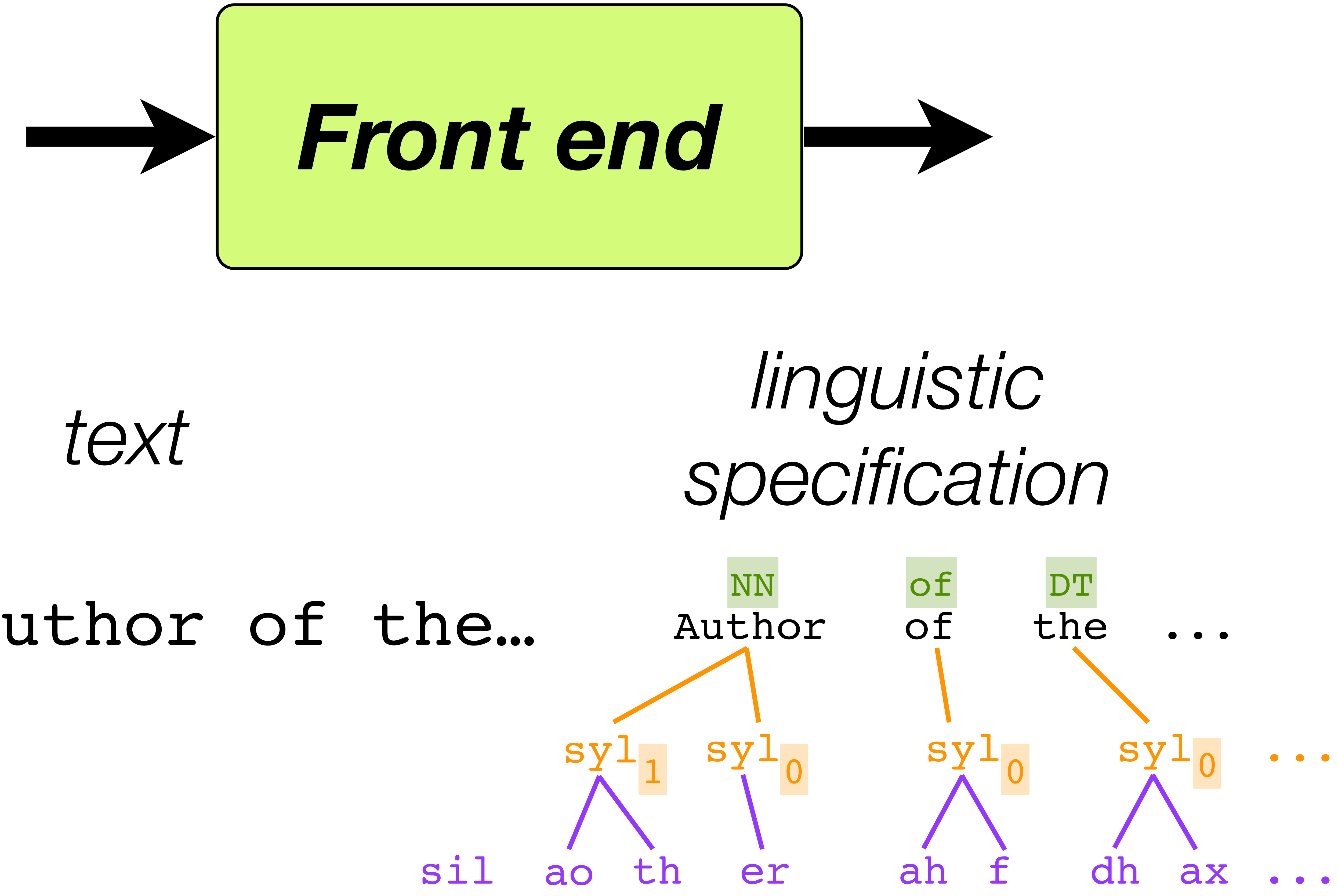
The classic three-stage pipeline of statistical parametric speech synthesis

The classic three-stage pipeline of statistical parametric speech synthesis

text

```
Author of the…
```

The classic three-stage pipeline of statistical parametric speech synthesis



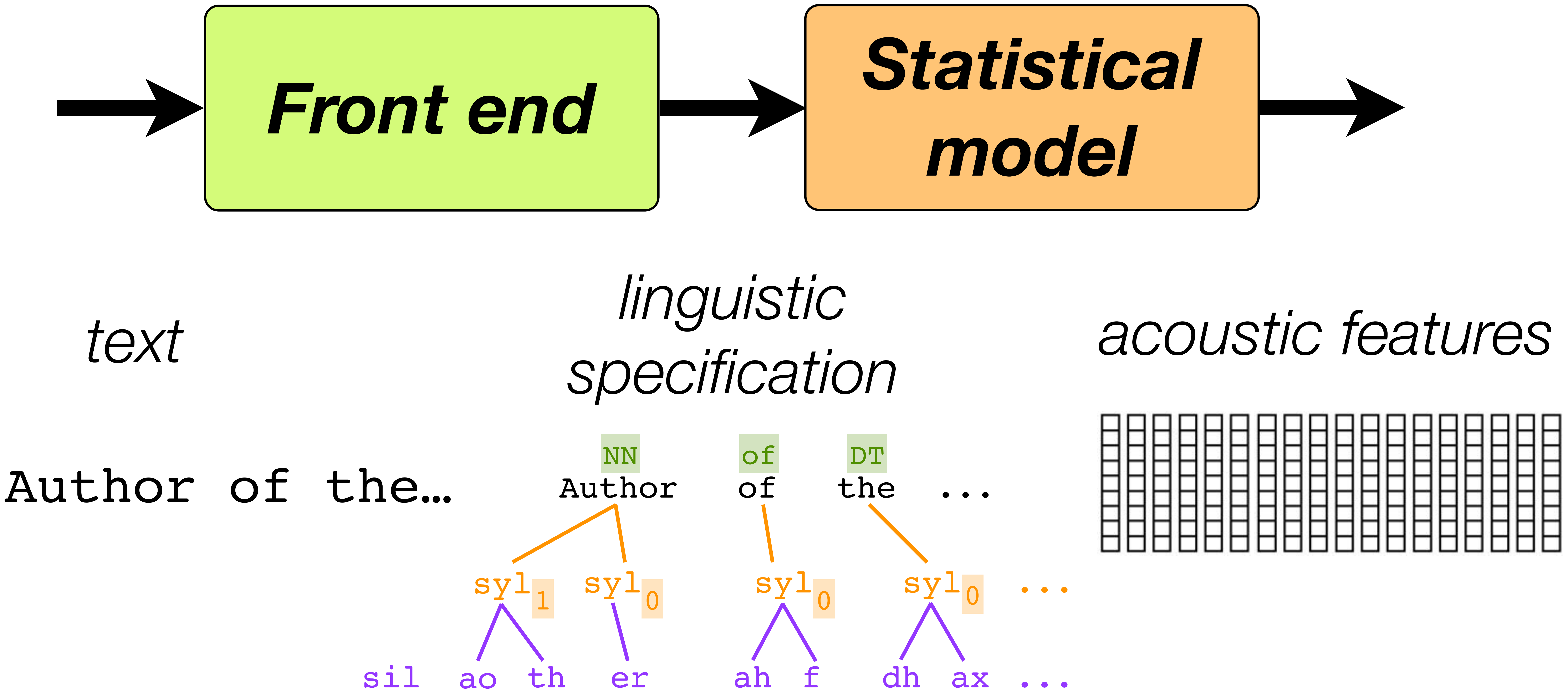
Front end

text

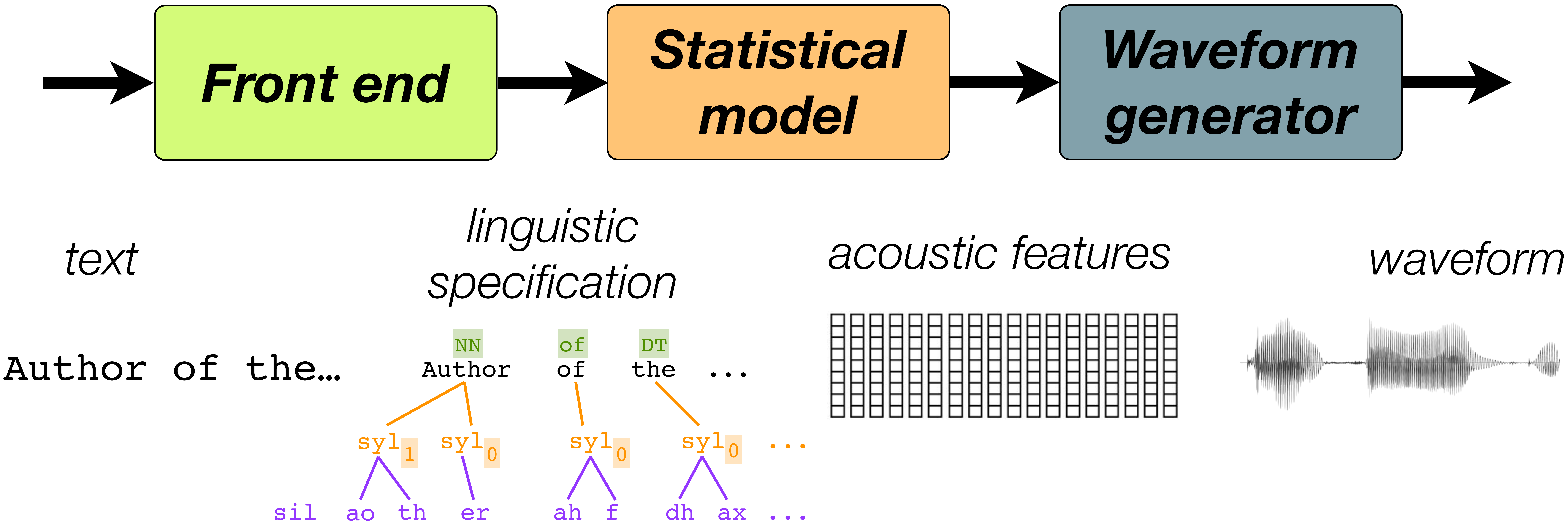
linguistic specification

Author of the…

The classic three-stage pipeline of statistical parametric speech synthesis



The classic three-stage pipeline of statistical parametric speech synthesis



Front end

Statistical model

Waveform generator

text

linguistic specification

acoustic features

waveform

Author of the…

The classic three-stage pipeline of statistical parametric speech synthesis

Statistical model

text

`Author of the…`

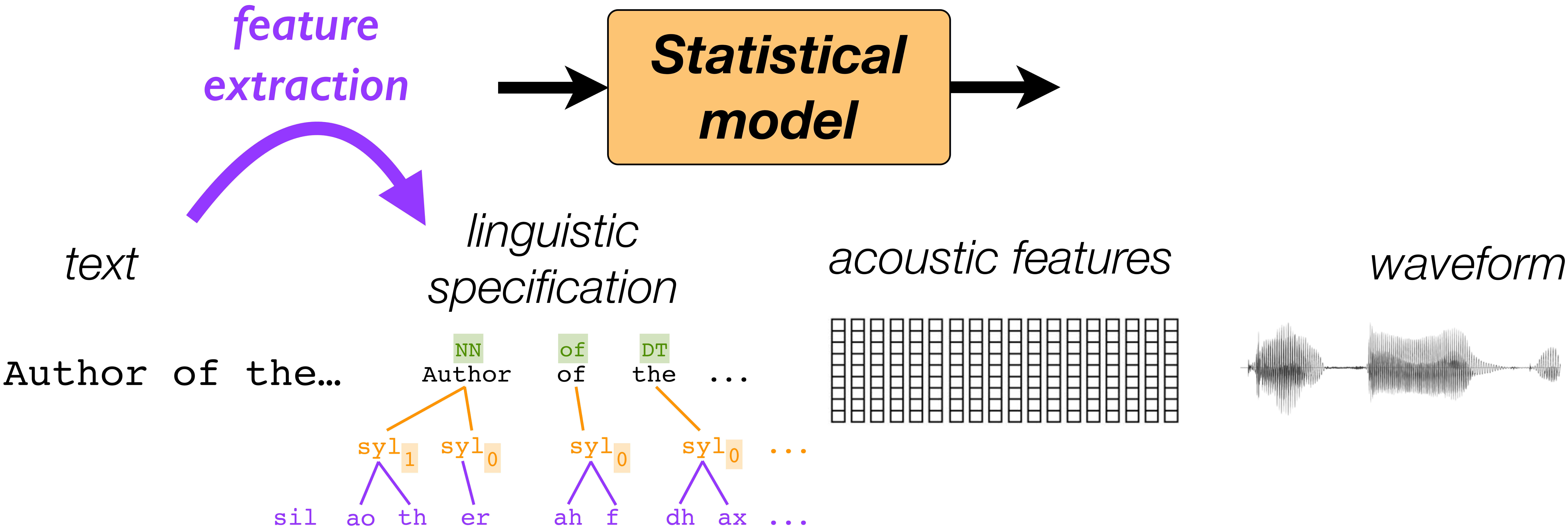
linguistic specification

```
NN      of   DT
Author  of   the  ...
syl1  syl0   syl0   syl0   ...
sil ao th er   ah f   dh ax  ...
```

acoustic features

waveform

The classic three-stage pipeline of statistical parametric speech synthesis



feature extraction

Statistical model

text

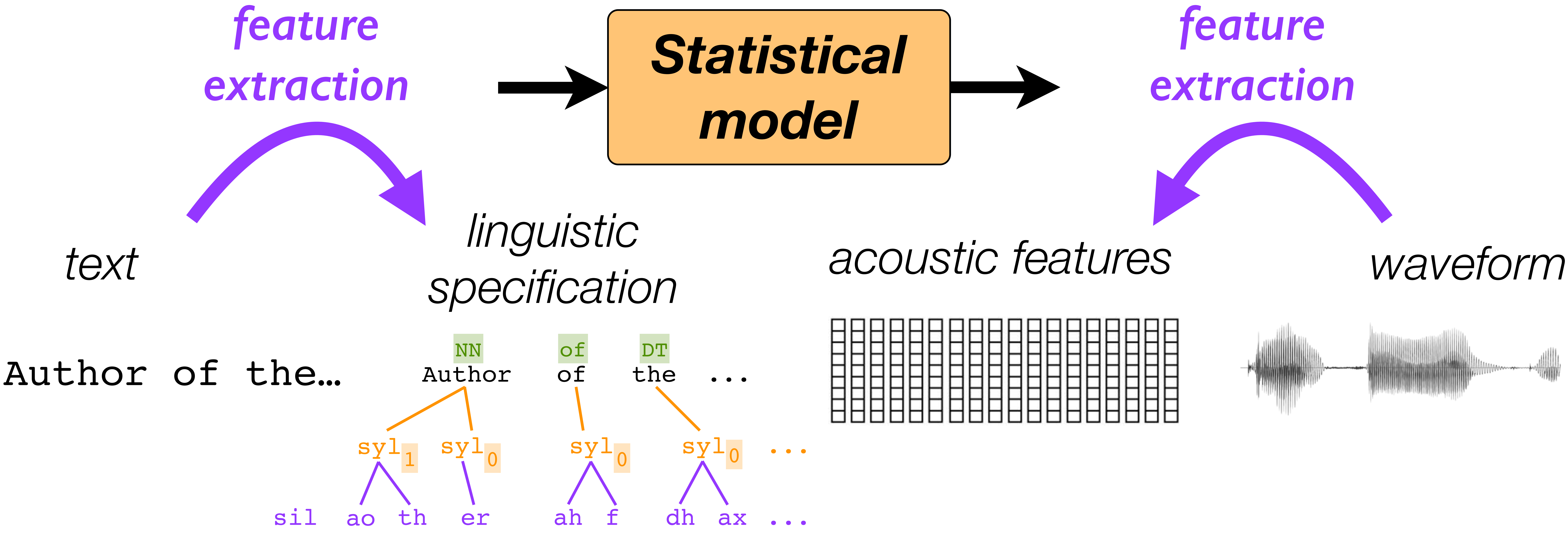
Author of the…

linguistic specification

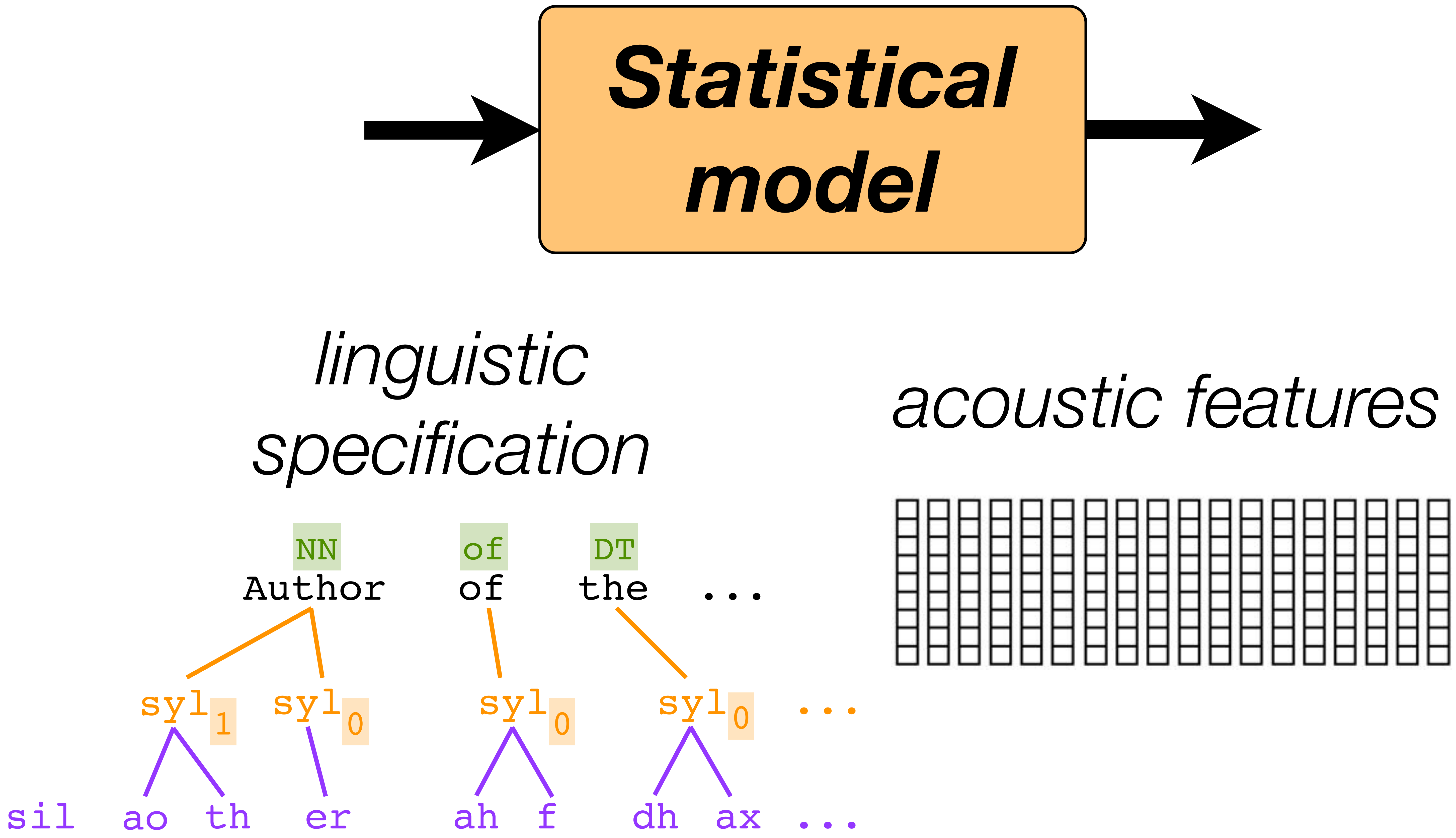
acoustic features

waveform

The classic three-stage pipeline of statistical parametric speech synthesis



The classic three-stage pipeline of statistical parametric speech synthesis

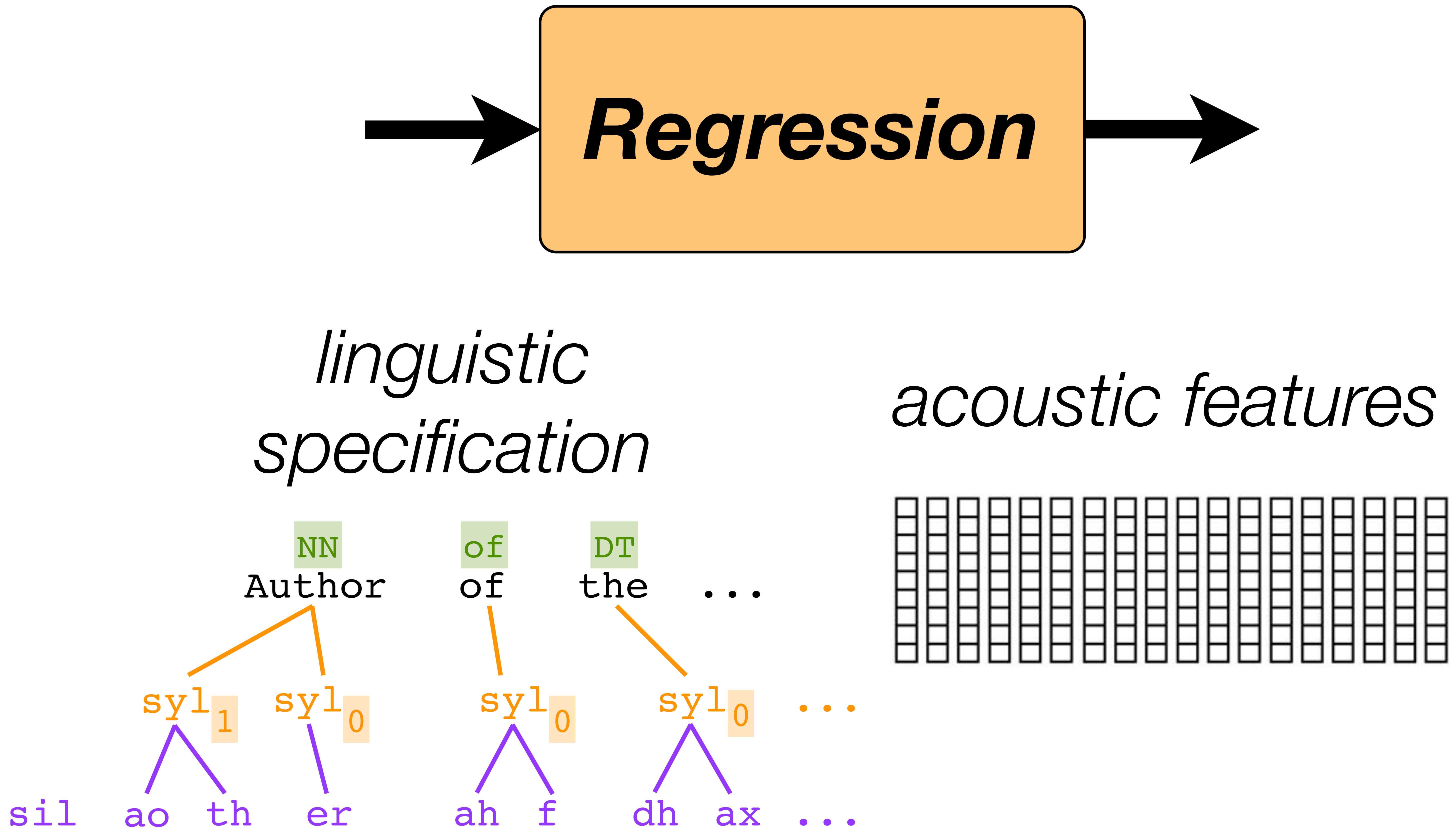


Statistical model

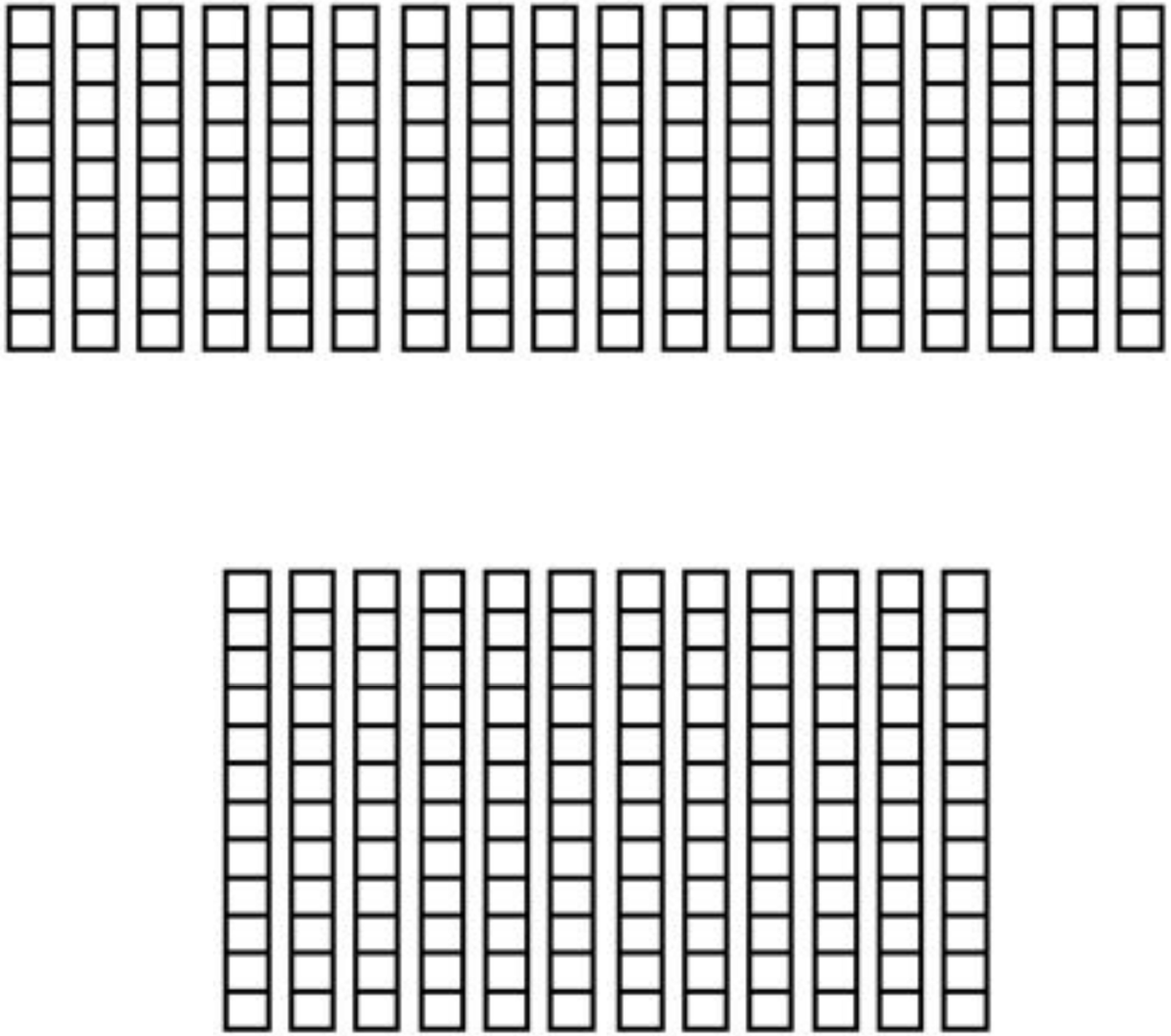
linguistic specification

acoustic features

The classic three-stage pipeline of statistical parametric speech synthesis



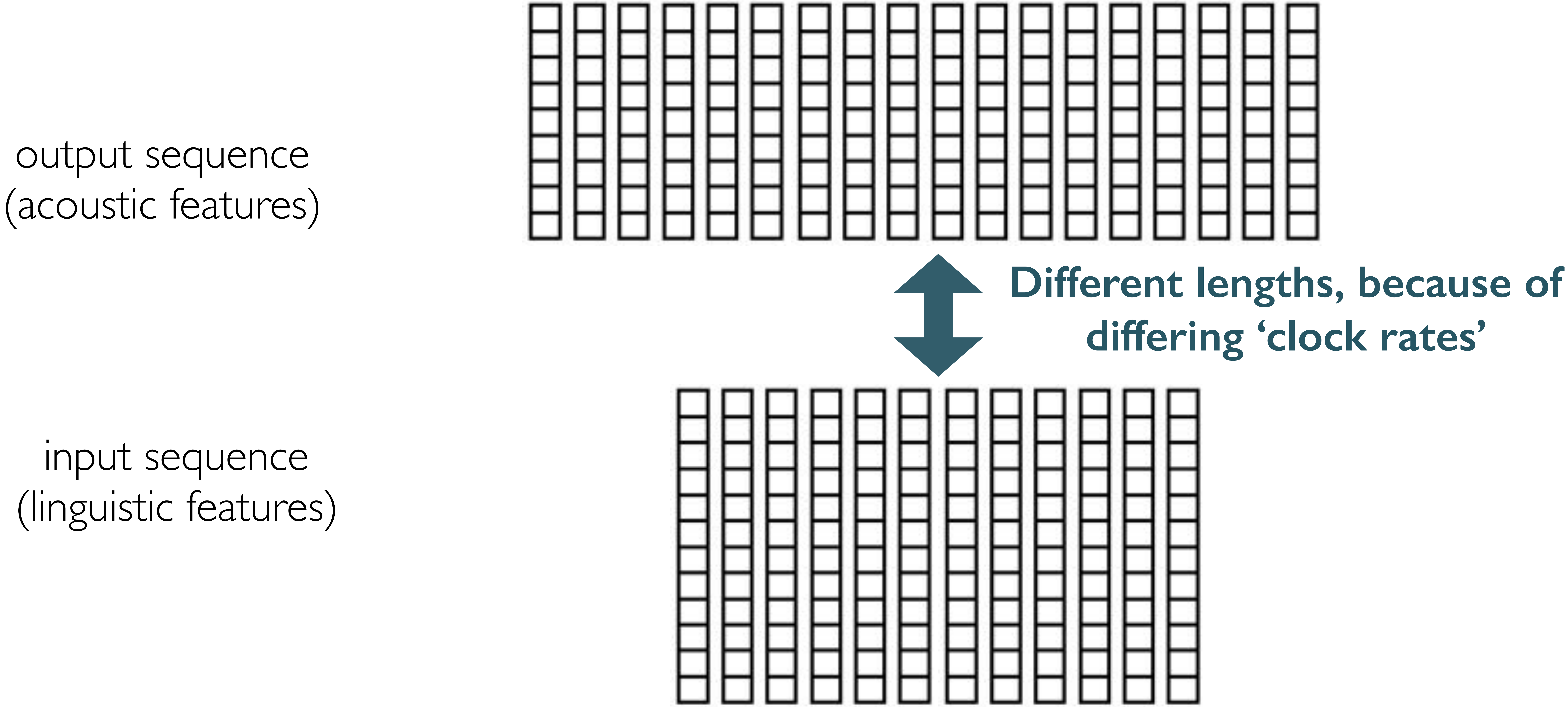
We can describe the core problem as **sequence-to-sequence regression**



output sequence
(acoustic features)

input sequence
(linguistic features)

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



output sequence
(acoustic features)

input sequence
(linguistic features)

Different lengths, because of differing ‘clock rates’

Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system

Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system

Orientation

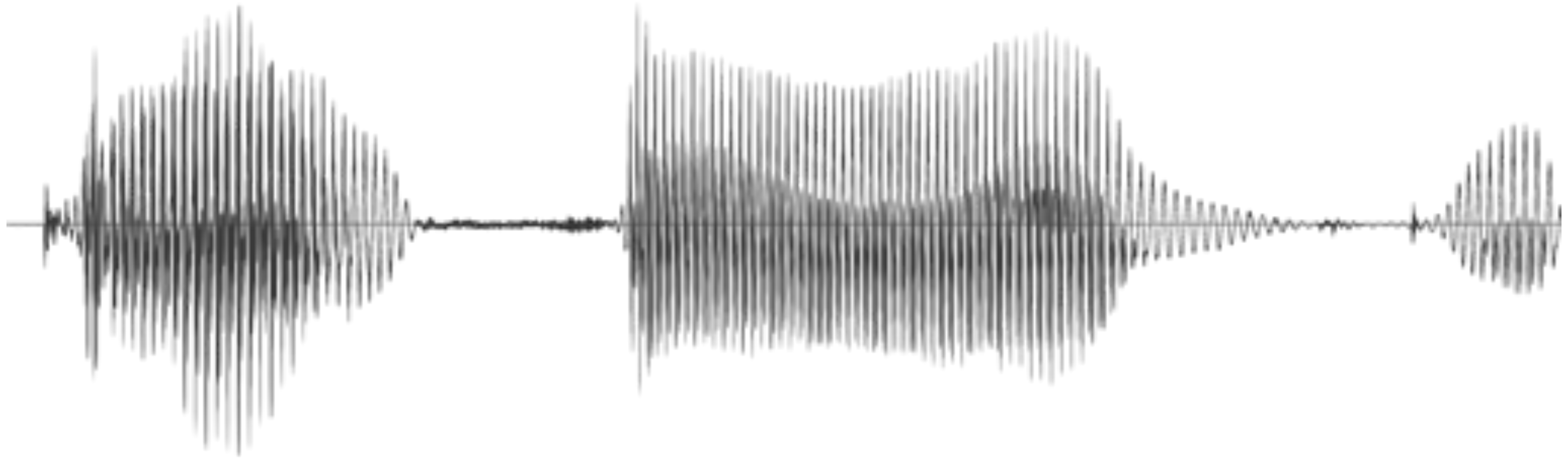
- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system

this is a deliberately **generic** way to talk about TTS

it will make it easier to understand:

- different **methods** for doing regression
- choices of input and output **features**

Orientation

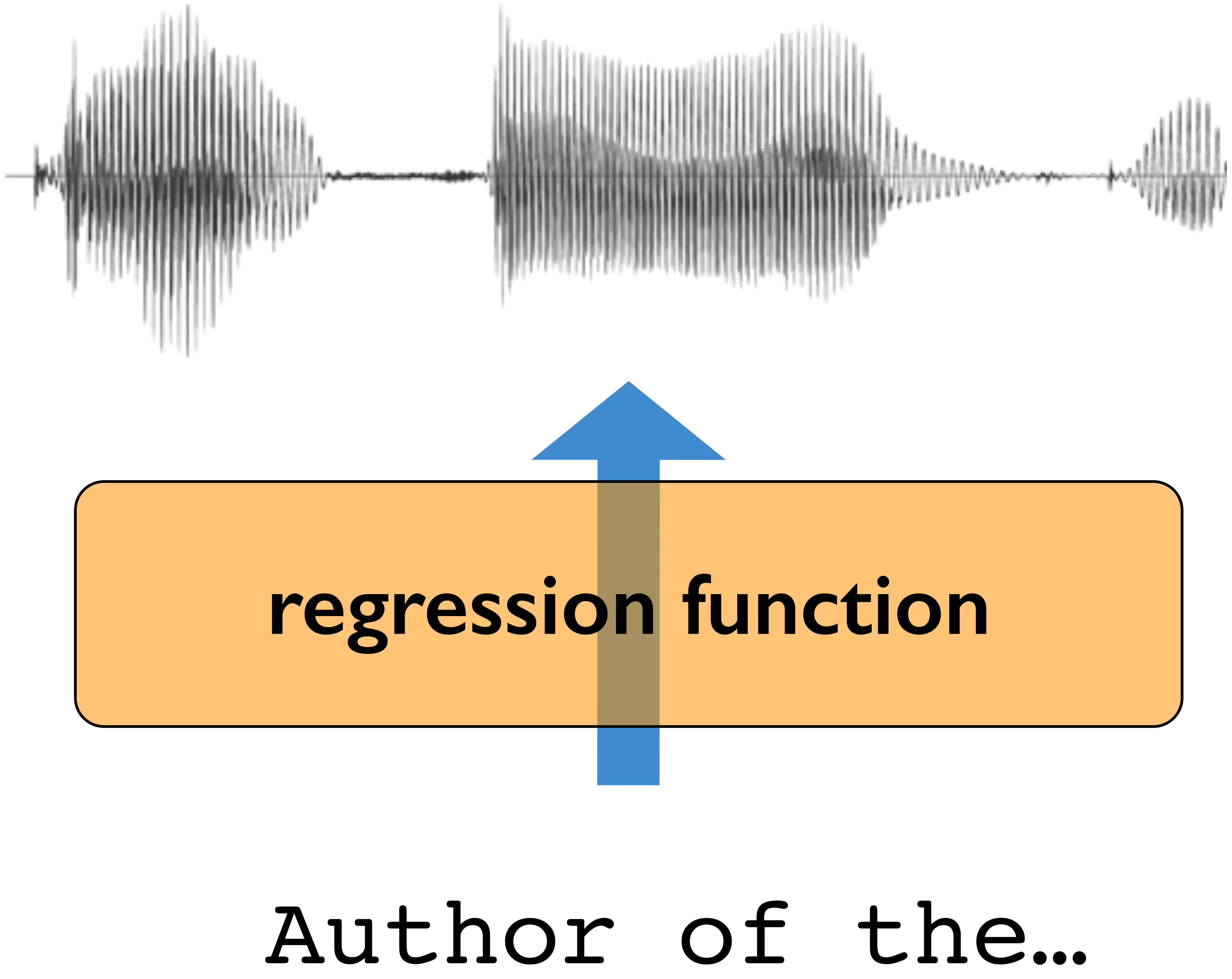


- <u>So far</u>
  - set up the problem of TTS as **sequence-to-sequence regression**
- <u>Next</u>
  - how TTS is done, using a pre-built system
- <u>Later</u>
  - how to build that system

Author of the…

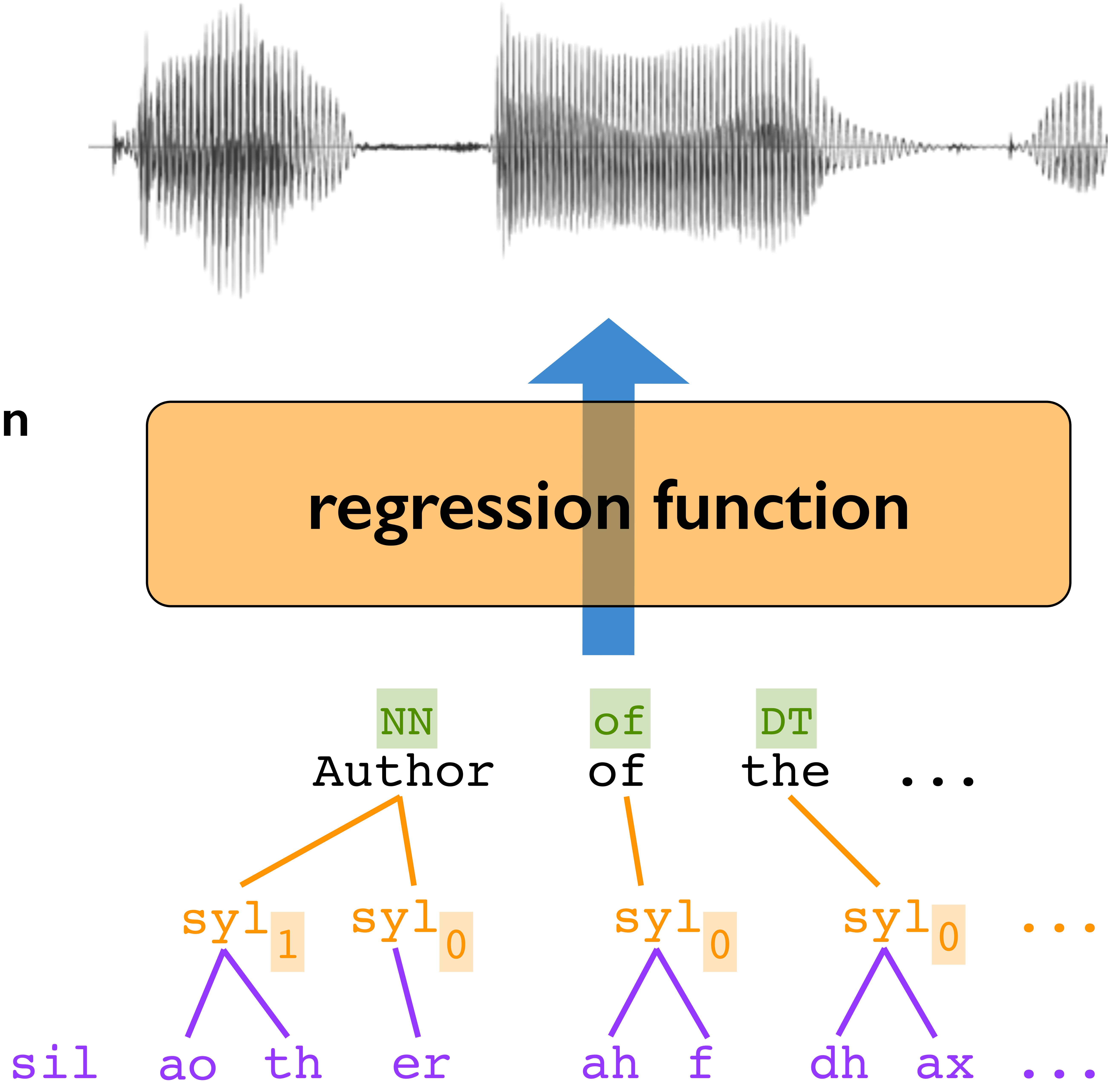
Orientation

- <u>So far</u>
  - set up the problem of TTS as **sequence-to-sequence regression**
- <u>Next</u>
  - how TTS is done, using a pre-built system
- <u>Later</u>
  - how to build that system



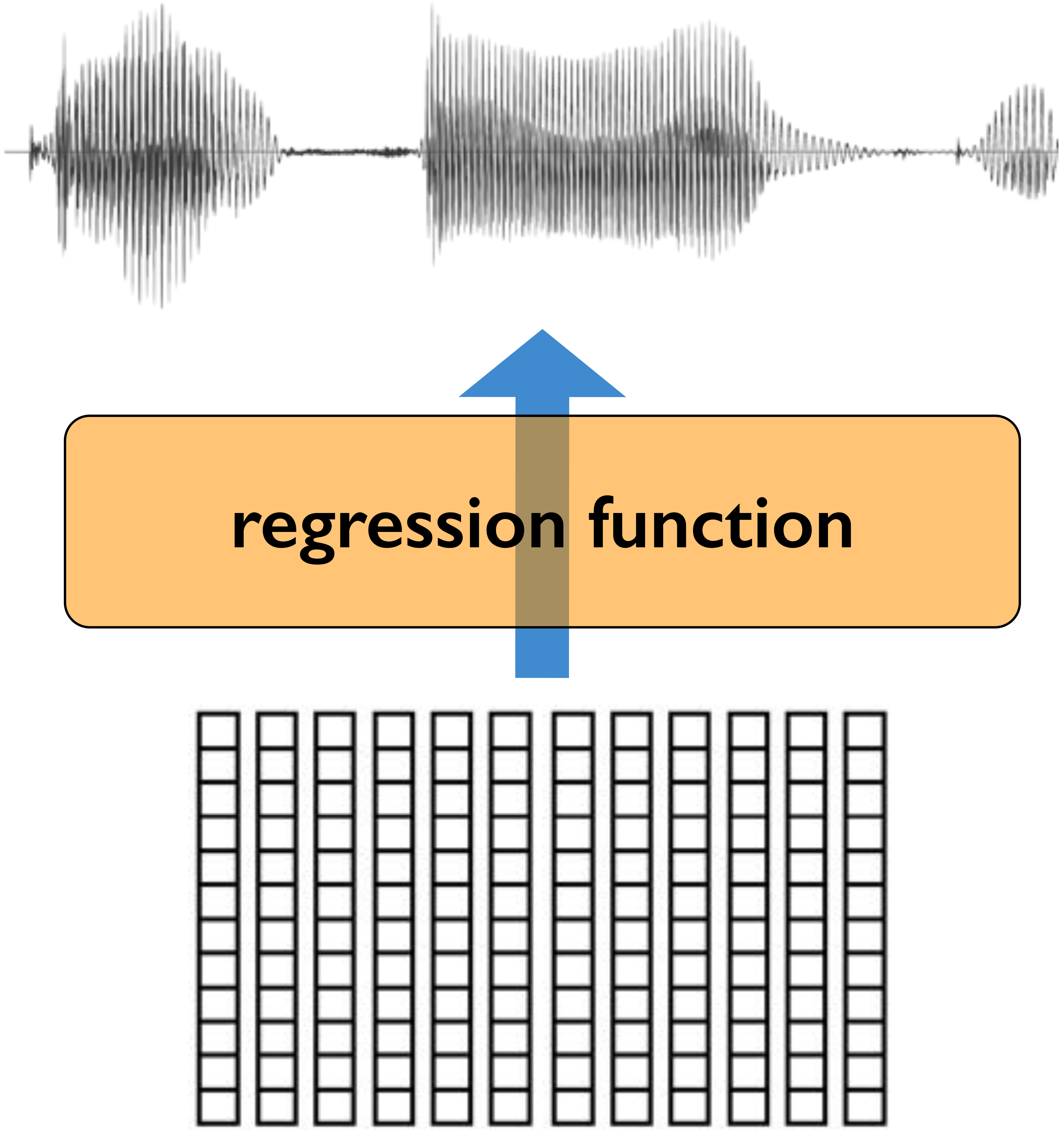
Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system



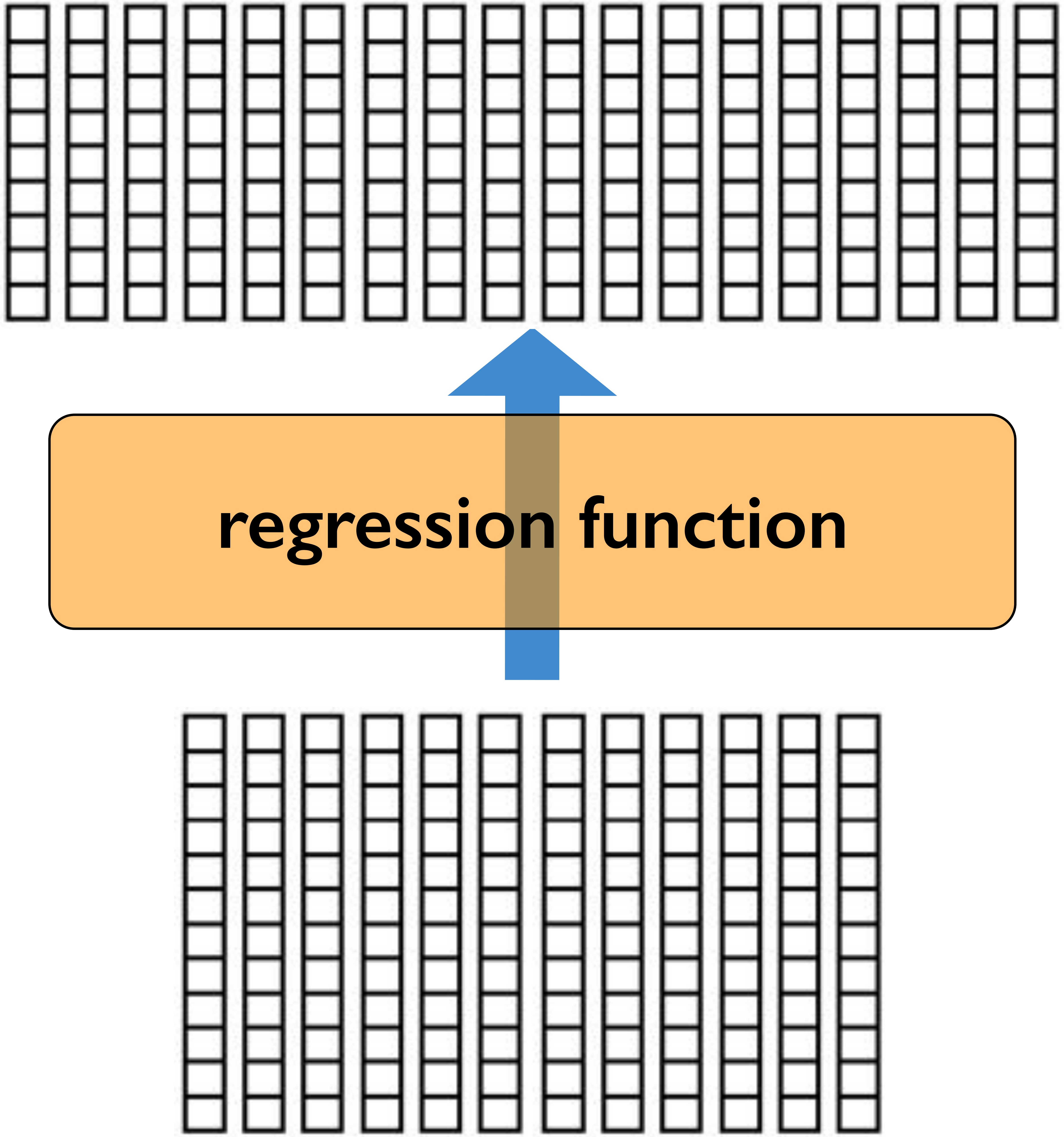
Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system



Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system



regression function

regression function

Number of inputs and outputs is **not to scale** !

Orientation

- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system
- Later
  - how to build that system

this is a deliberately **generic** way to talk about TTS

it will make it easier to understand:

- different **methods** for doing regression
- choices of input and output **features**

Orientation

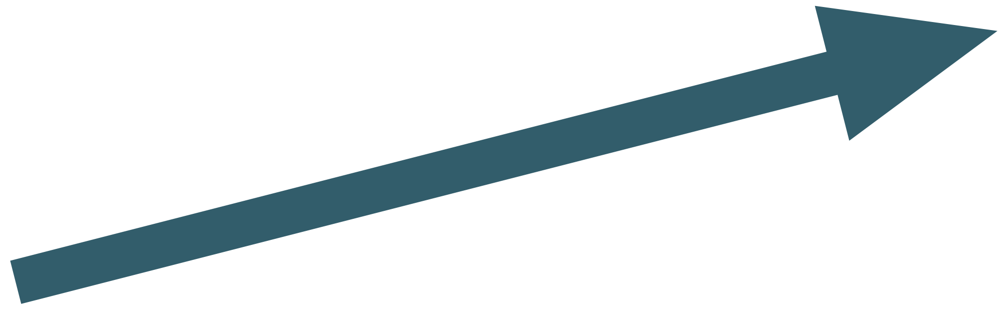
- So far
  - set up the problem of TTS as **sequence-to-sequence regression**
- Next
  - how TTS is done, using a pre-built system

a **quick** run through the complete pipeline, from text input to waveform output

- Later
  - how to build that system

Orientation

- <u>So far</u>
  - set up the problem of TTS as **sequence-to-sequence regression**
- <u>Next</u>
  - how TTS is done, using a pre-built system
- <u>Later</u>
  - how to build that system



a slower, **step-by-step** run through the complete pipeline, concentrating on how to **create** a new system (for any language)

Terminology

- Front end

- Regression

- Waveform generator

Terminology

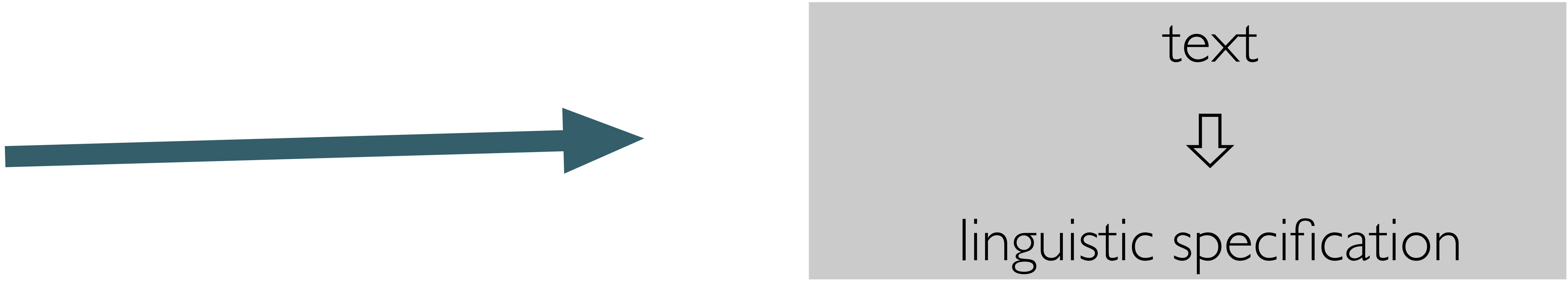
- Front end

- Regression

- Waveform generator

Terminology

- Front end



text

⇩

linguistic specification

- Regression

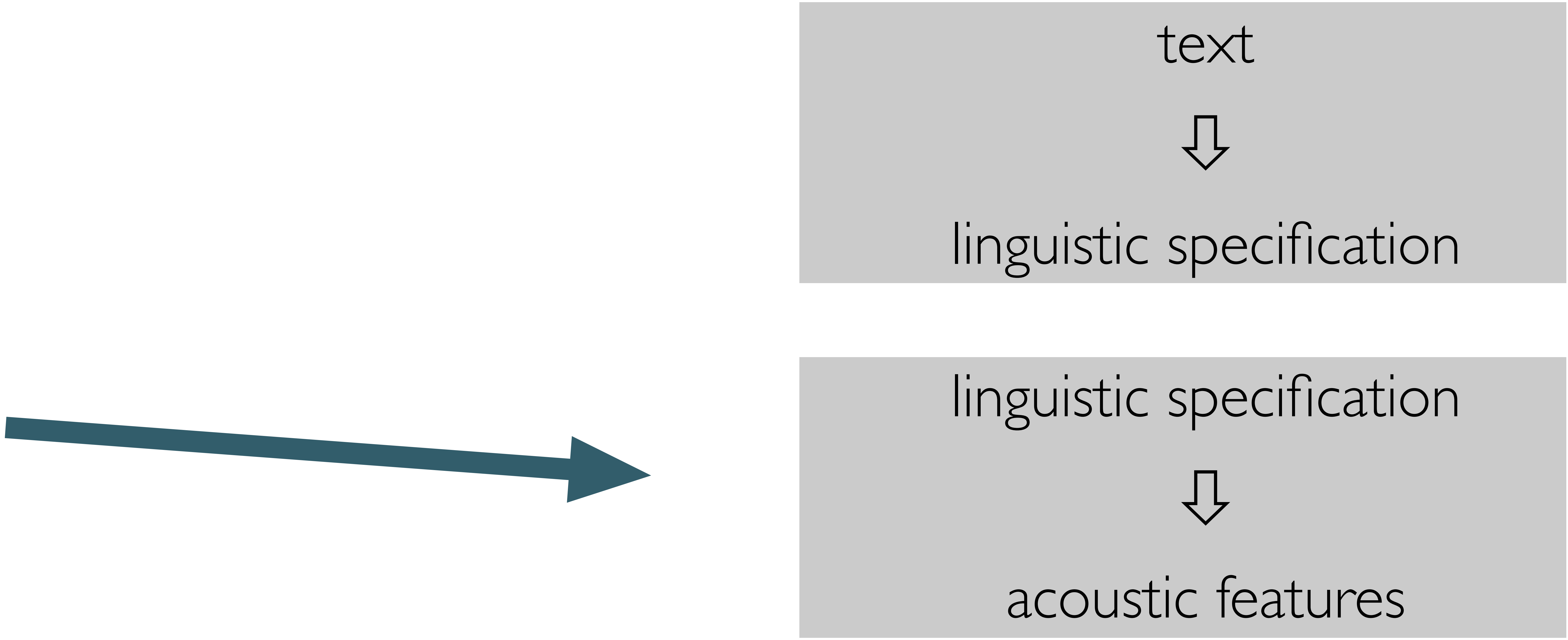
- Waveform generator

Terminology

- Front end

- Regression

- Waveform generator



text

⇩

linguistic specification

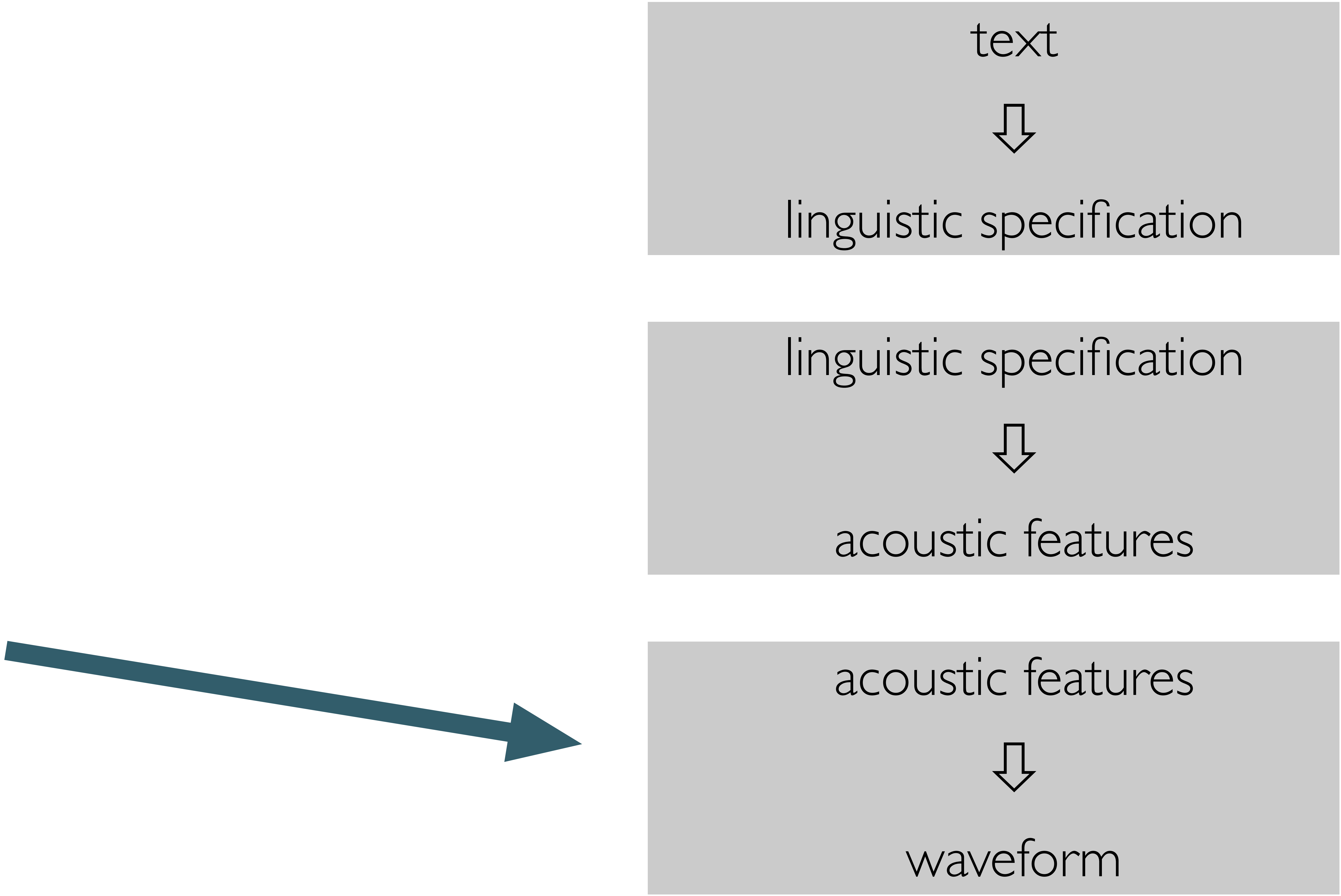
linguistic specification

⇩

acoustic features

Terminology

- Front end
- Regression
- Waveform generator



text ⇩ linguistic specification

linguistic specification ⇩ acoustic features

acoustic features ⇩ waveform

Terminology

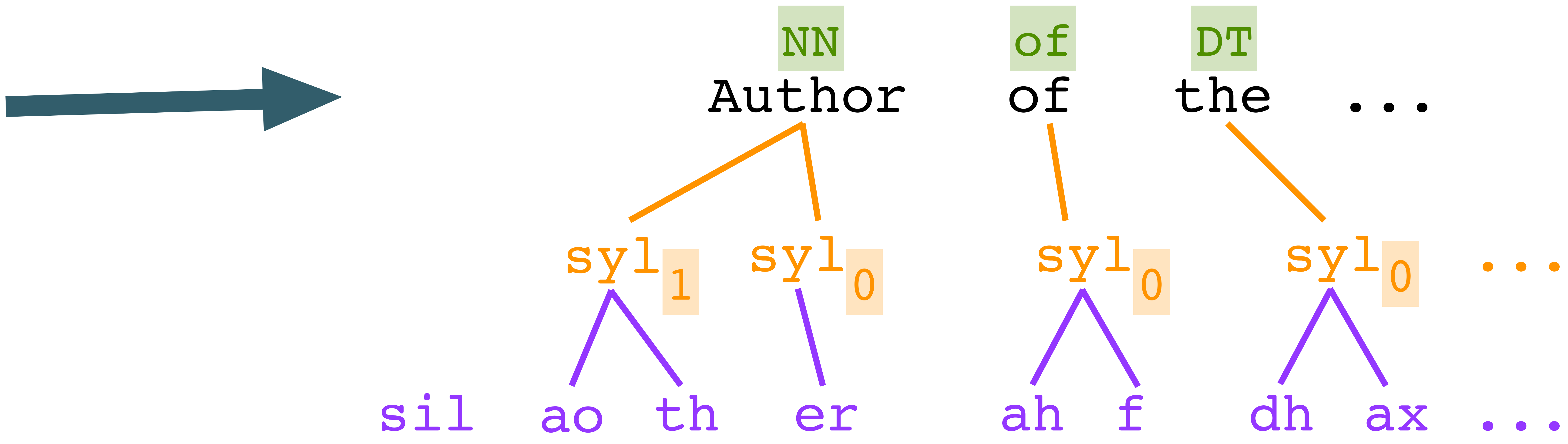
- Linguistic specification
- Linguistic features
- Acoustic features

Terminology

- Linguistic specification
  - this entire thing
- Linguistic features
  - individual elements
- Acoustic features
  - sequence of frames

Terminology

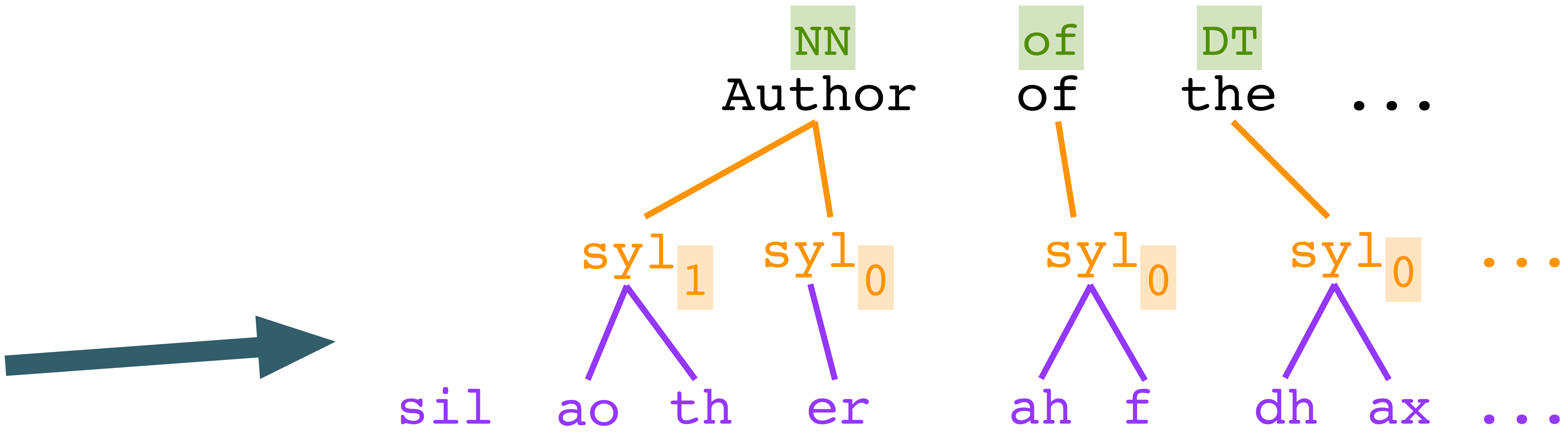
- Linguistic specification
  - this entire thing



- Linguistic features
  - individual elements
- Acoustic features
  - sequence of frames

Terminology

- Linguistic specification
  - this entire thing
- Linguistic features
  - individual elements
- Acoustic features
  - sequence of frames



NN of DT

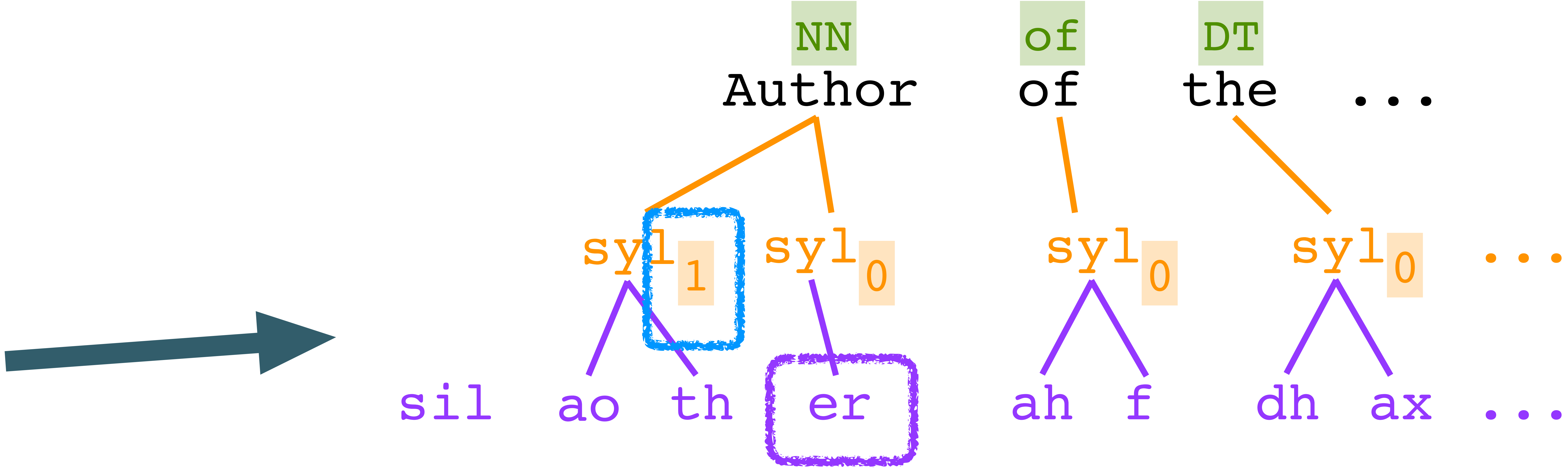
Author of the ...

syl_1 syl_0 syl_0 syl_0 ...

sil ao th er ah f dh ax ...

Terminology

- Linguistic specification
  - this entire thing
- Linguistic features
  - individual elements
- Acoustic features
  - sequence of frames



Terminology

- Linguistic specification
  - this entire thing

- Linguistic features
  - individual elements

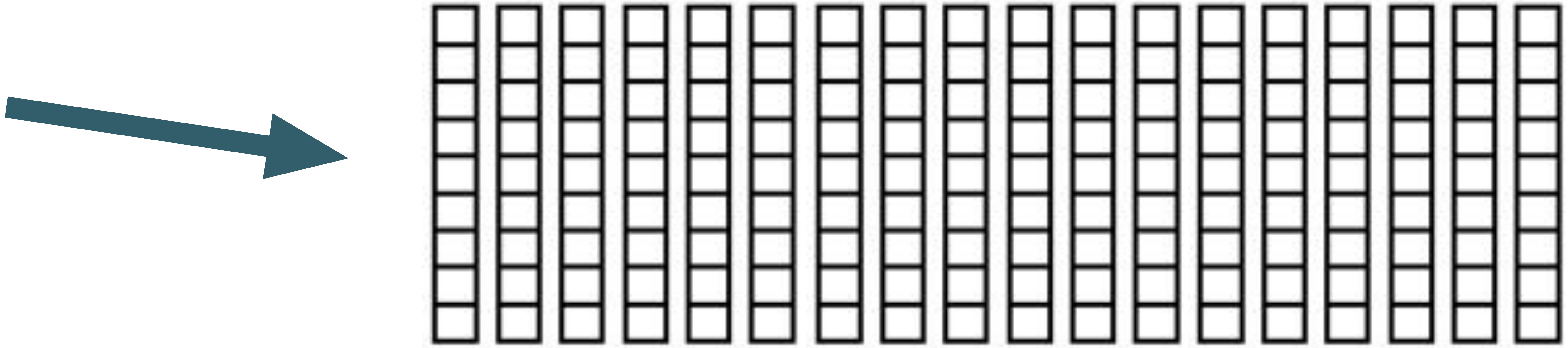
- Acoustic features
  - sequence of frames

Terminology

- Linguistic specification
  - this entire thing

- Linguistic features
  - individual elements

- Acoustic features
  - sequence of frames

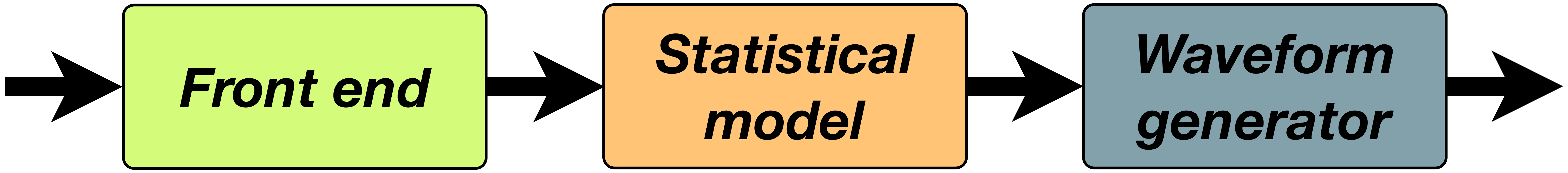


From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing

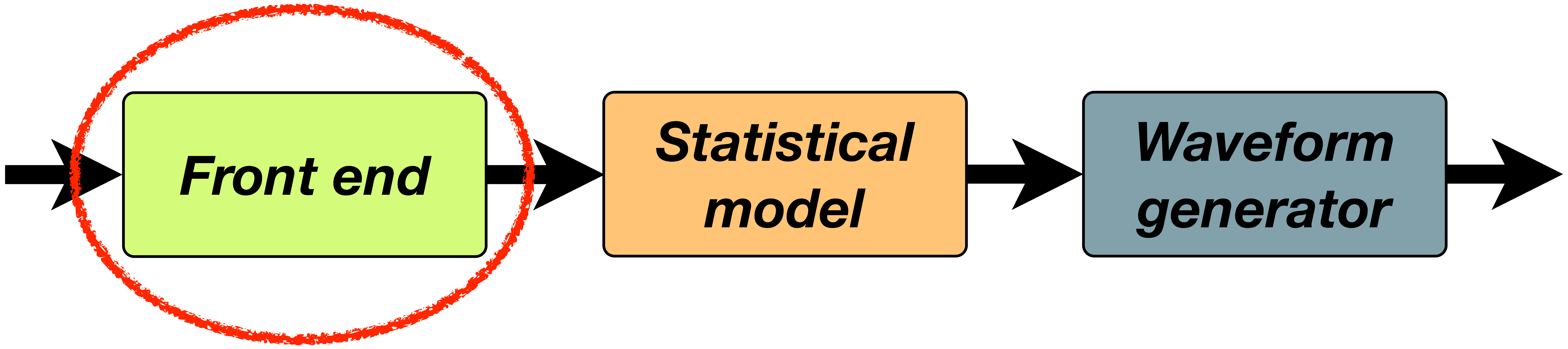
From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing

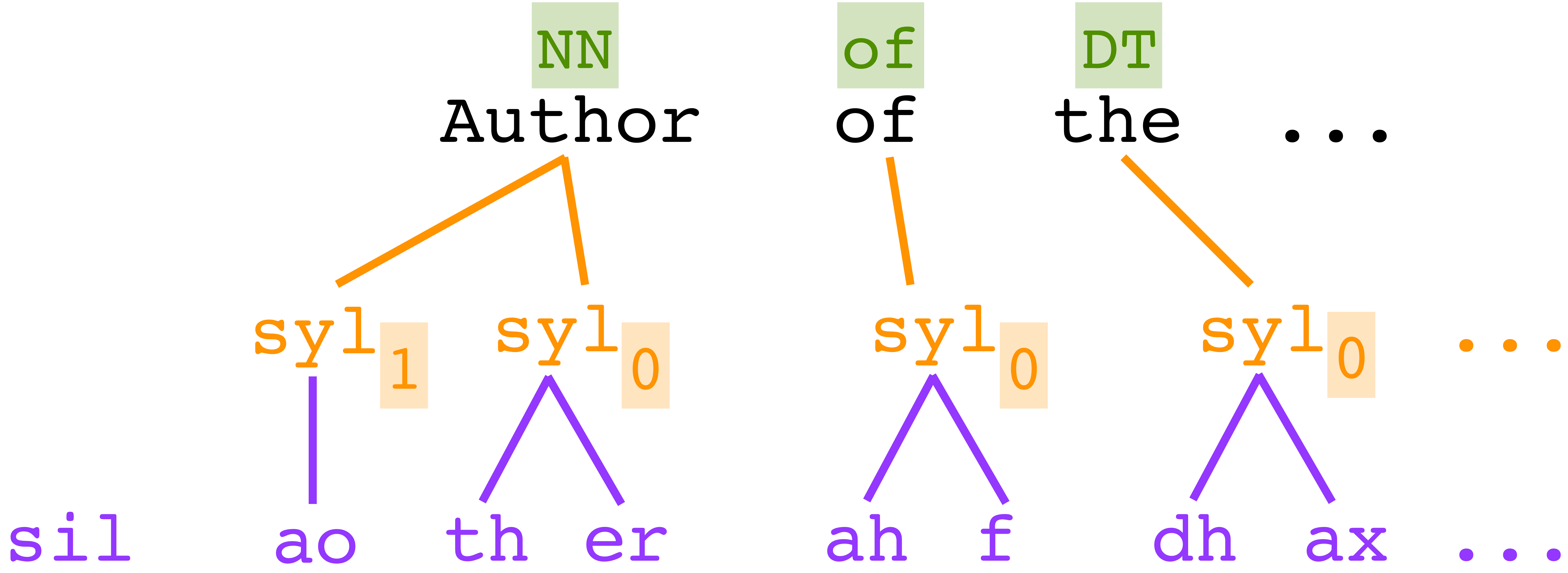


From text to speech

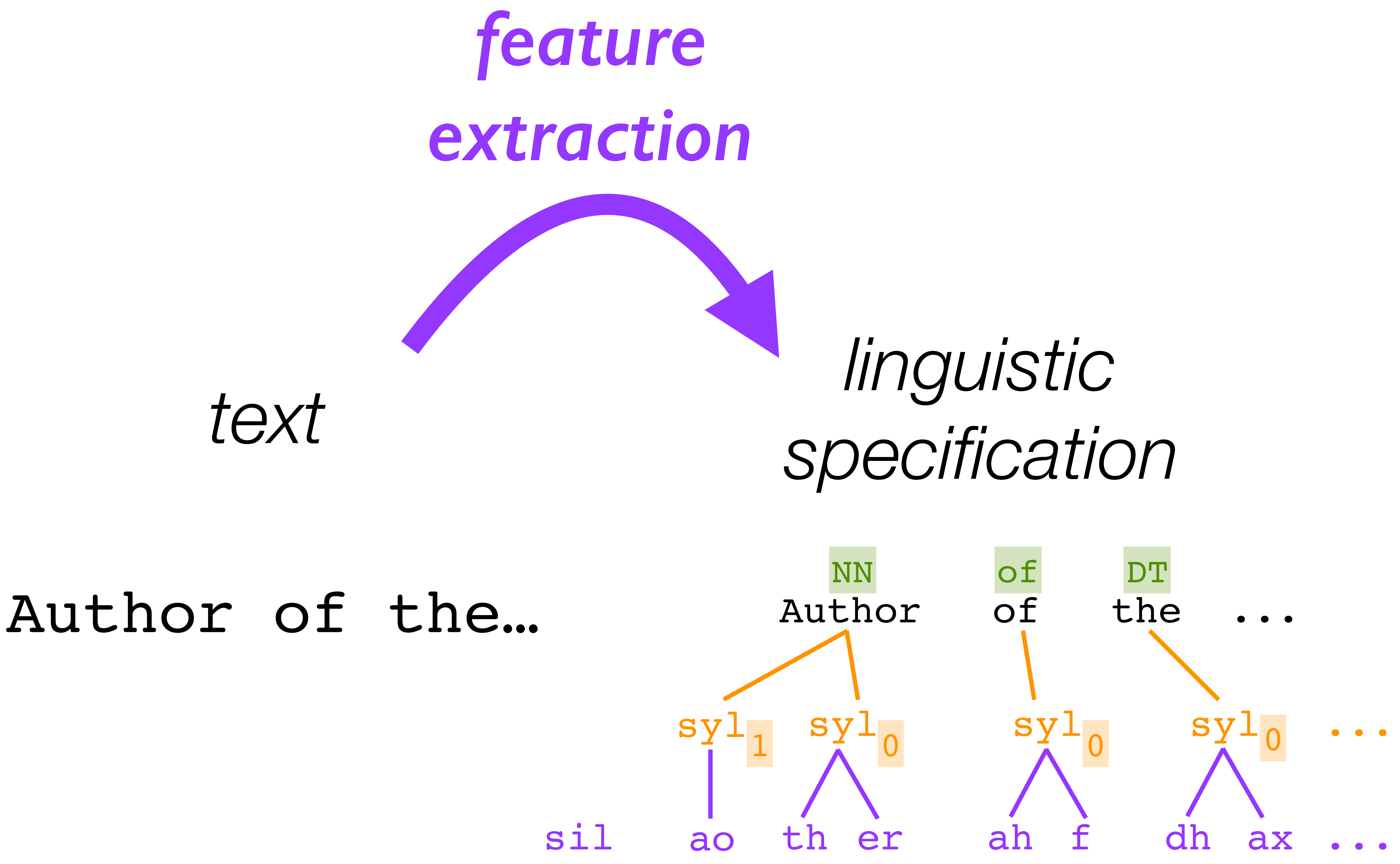
- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing



The linguistic specification



Extracting features from text using the front end



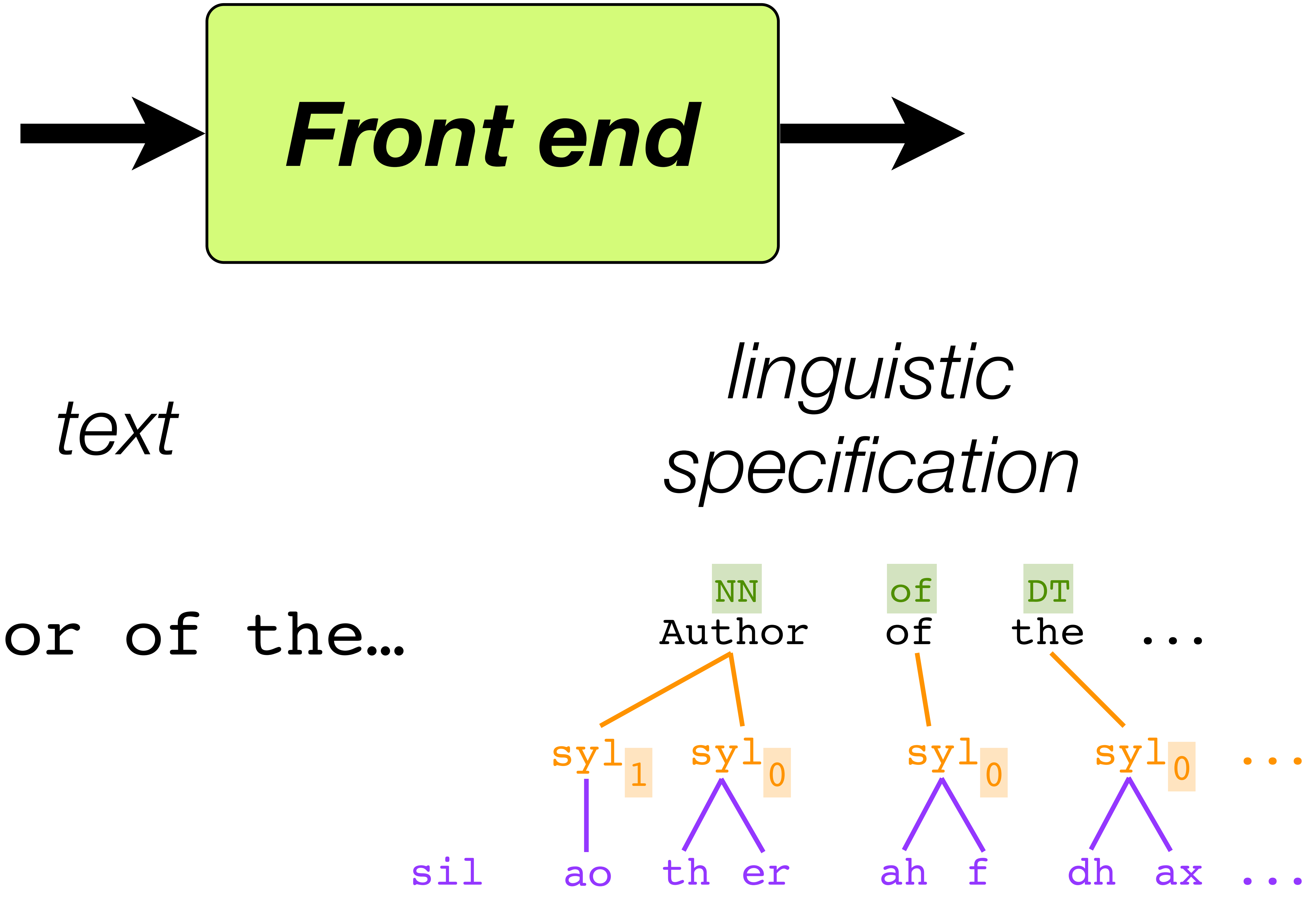
feature extraction

text

linguistic specification

Author of the…

Extracting features from text using the front end



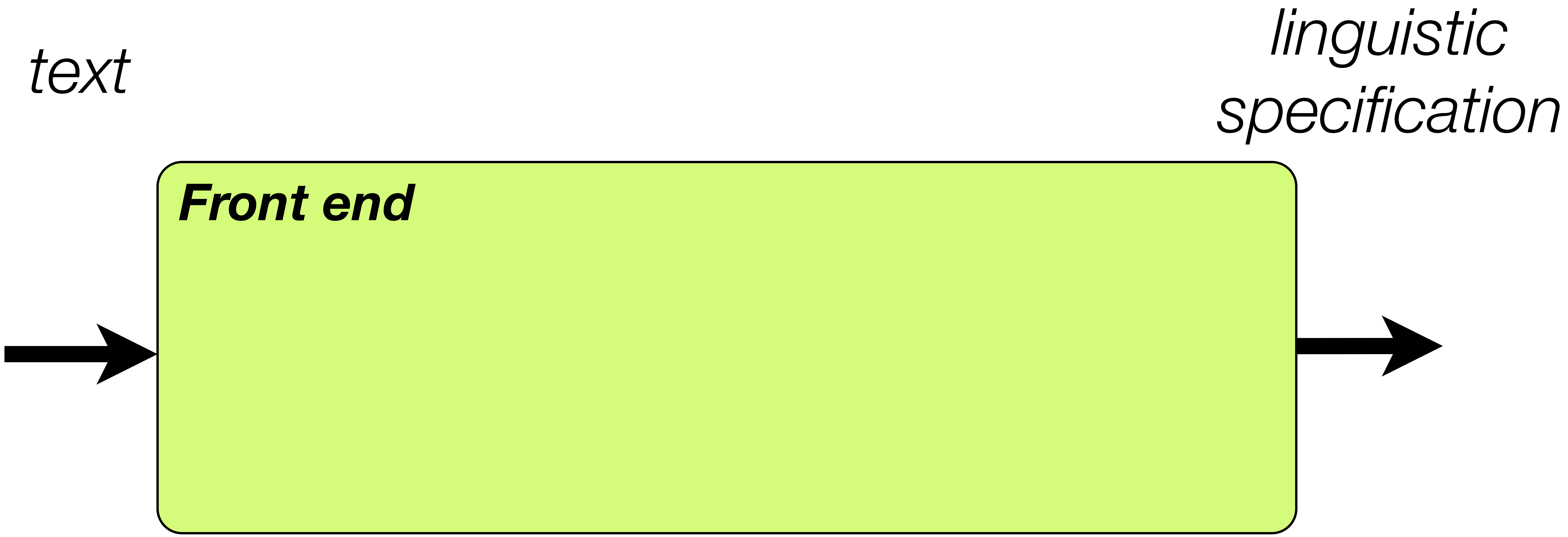
Front end

text

linguistic specification

`Author of the…`

Text processing pipeline

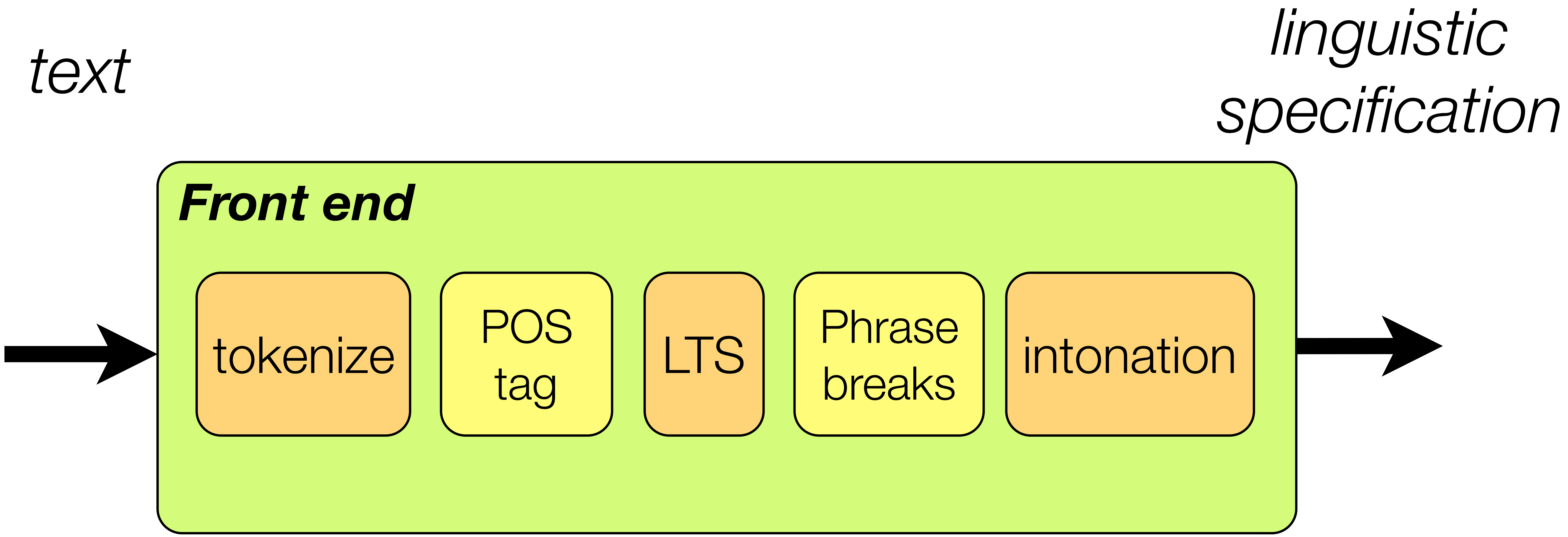


text

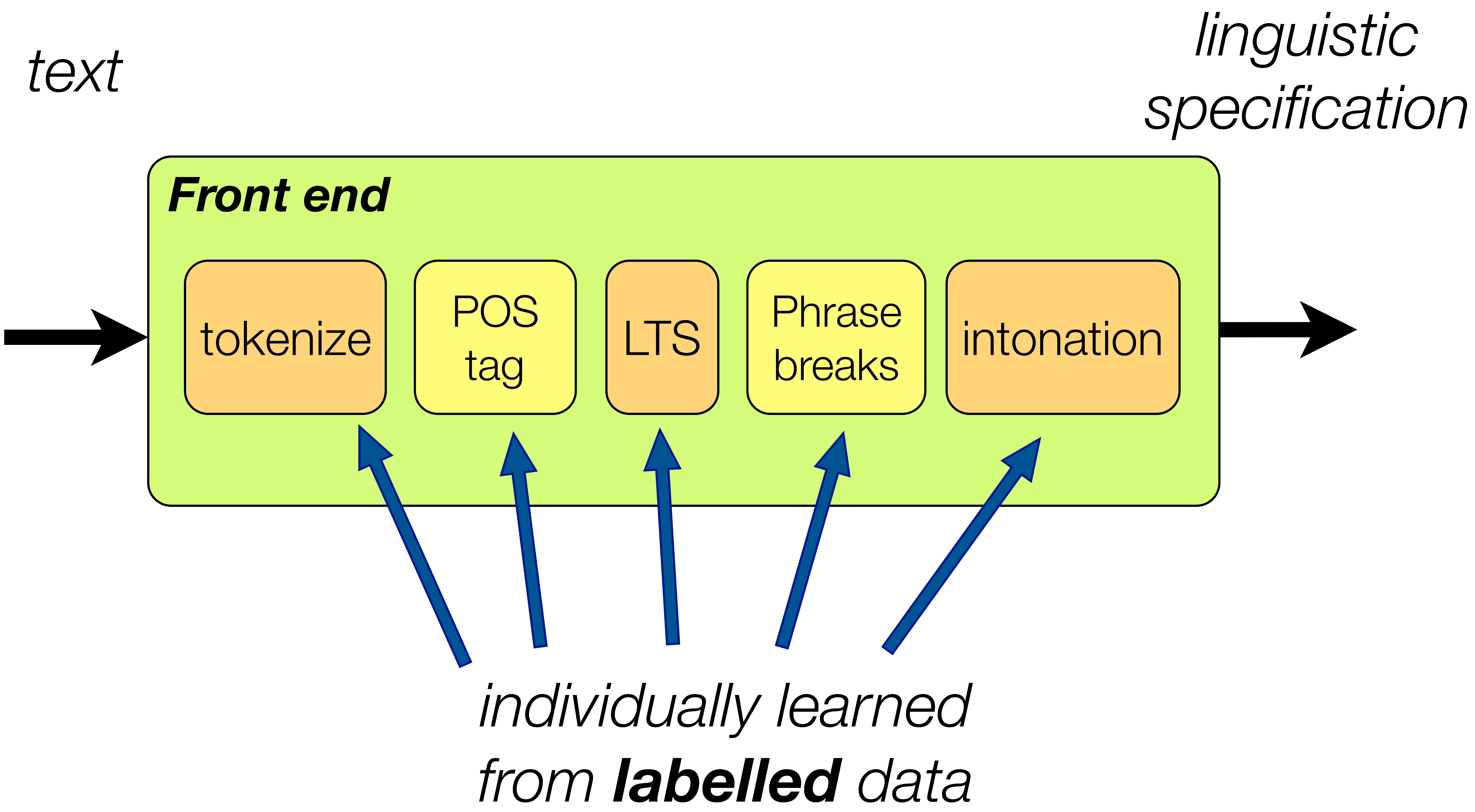
Front end

linguistic specification

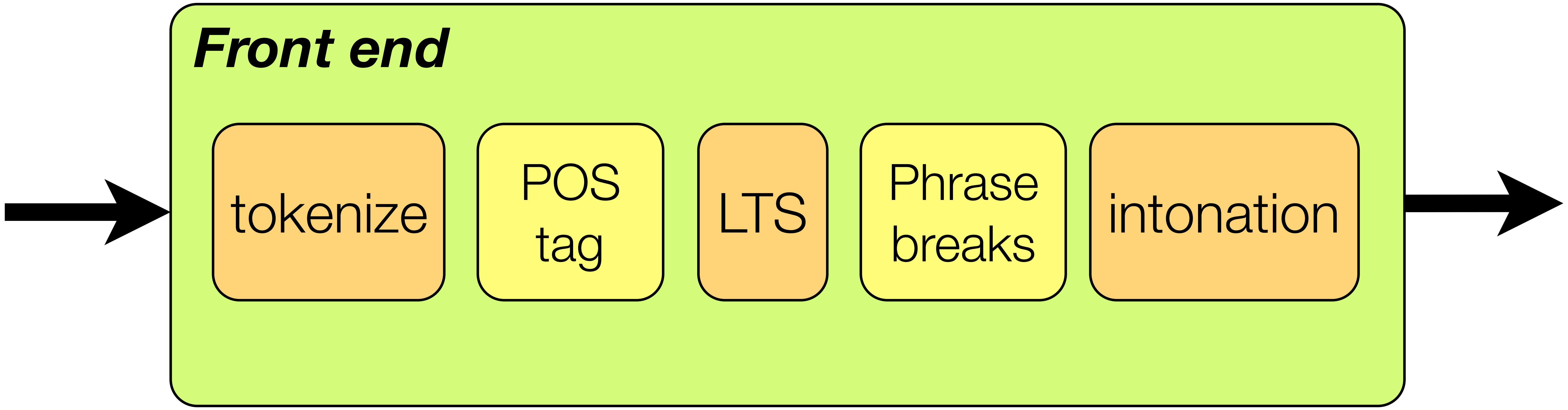
Text processing pipeline

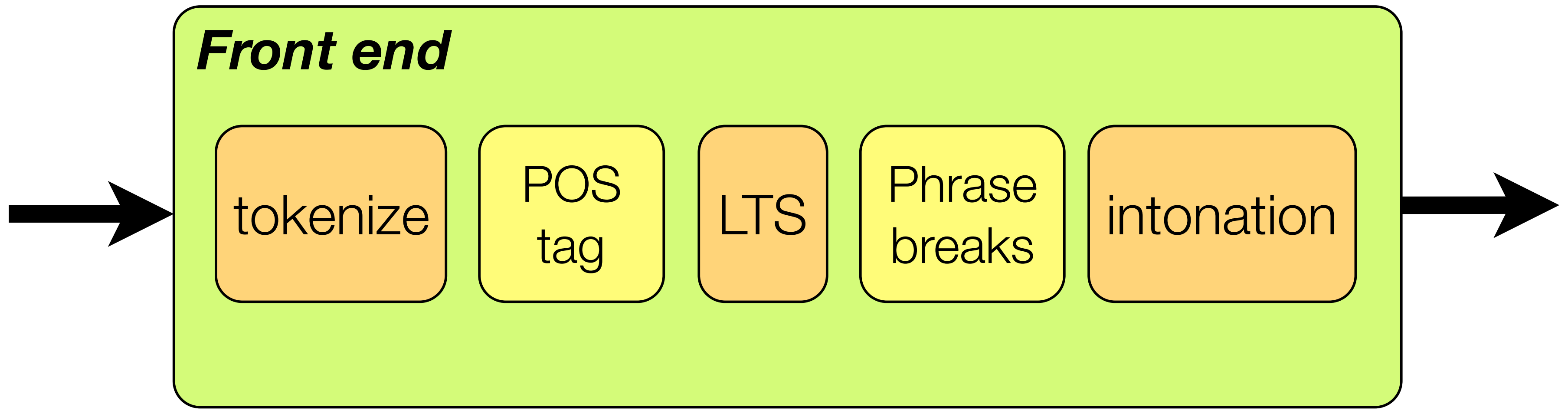


Text processing pipeline



Text processing pipeline



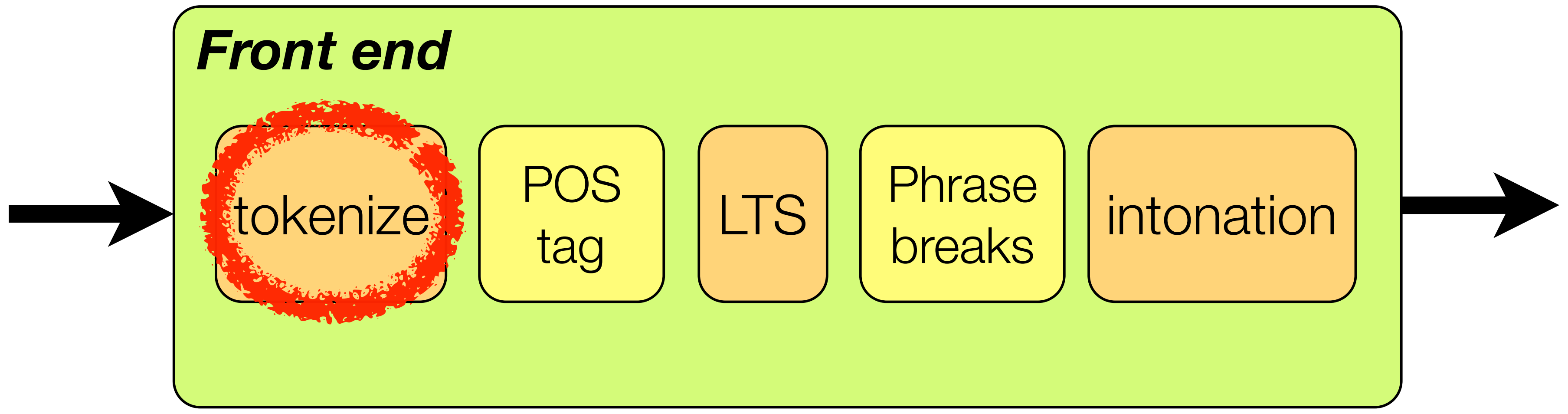


Tokenize & Normalize

- Step 1: divide input stream into tokens, which are potential words

- For English and many other languages
  - rule based
  - whitespace and punctuation are good features

- For some other languages, especially those that don’t use whitespace
  - may be more difficult
  - other techniques required (out of scope here)

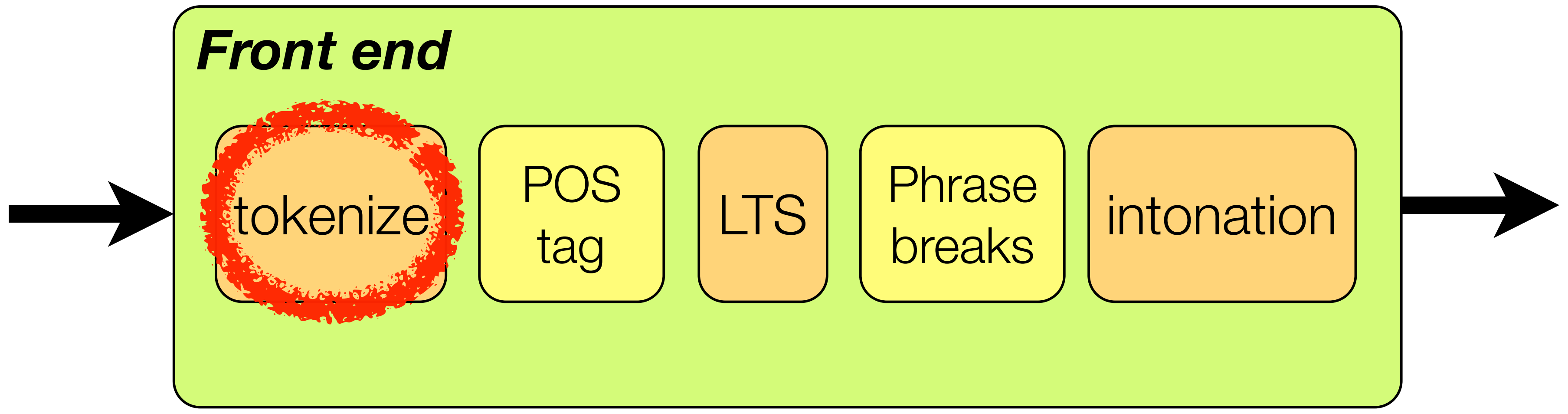


Tokenize & Normalize

- Step 1: divide input stream into tokens, which are potential words

- For English and many other languages
  - rule based
  - whitespace and punctuation are good features

- For some other languages, especially those that don't use whitespace
  - may be more difficult
  - other techniques required (out of scope here)

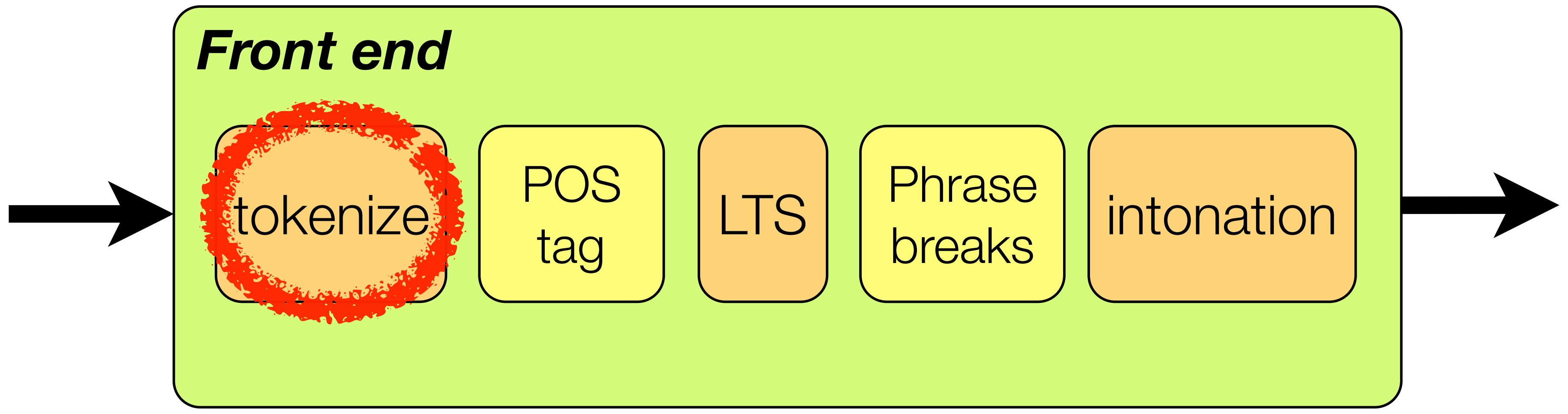


Tokenize & Normalize

- Step 2: classify every token, finding **Non-Standard Words** that need further processing

```
In 2011, I spent £100 at IKEA on 100 DVD holders.
```

Tokenize & Normalize



Front end

tokenize

POS tag

LTS

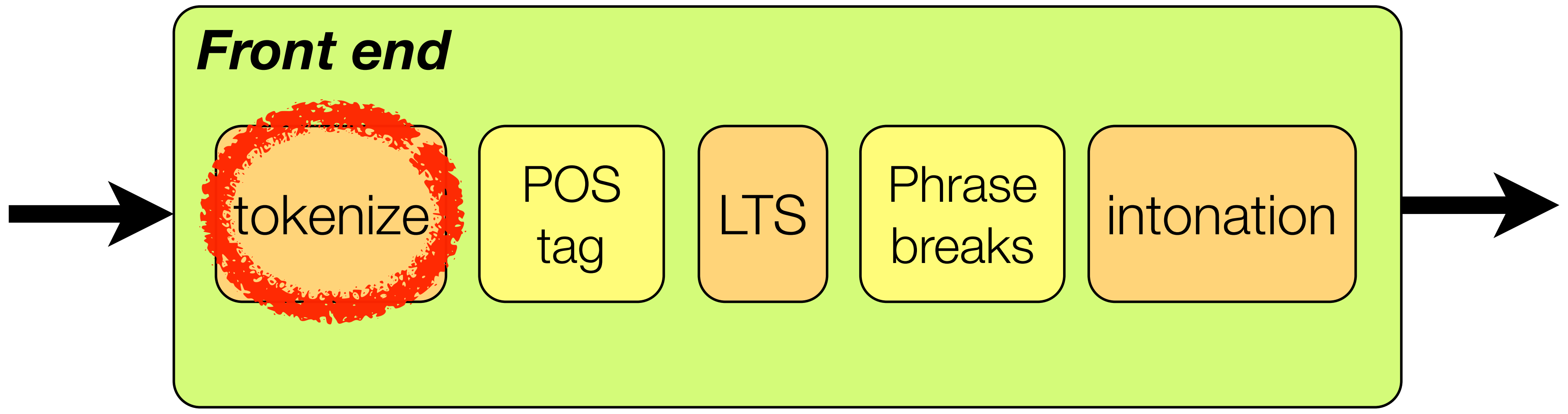
Phrase breaks

intonation

- Step 2: classify every token, finding **Non-Standard Words** that need further processing

```
In 2011, I spent £100 at IKEA on 100 DVD holders.

   NYER          MONEY   ASWD    NUM LSEQ
```

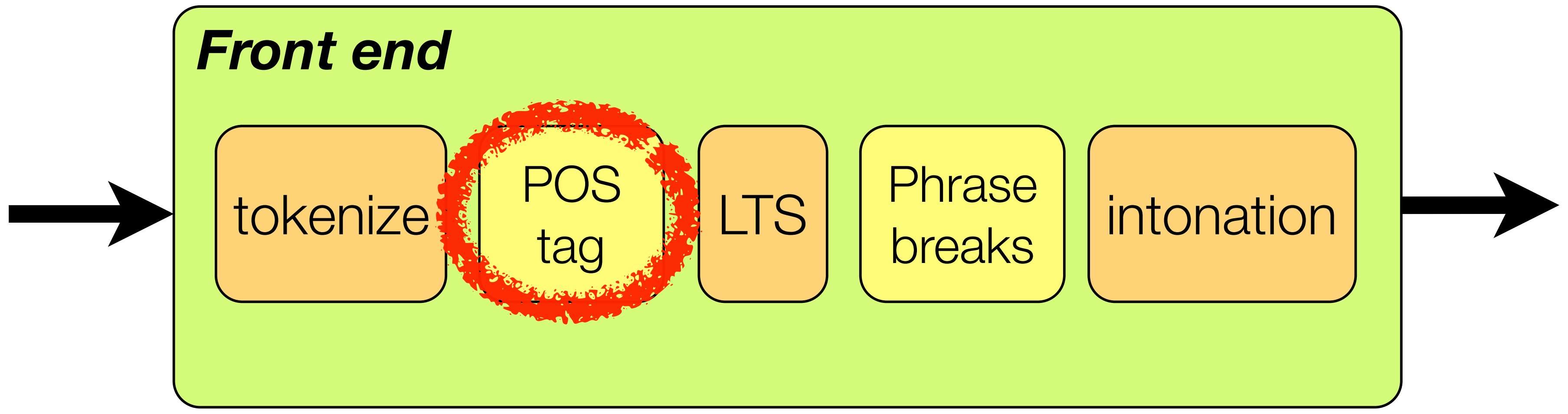
Tokenize & Normalize



- Step 3: a set of specialised modules to process NSWs of a each type

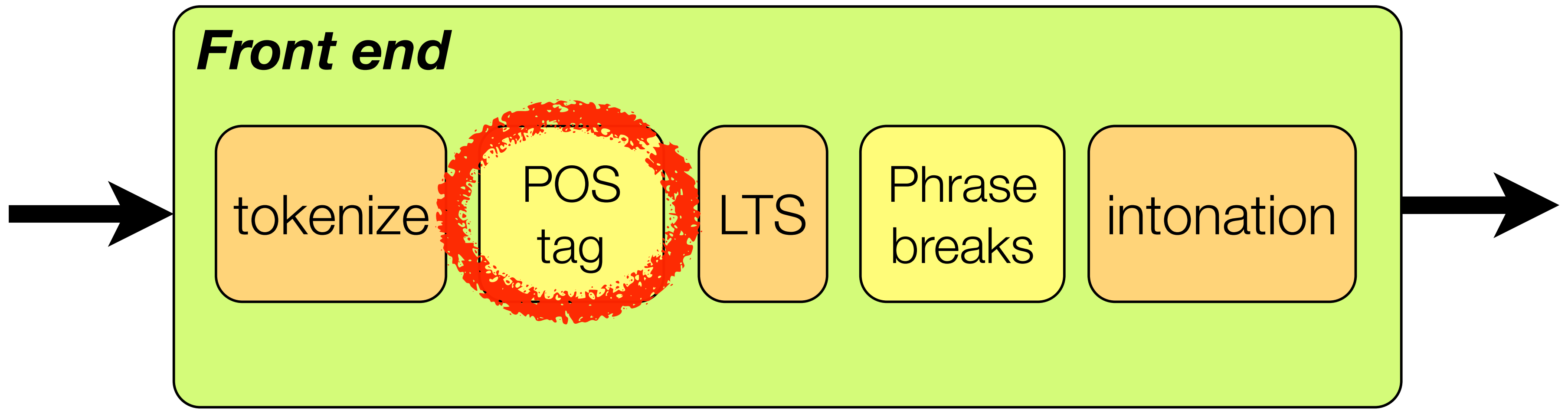
2011	⇨	NYER	⇨	twenty eleven
£100	⇨	MONEY	⇨	one hundred pounds
IKEA	⇨	ASWD	⇨	*apply letter-to-sound*
100	⇨	NUM	⇨	one hundred
DVD	⇨	LSEQ	⇨	D. V. D. ⇨ dee vee dee

POS tagging



- Part-of-speech tagger
- Accuracy can be very high
- Trained on **annotated** text data
- **Categories** are designed for text, not speech

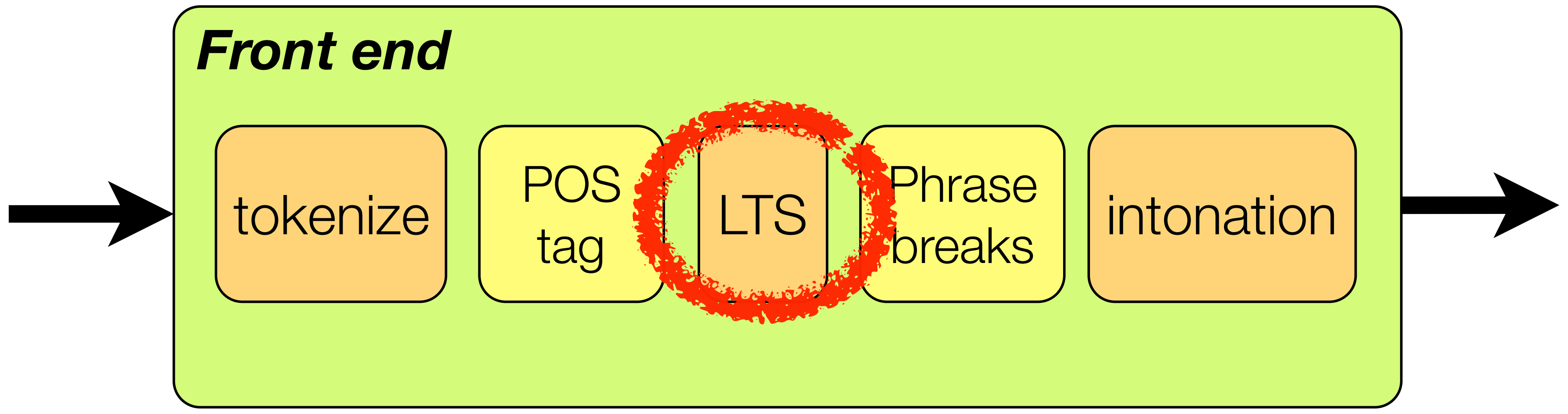
POS tagging



- Part-of-speech tagger
- Accuracy can be very high
- Trained on **annotated** text data
- **Categories** are designed for text, not speech

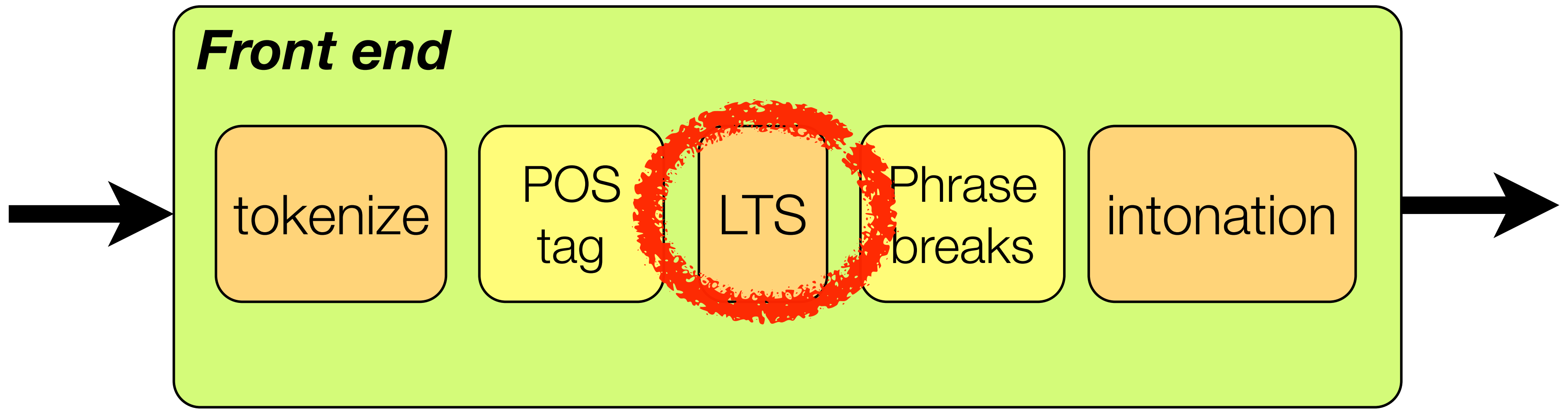
```
NP  Ed
NP  Beard,
VBZ says
DT  the
NN  push
IN  for
VBP do
PP  it
PP  yourself
NN  lawmaking
VBZ comes
IN  from
NNS voters
WP  who
VBP feel
VBN frustrated
IN  by
PP$ their
JJ  elected
NNS officials.
CC  But
DT  the
NN  initiative
```

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

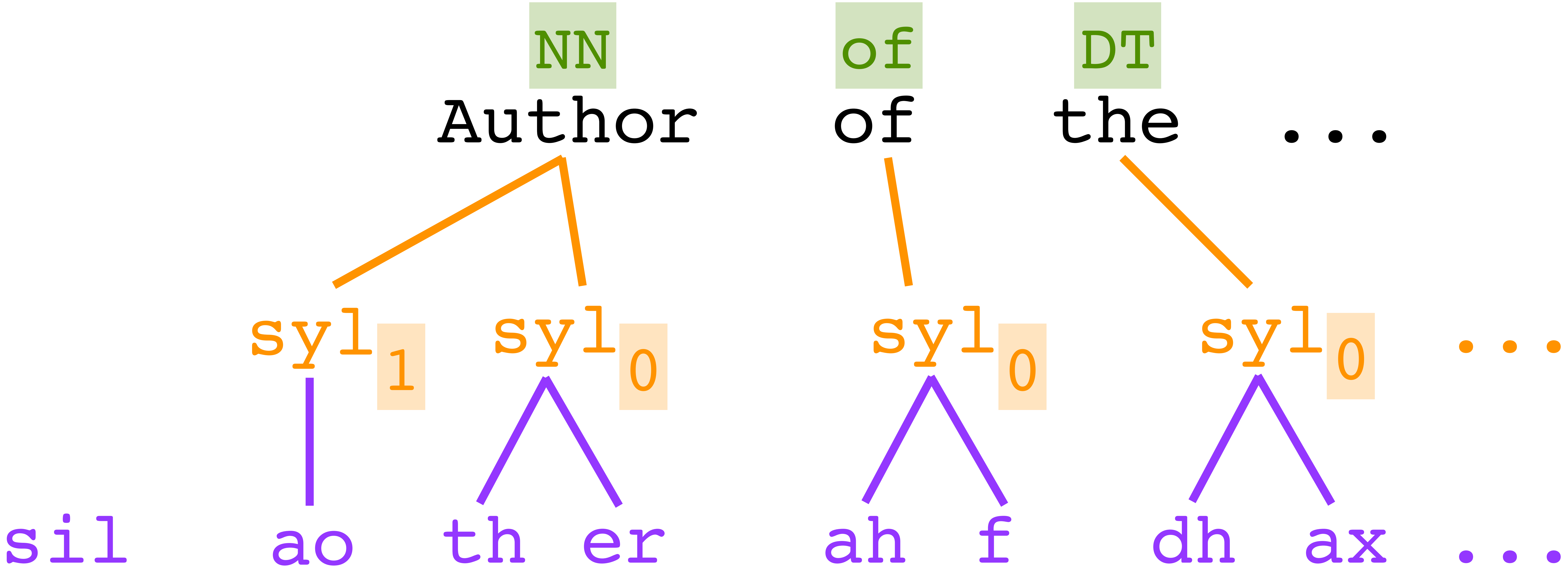
Pronunciation / LTS



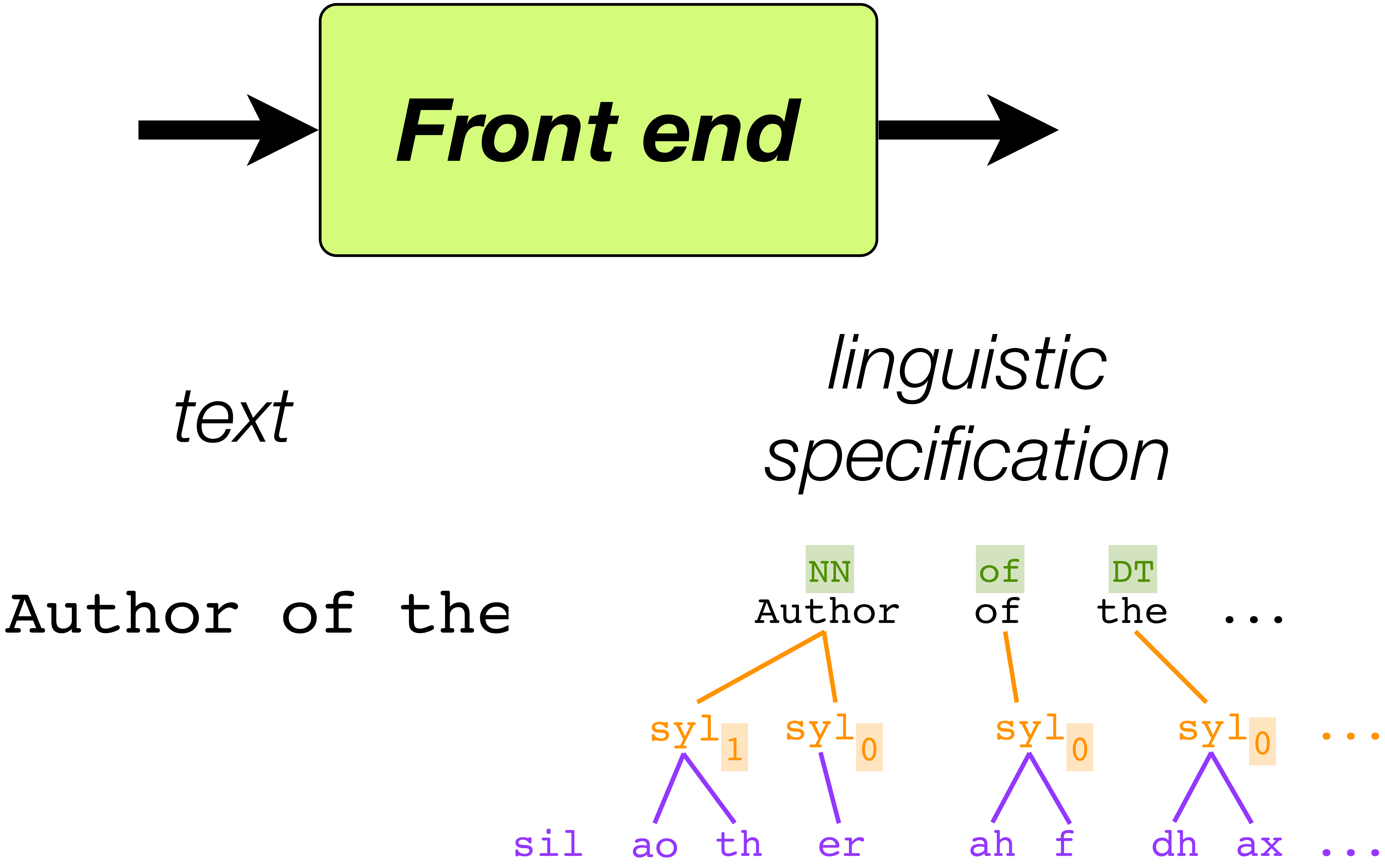
- Pronunciation model
  - dictionary look-up, *plus*
  - letter-to-sound model
- <u>But</u>
  - need deep **knowledge** of the language to design the phoneme set
  - human **expert** must write dictionary

```
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
AEROBATIC  EH2 R AH0 B AE1 T IH0 K
AEROBATICS  EH2 R AH0 B AE1 T IH0 K S
AEROBIC  EH0 R OW1 B IH0 K
AEROBICALLY  EH0 R OW1 B IH0 K L IY0
AEROBICS  ER0 OW1 B IH0 K S
AERODROME  EH1 R AH0 D R OW2 M
AERODROMES  EH1 R AH0 D R OW2 M Z
AERODYNAMIC  EH2 R OW0 D AY0 N AE1 M IH0 K
AERODYNAMICALLY  EH2 R OW0 D AY0 N AE1 M IH0 K L
AERODYNAMICIST  EH2 R OW0 D AY0 N AE1 M IH0 S IH
AERODYNAMICISTS  EH2 R OW0 D AY0 N AE1 M IH0 S I
AERODYNAMICISTS(1)  EH2 R OW0 D AY0 N AE1 M IH0
AERODYNAMICS  EH2 R OW0 D AY0 N AE1 M IH0 K S
AERODYNE  EH1 R AH0 D AY2 N
AERODYNE'S  EH1 R AH0 D AY2 N Z
AEROFLOT  EH1 R OW0 F L AA2 T
```

The linguistic specification



Linguistic feature engineering



Front end

text

linguistic specification

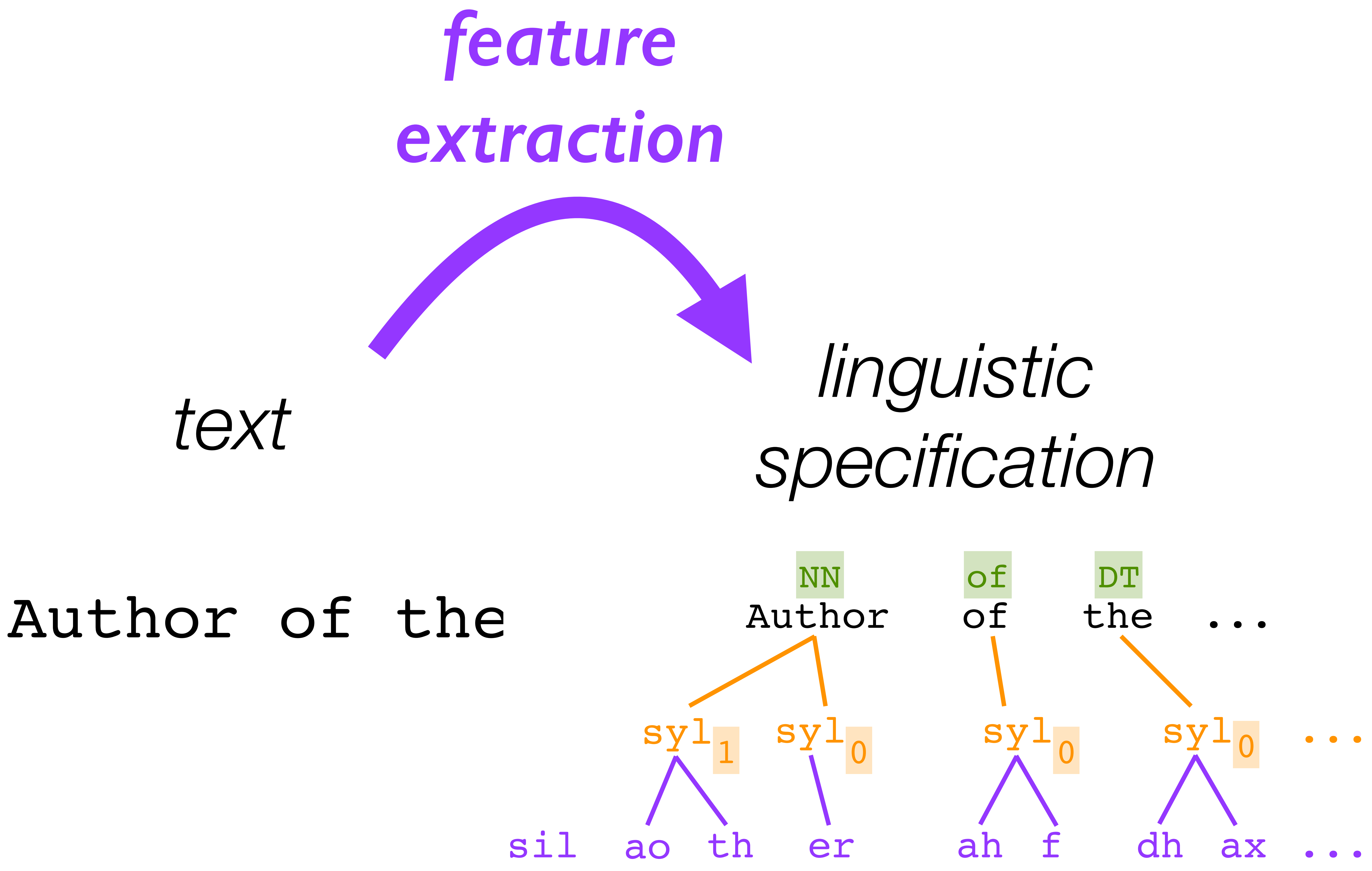
Author of the

NN Author   of of   DT the ...

syl_1 syl_0 syl_0 syl_0 ...

sil ao th er ah f dh ax ...

Linguistic feature engineering



feature extraction

text

linguistic specification

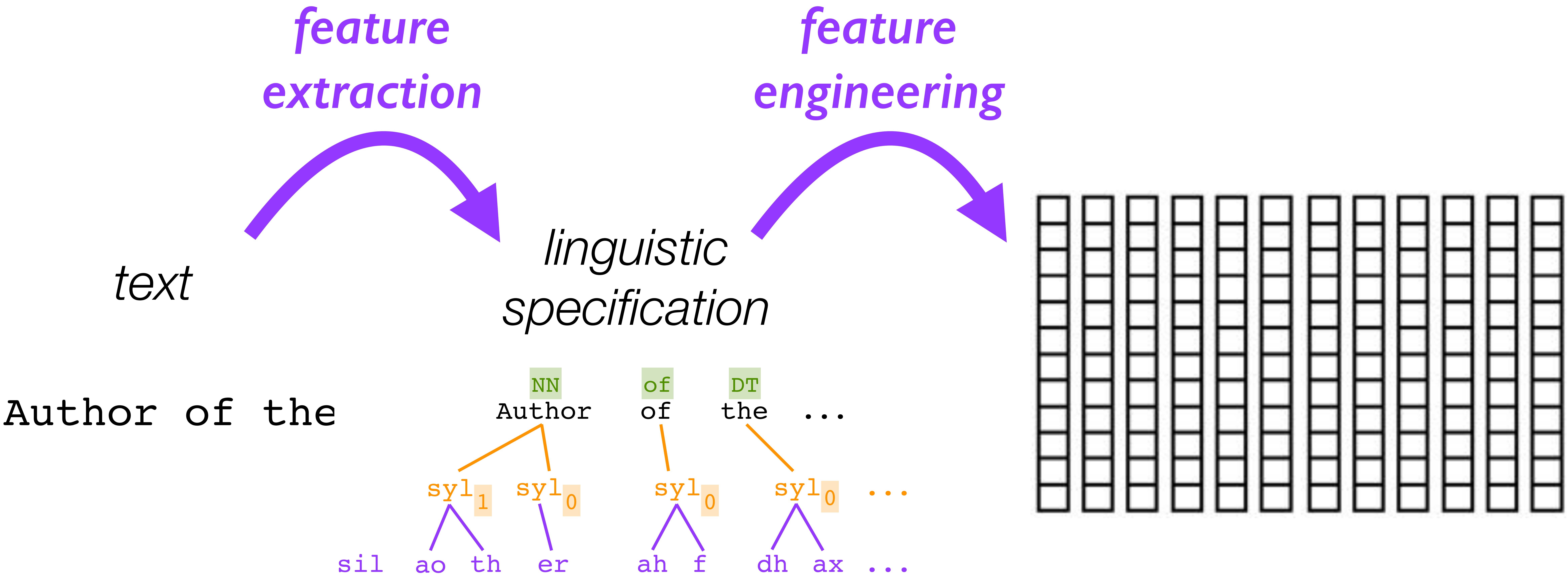
`Author of the`

```
NN      of   DT
Author  of   the  ...
```

syl$_1$ syl$_0$ syl$_0$ syl$_0$ ...

sil ao th er ah f dh ax ...

Linguistic feature engineering



feature extraction

feature engineering

text

linguistic specification

Author of the

Terminology

- Flatten

- Encode

- Upsample

Terminology

- Flatten

- Encode

- Upsample

Terminology

- Flatten

linguistic specification

⇩

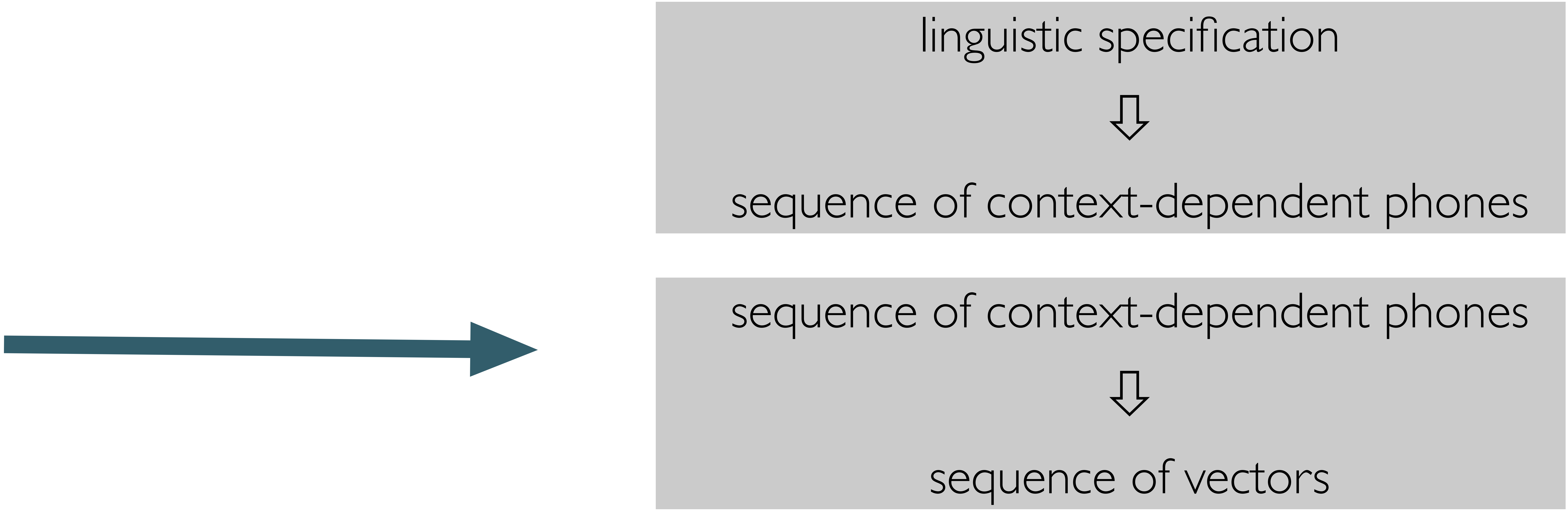
sequence of context-dependent phones

- Encode

- Upsample

Terminology

- Flatten
- Encode
- Upsample



linguistic specification

⇩

sequence of context-dependent phones

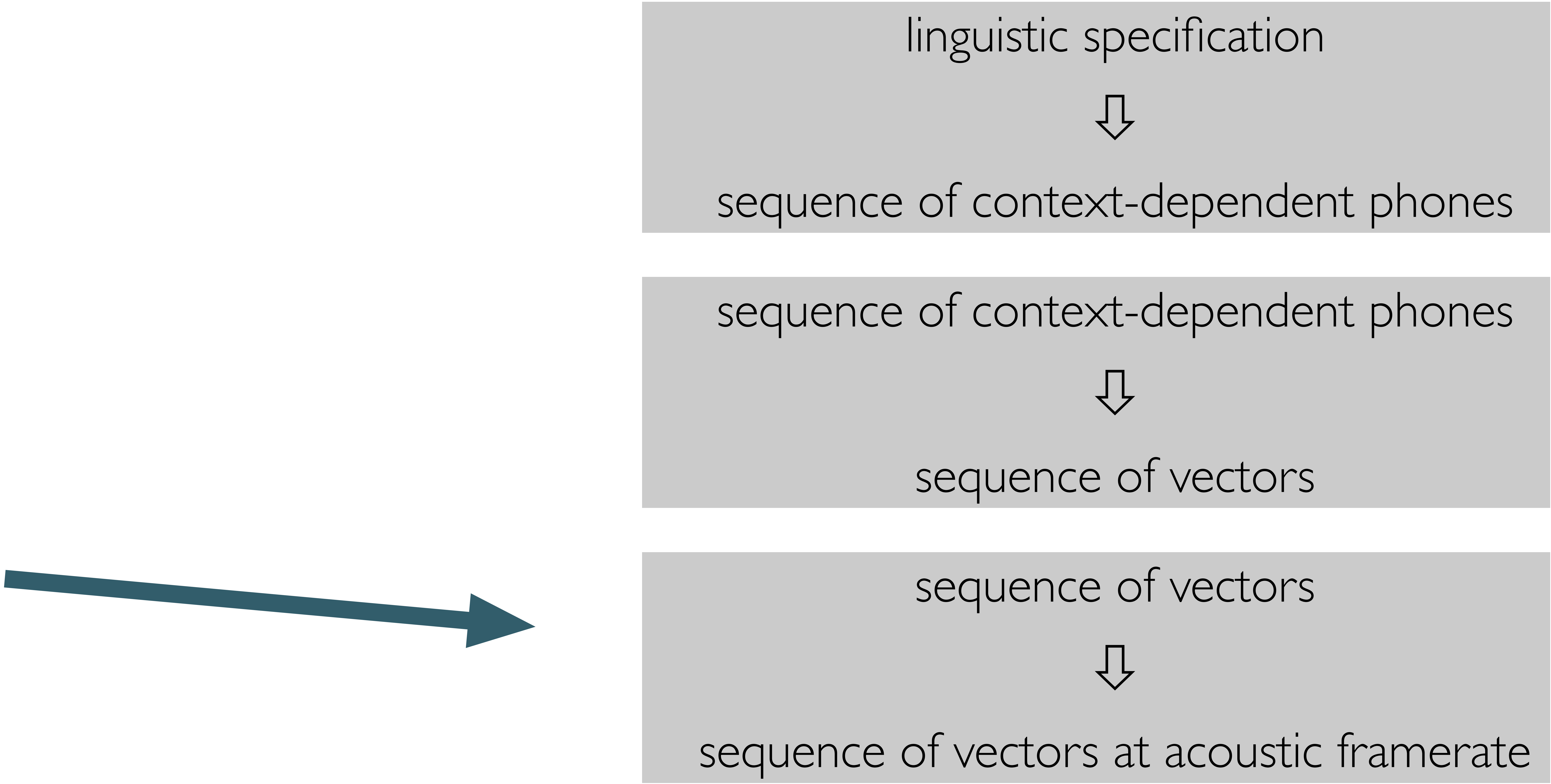
sequence of context-dependent phones

⇩

sequence of vectors

Terminology

- Flatten
- Encode
- Upsample



linguistic specification

⇩

sequence of context-dependent phones

sequence of context-dependent phones

⇩

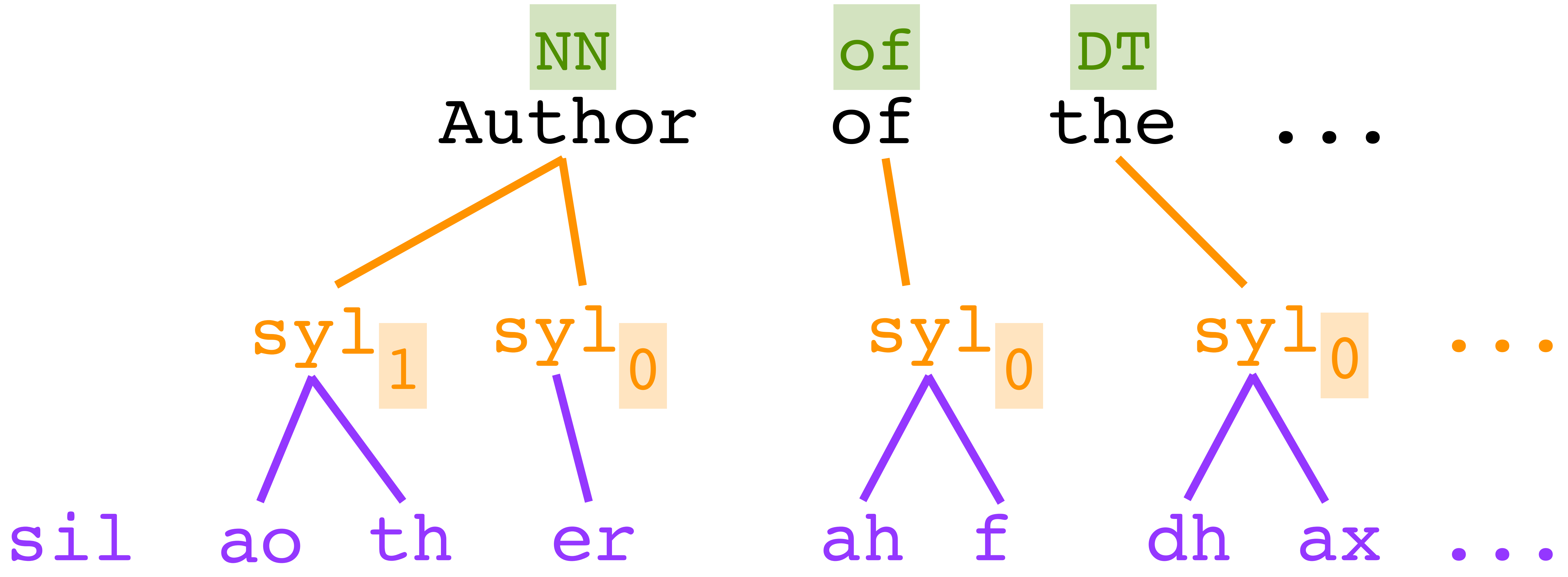
sequence of vectors

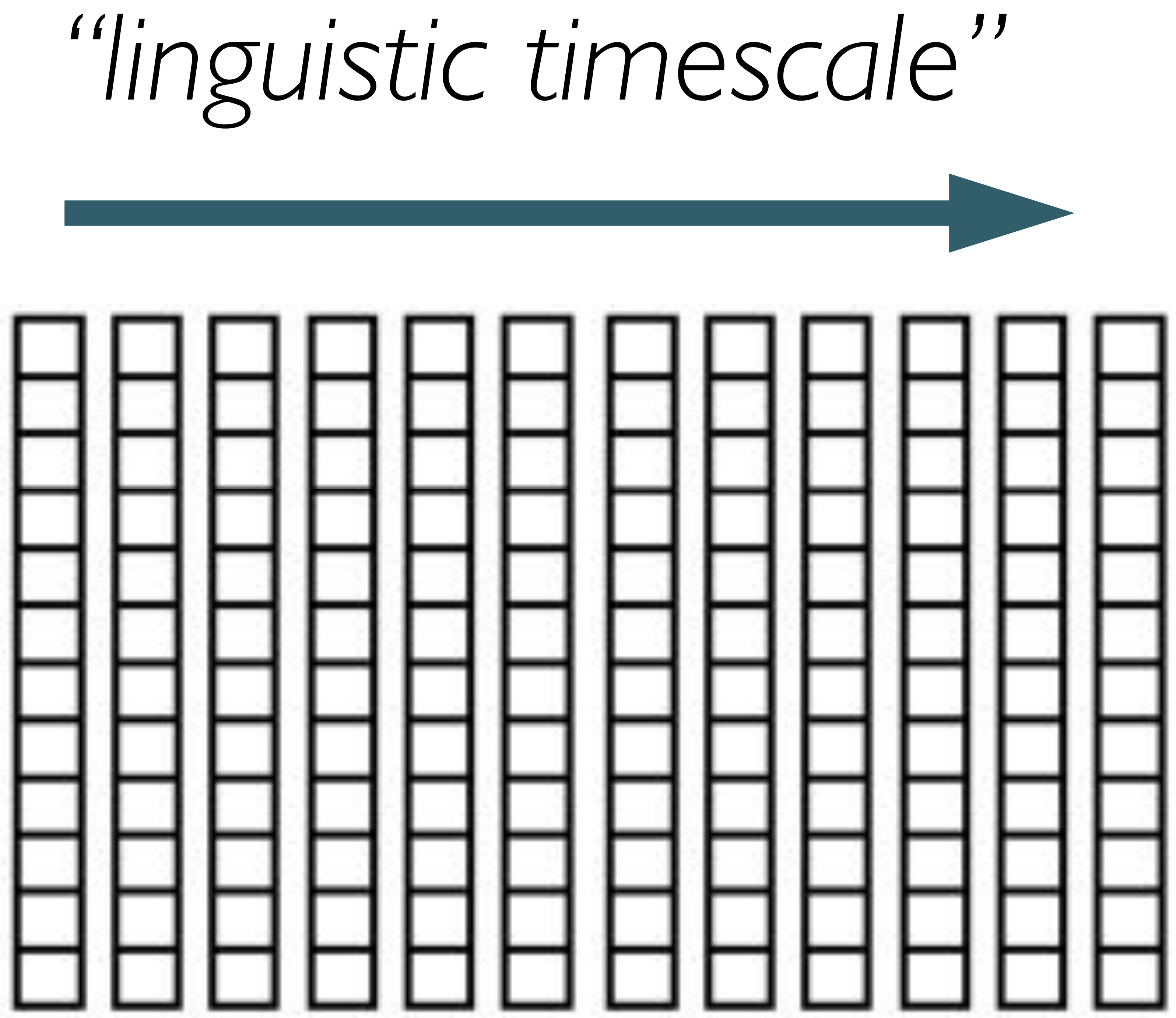
sequence of vectors

⇩

sequence of vectors at acoustic framerate

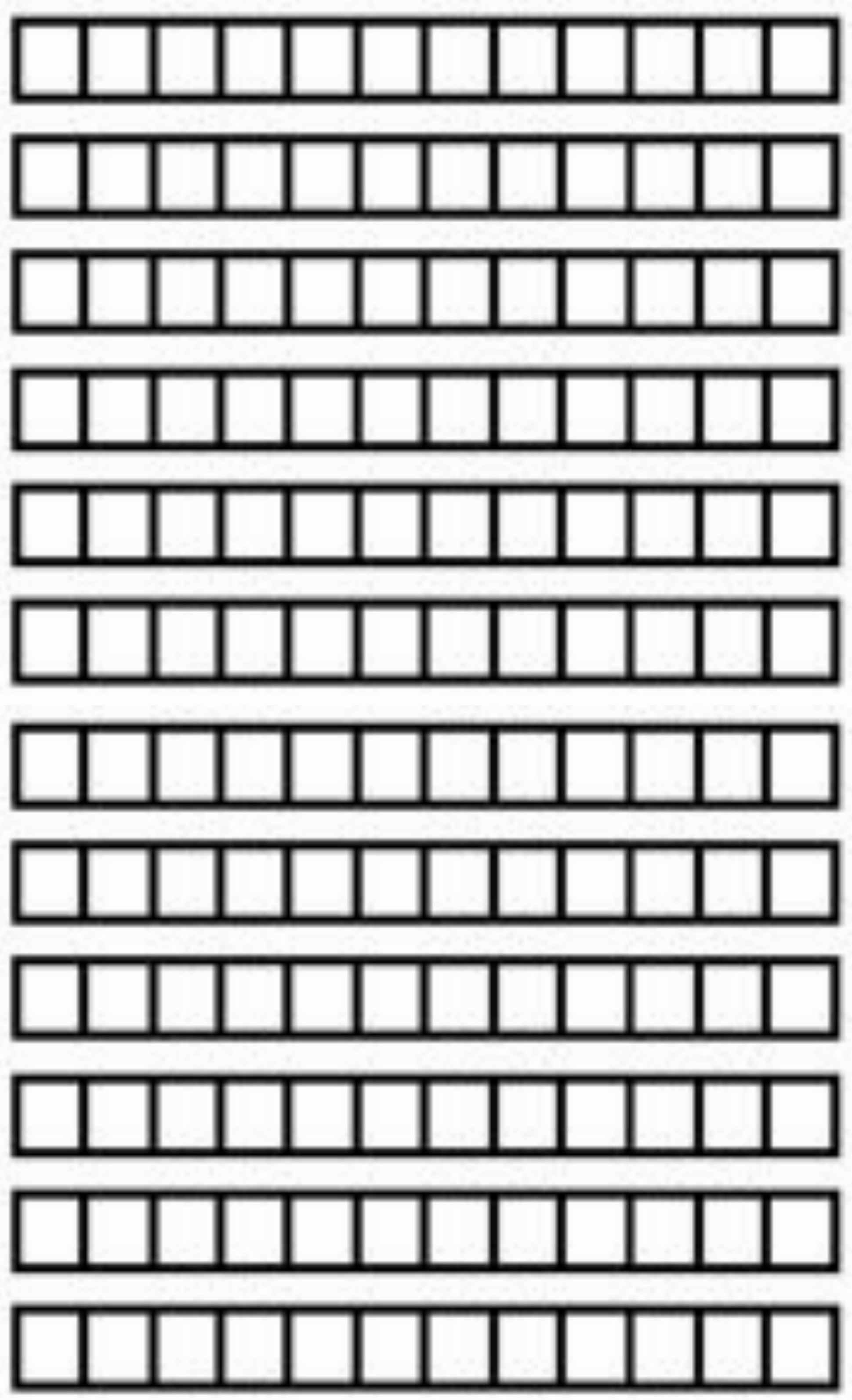
Flatten & encode: convert linguistic specification to vector sequence





Upsample: add duration information

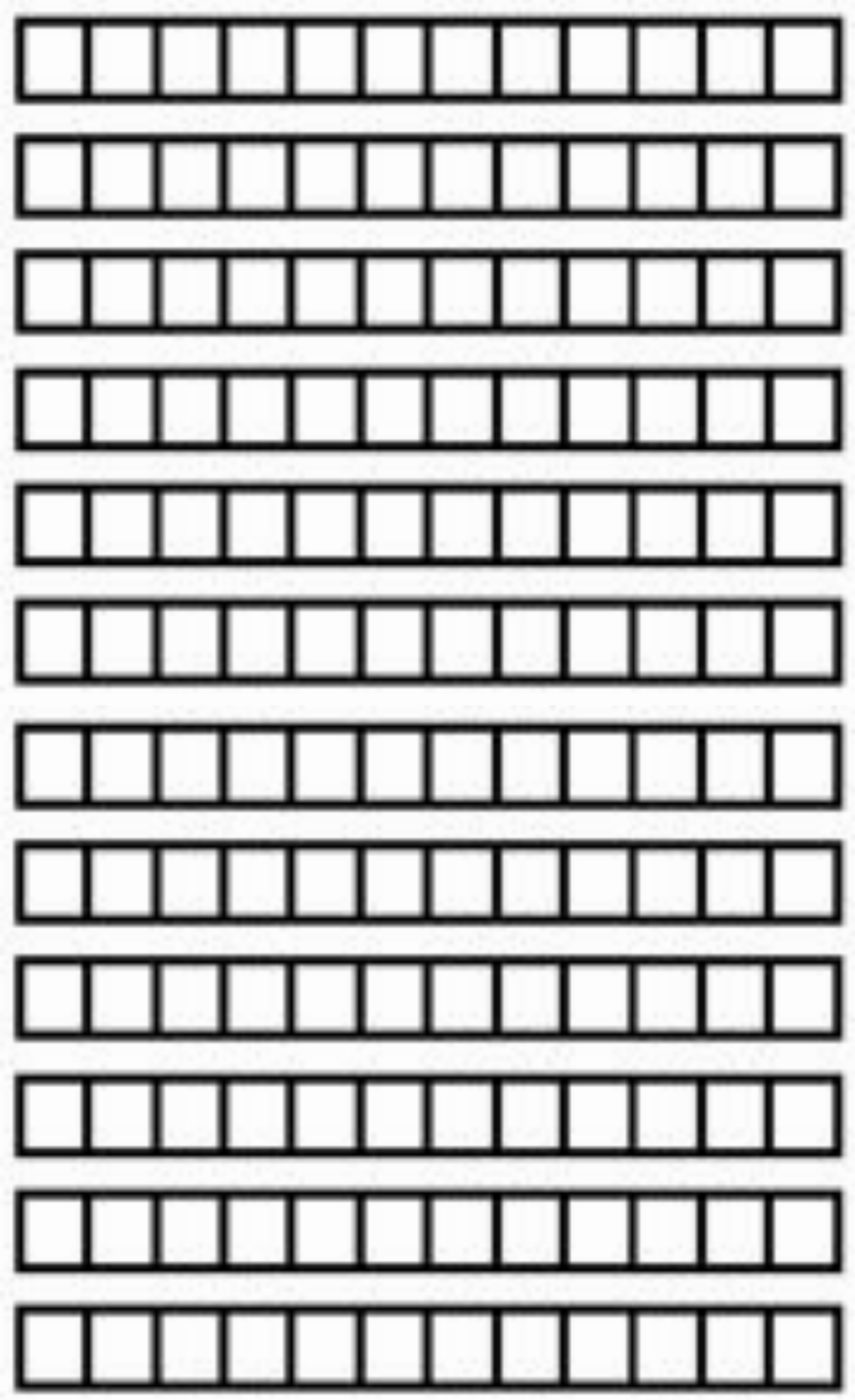
linguistic timescale



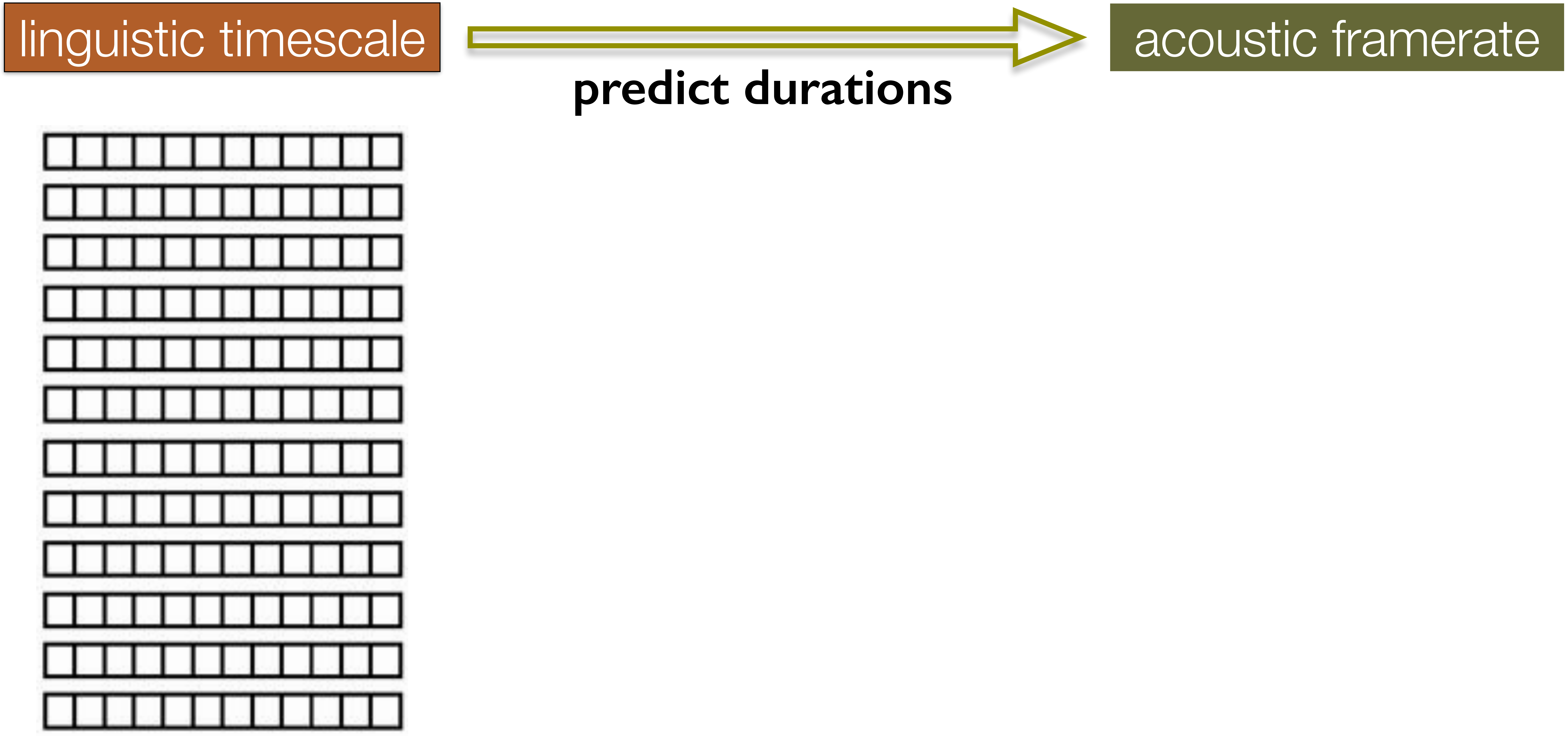
Upsample: add duration information

linguistic timescale

acoustic framerate



Upsample: add duration information



linguistic timescale

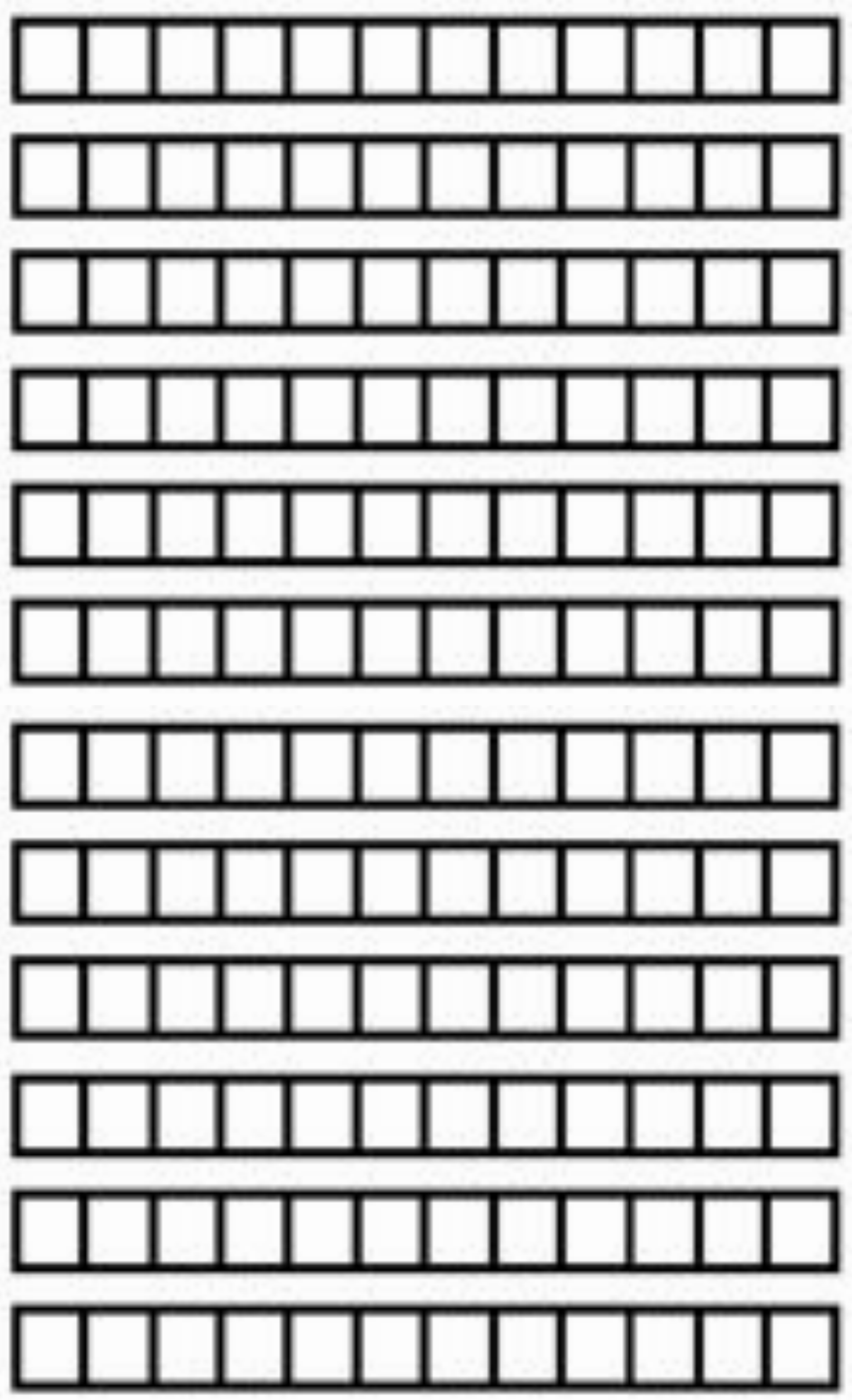
predict durations

acoustic framerate

Upsample: add duration information

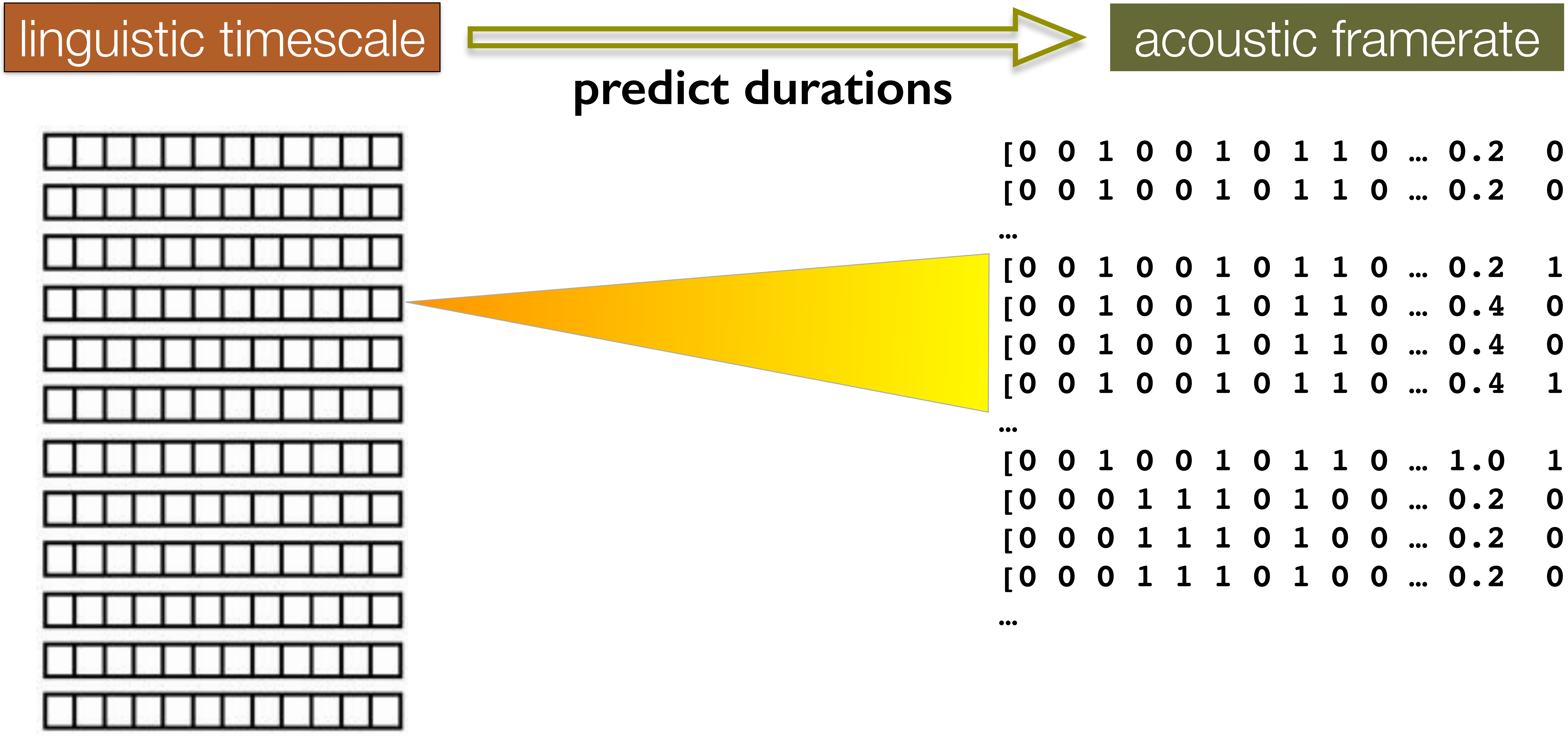
linguistic timescale → acoustic framerate

predict durations



```
[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]
…
```

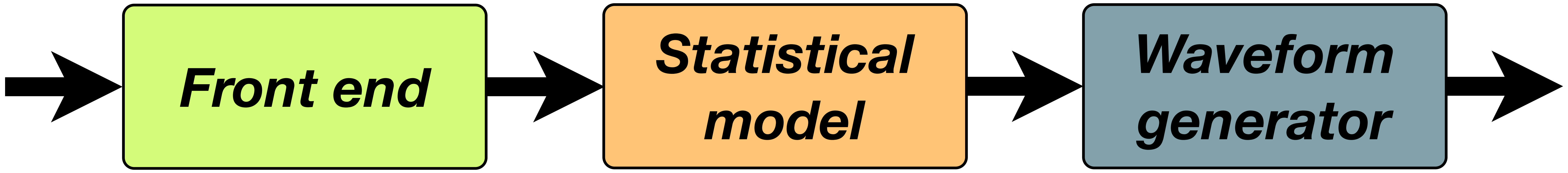
Upsample: add duration information



```
[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]
…
```

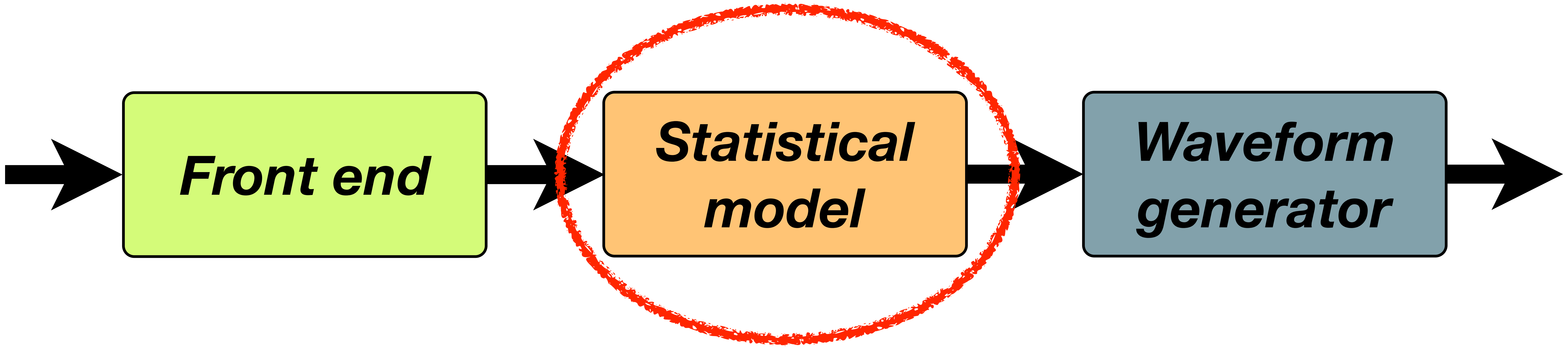
From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing

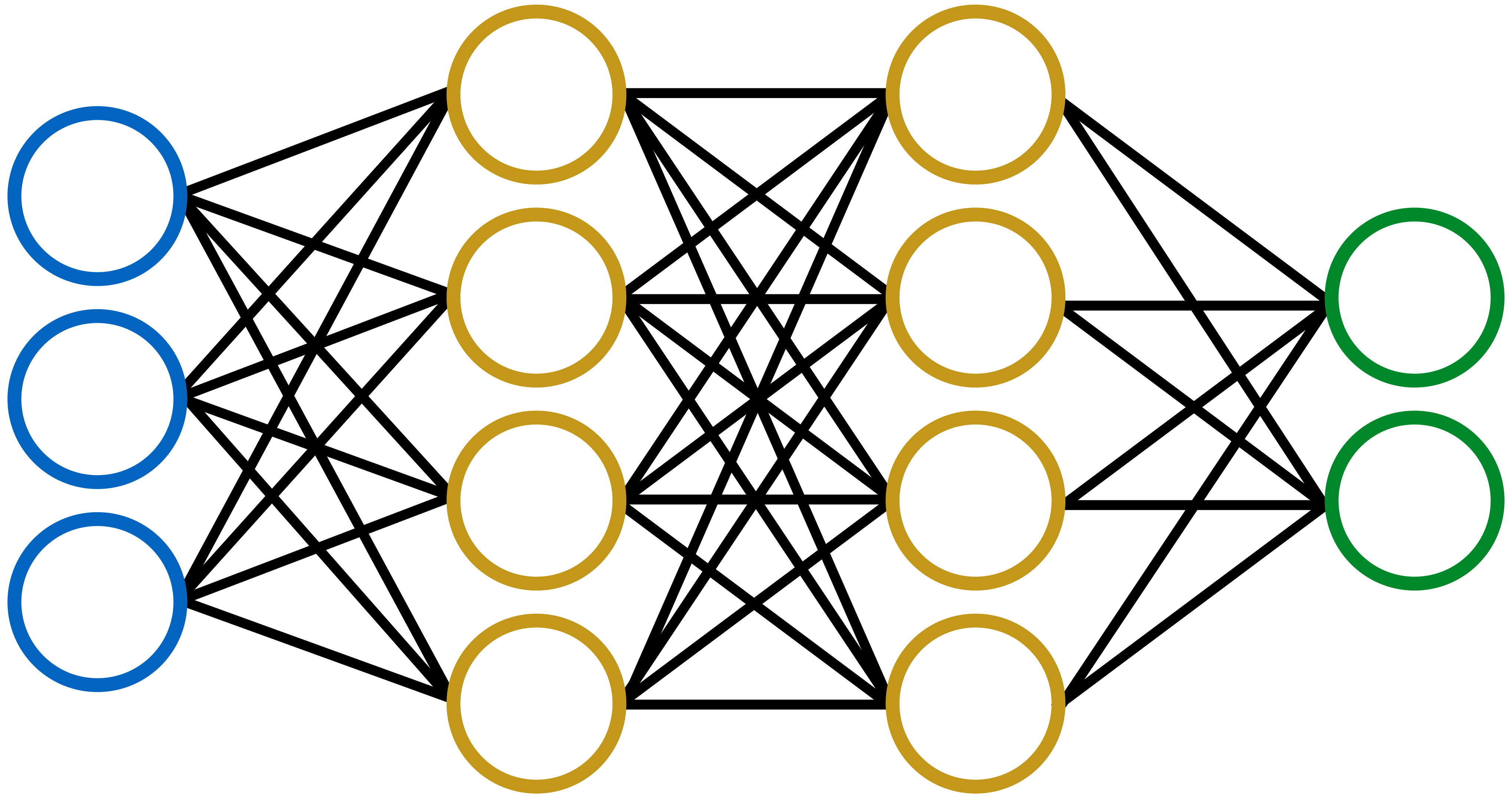


From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- <u>Regression</u>
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing

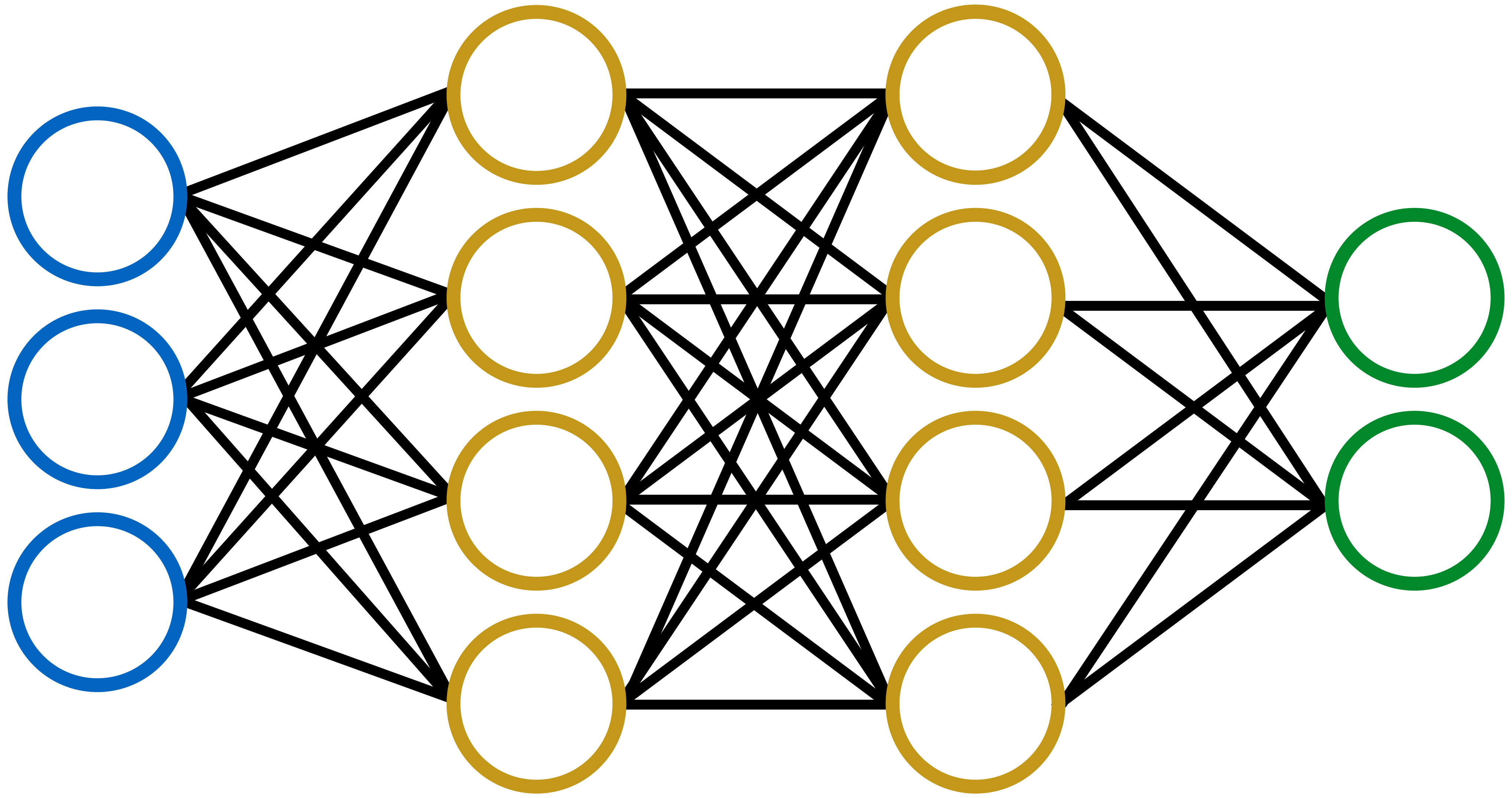


Acoustic model: a simple "feed forward" neural network

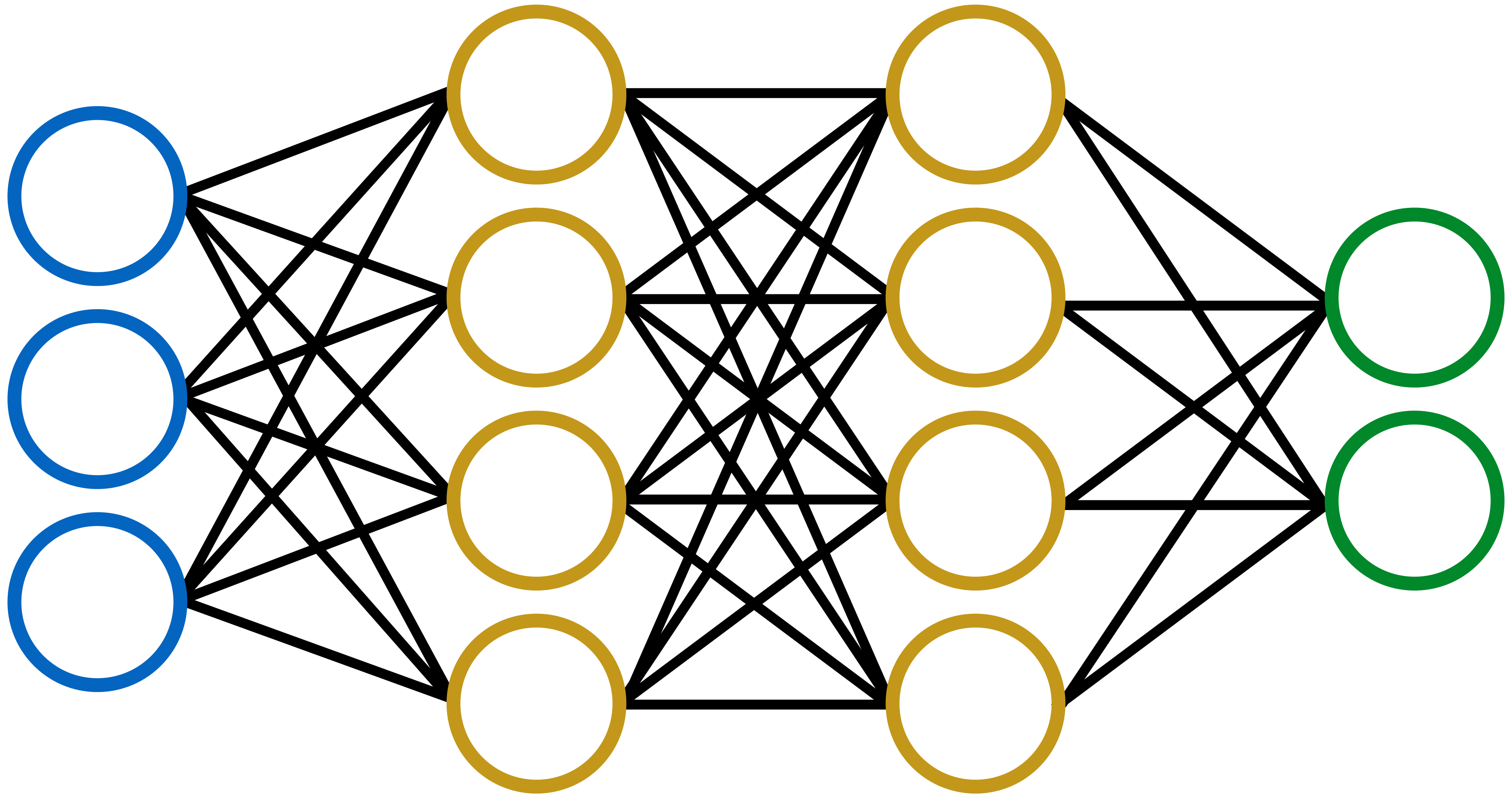


Acoustic model: a simple "feed forward" neural network

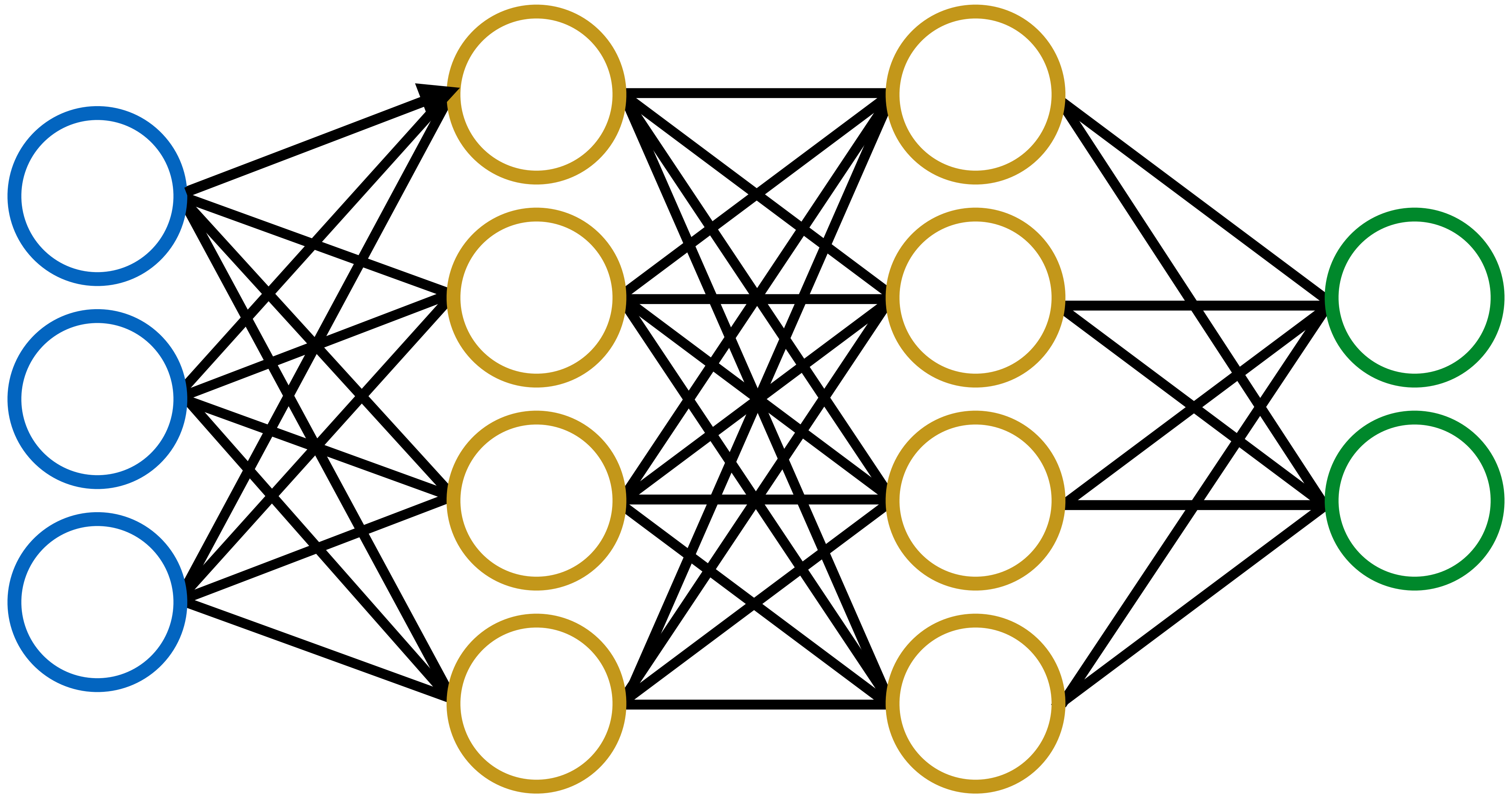
units (or "neurons"), each with an **activation function**



Acoustic model: a simple "feed forward" neural network

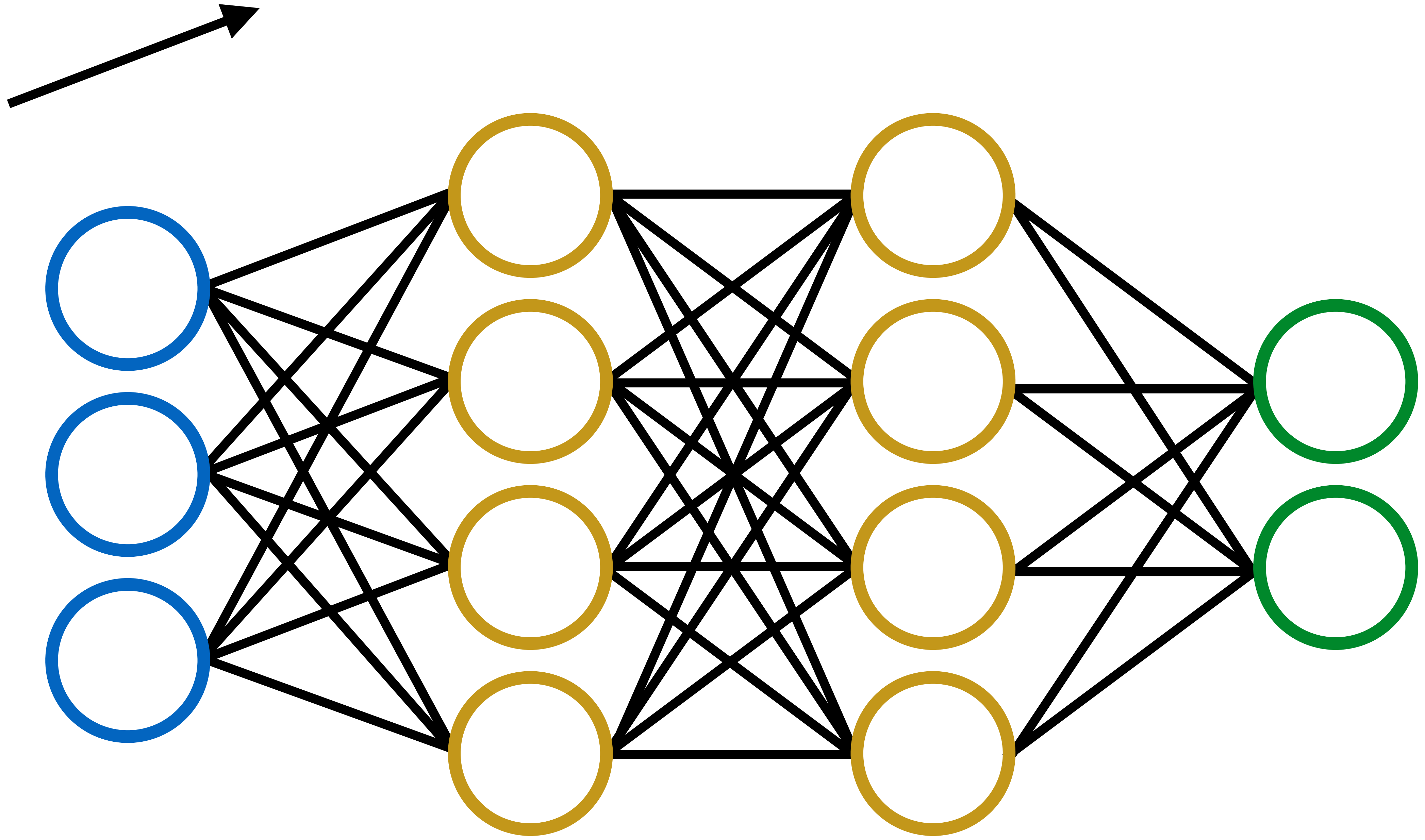


Acoustic model: a simple "feed forward" neural network

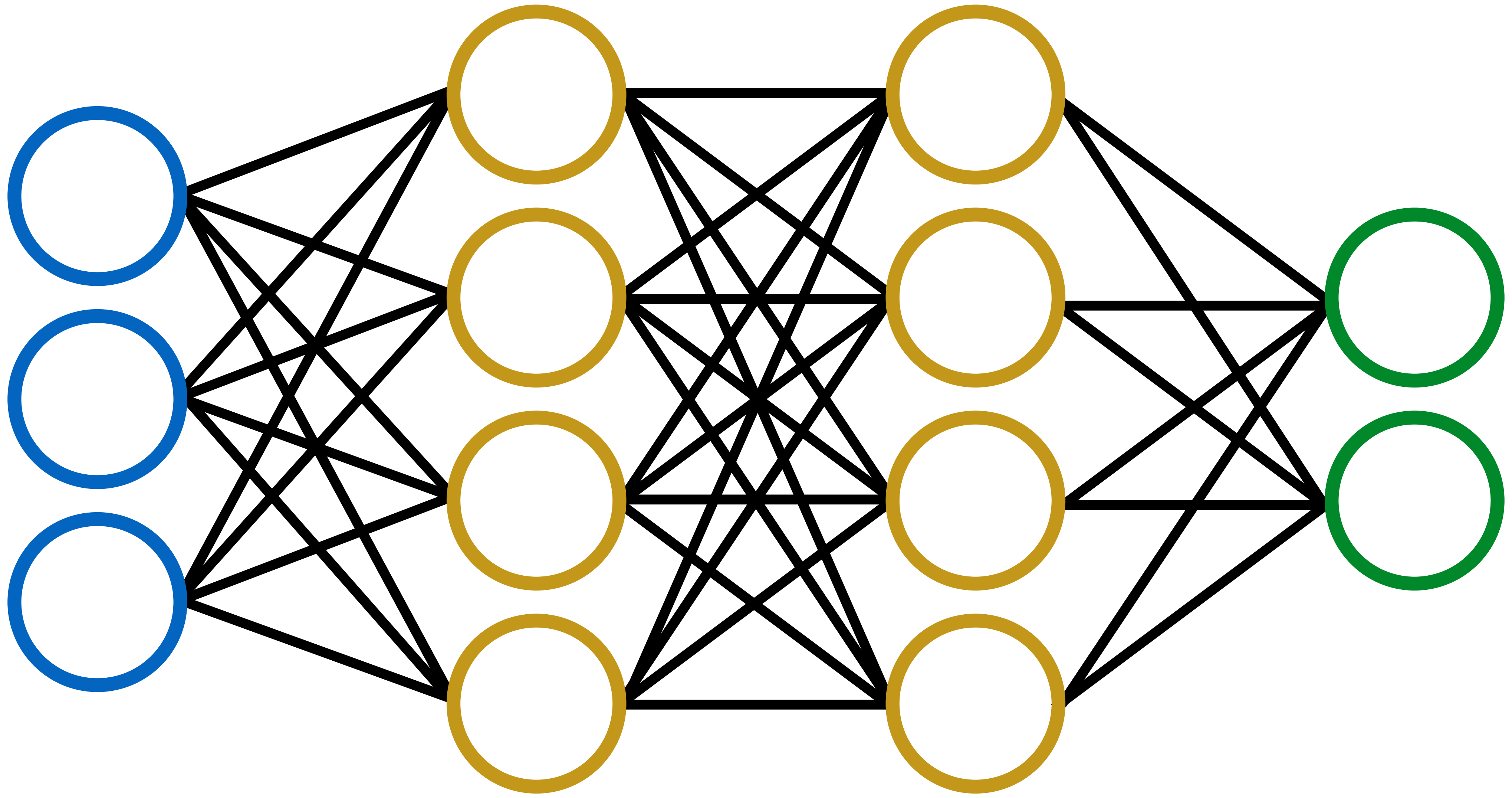


Acoustic model: a simple "feed forward" neural network

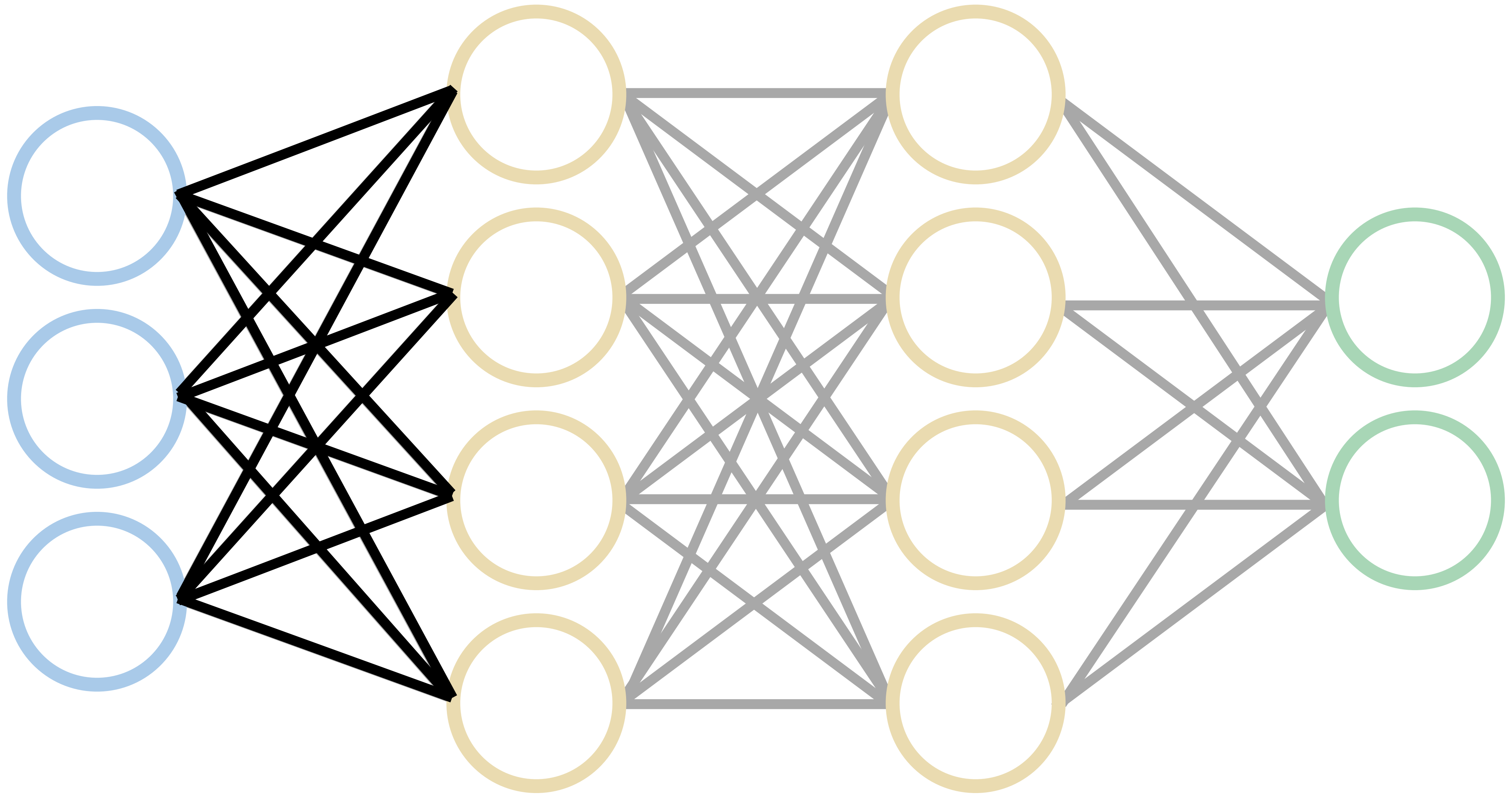
directed connections,
each with a **weight**



Acoustic model: a simple "feed forward" neural network

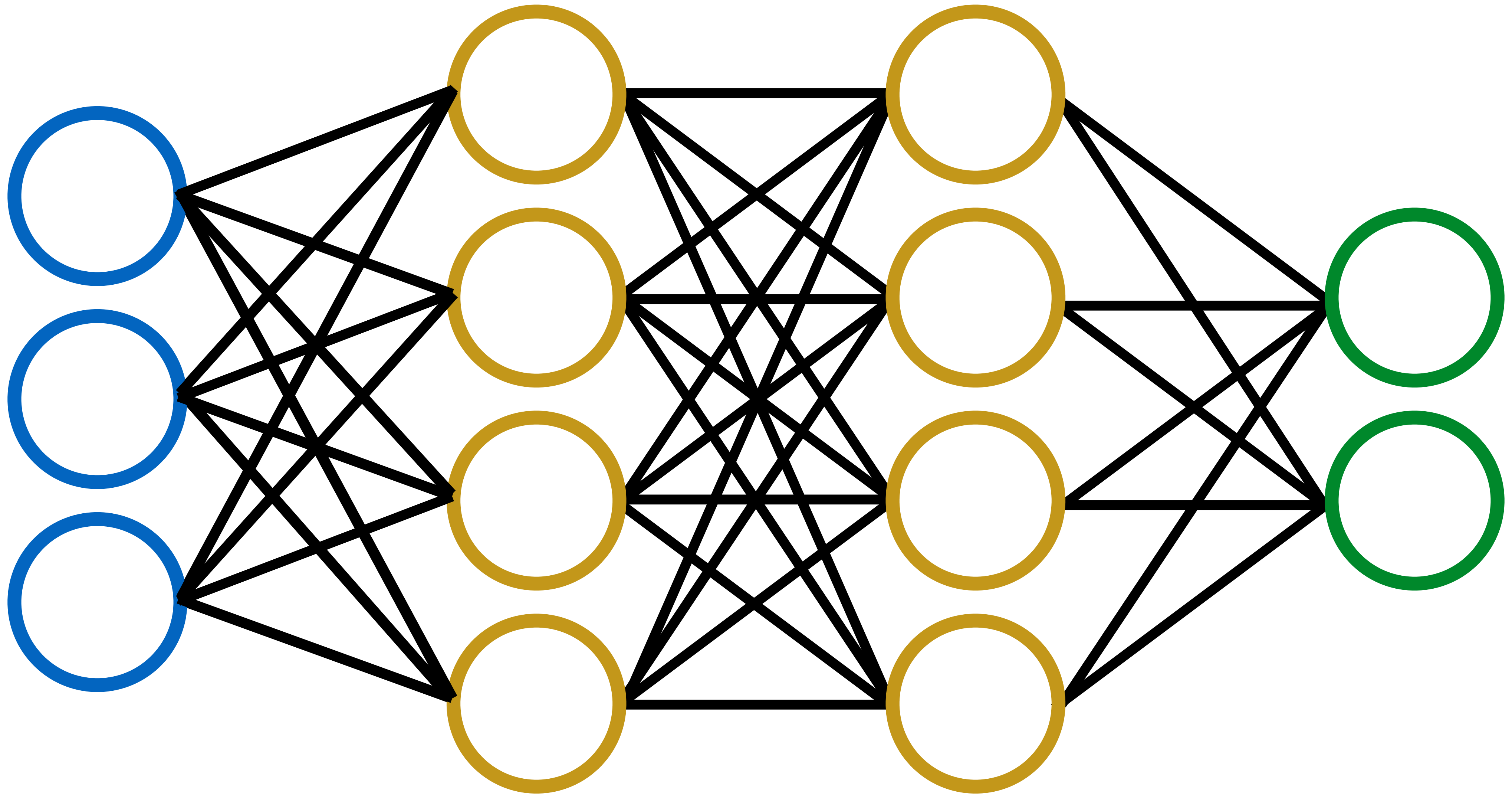


Acoustic model: a simple "feed forward" neural network

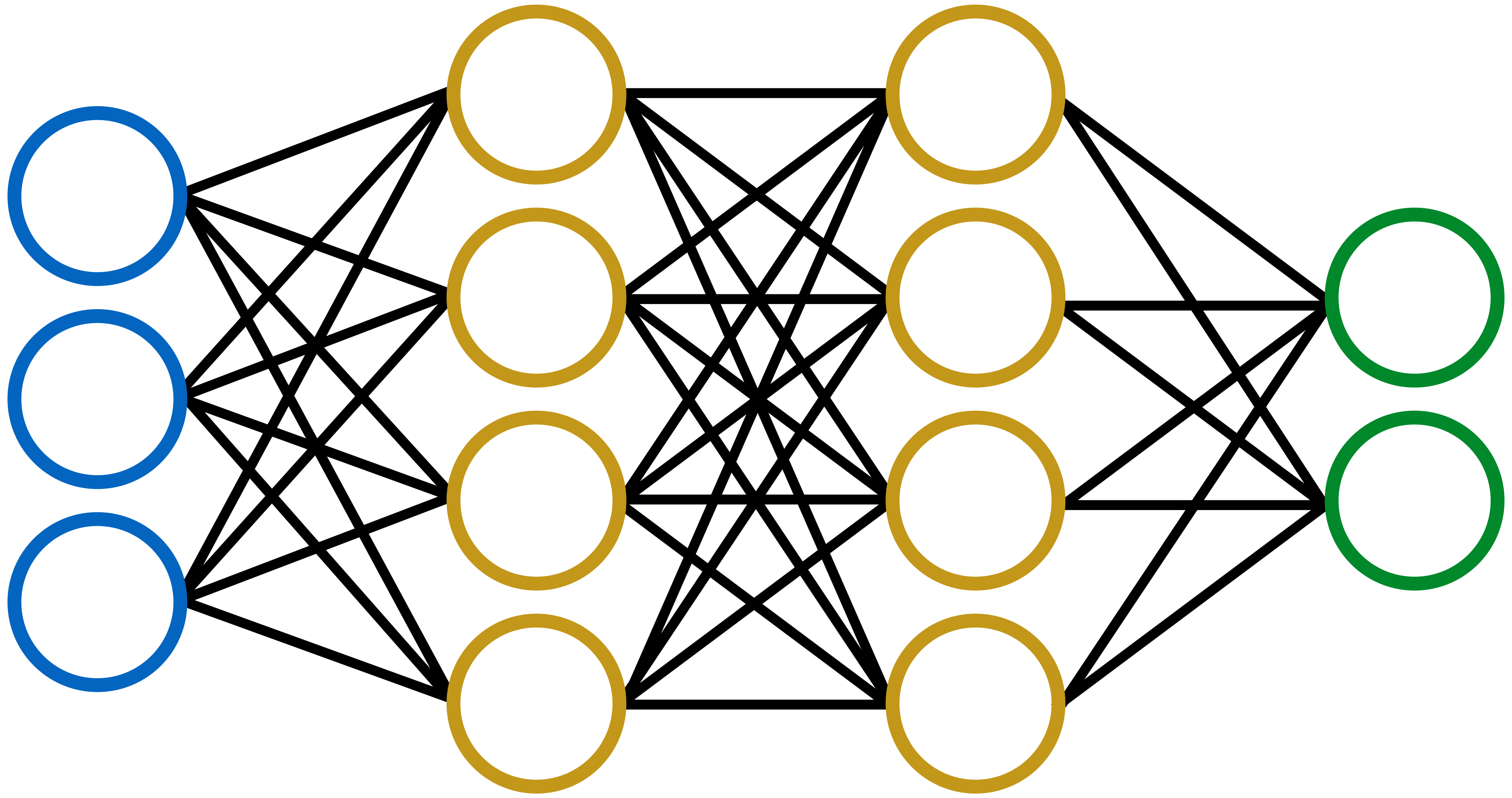


a weight **matrix**

Acoustic model: a simple "feed forward" neural network

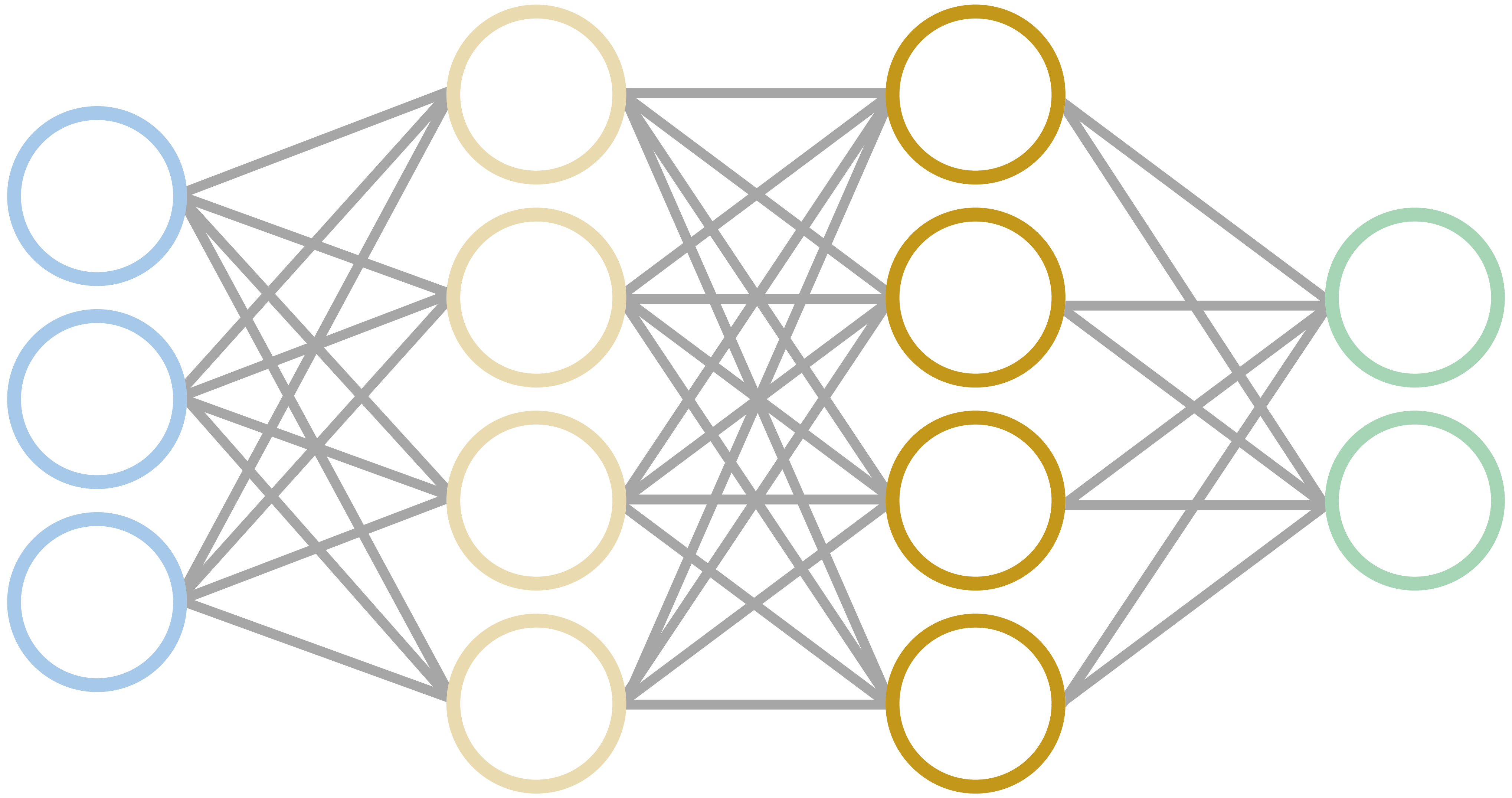


Acoustic model: a simple “feed forward” neural network

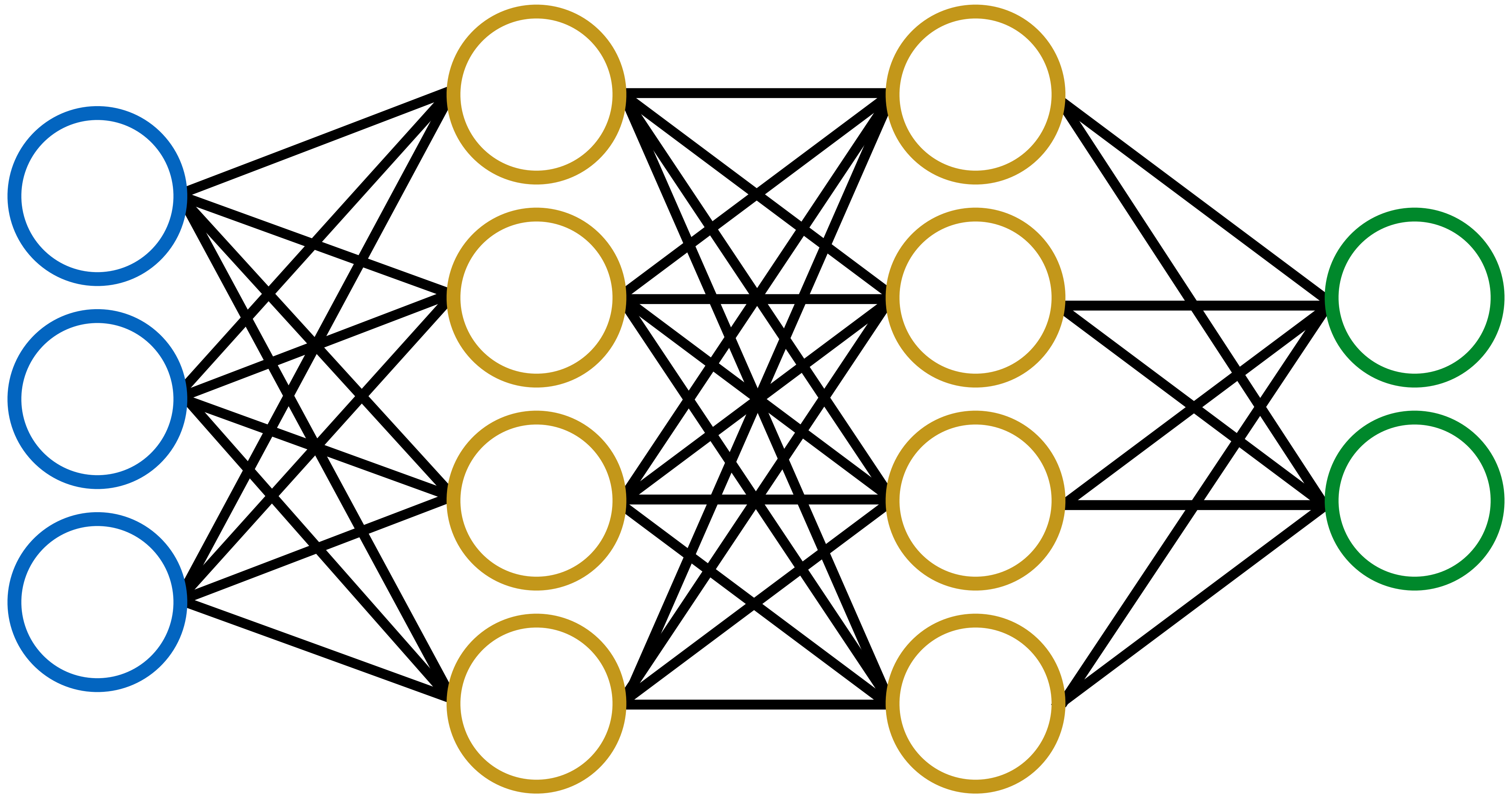


Acoustic model: a simple "feed forward" neural network

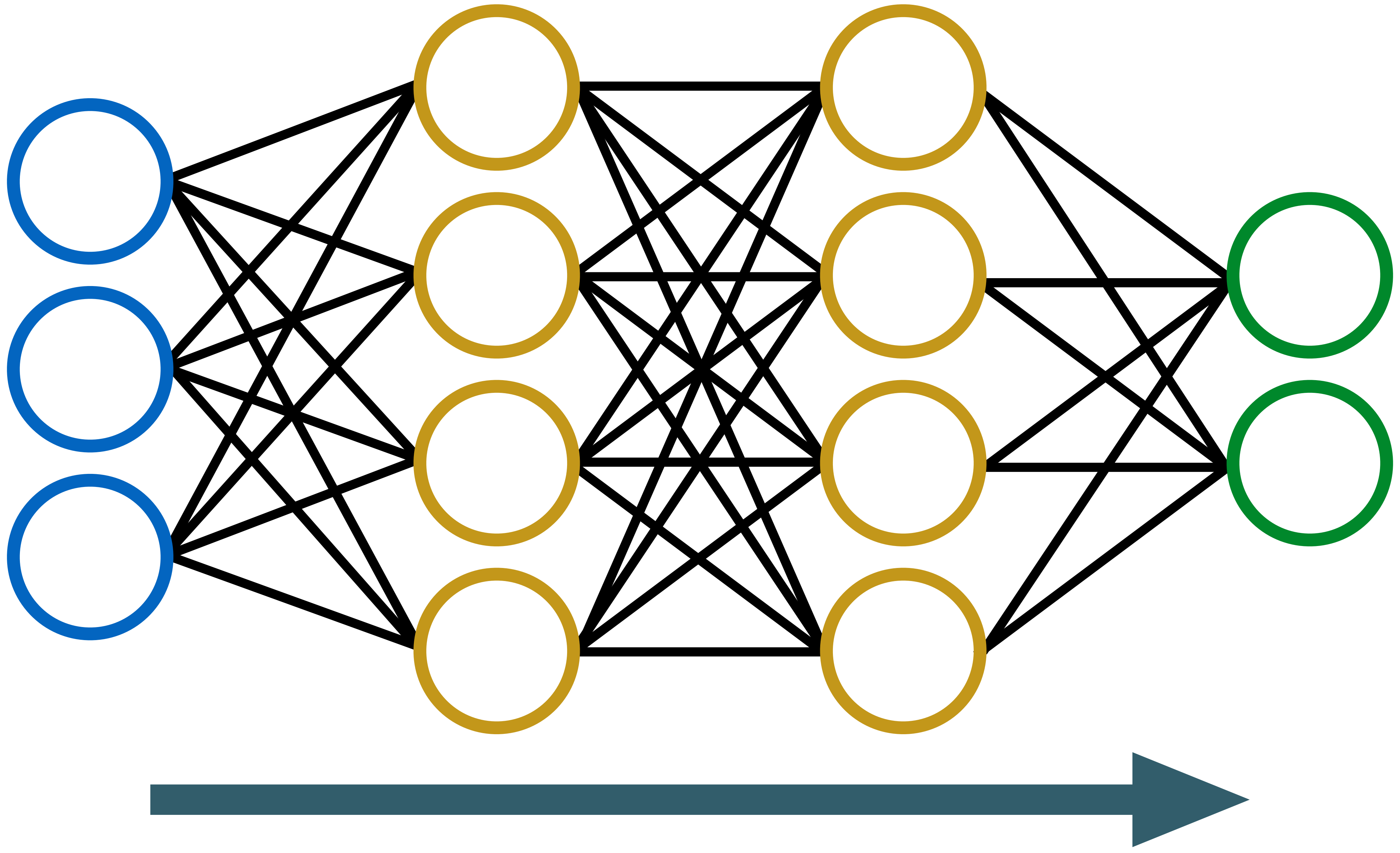
a hidden **layer**



Acoustic model: a simple “feed forward” neural network



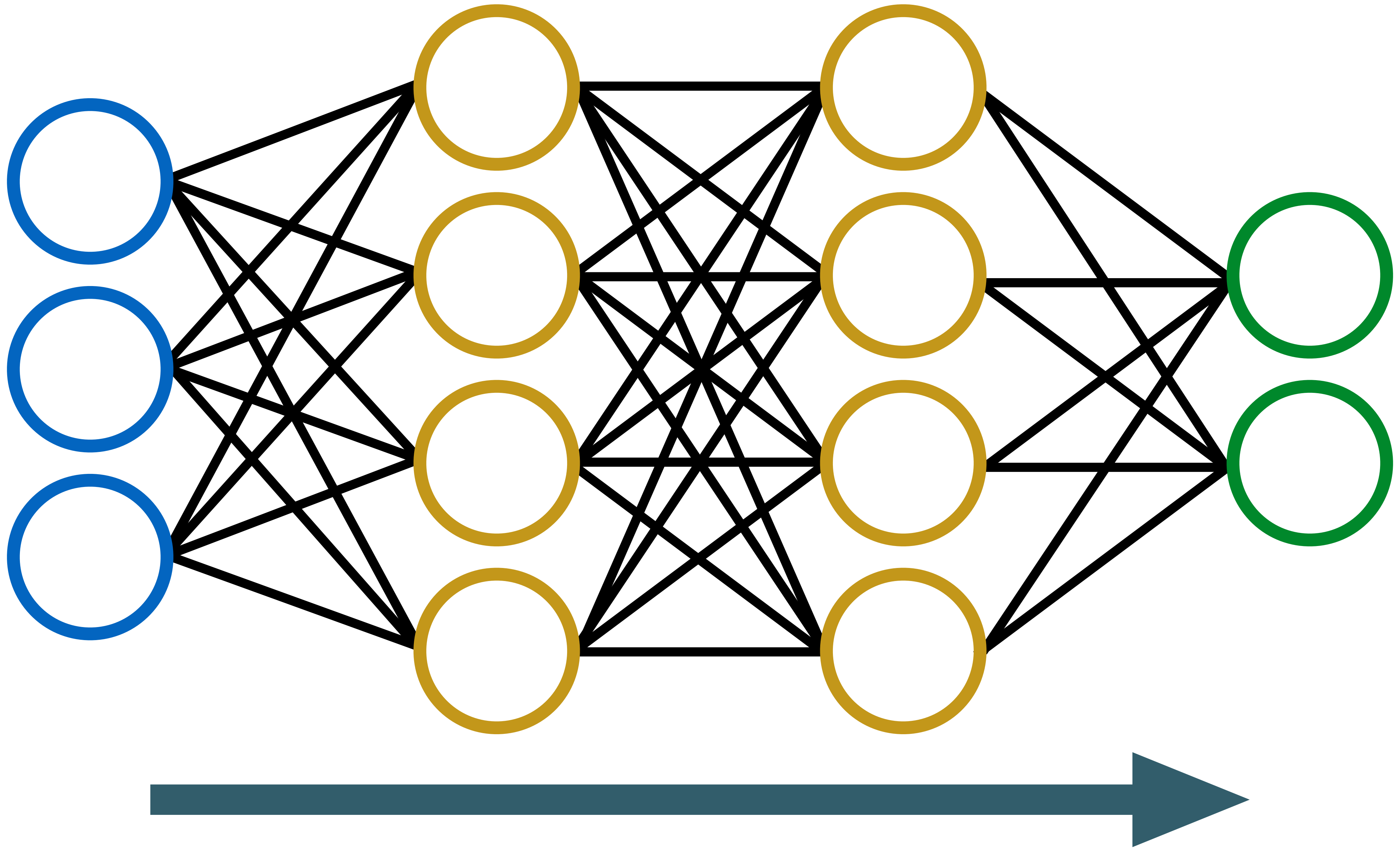
Acoustic model: a simple “feed forward” neural network



information flows in this direction

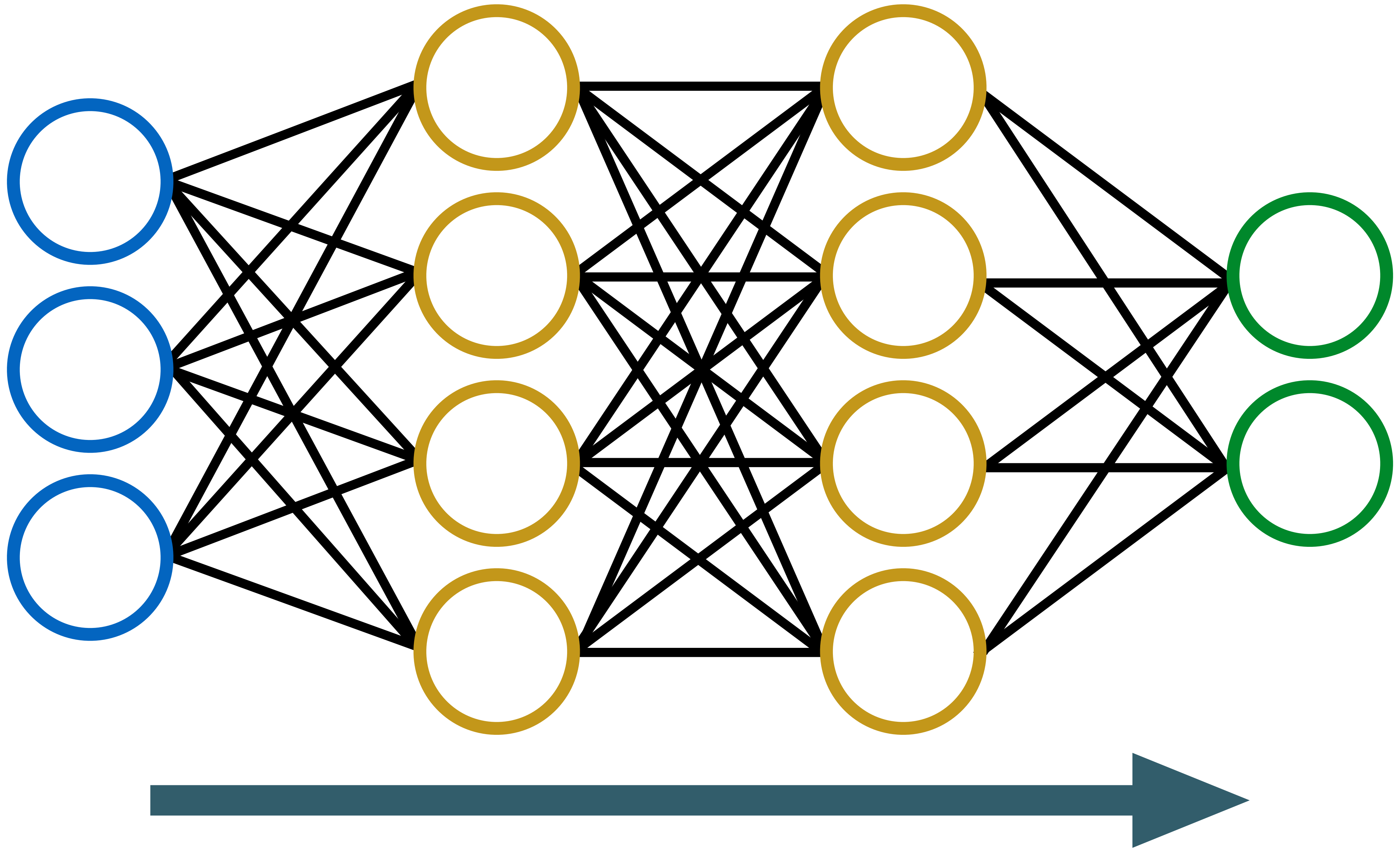
Acoustic model: a simple “feed forward” neural network

input **layer**



information flows in this direction

Acoustic model: a simple "feed forward" neural network

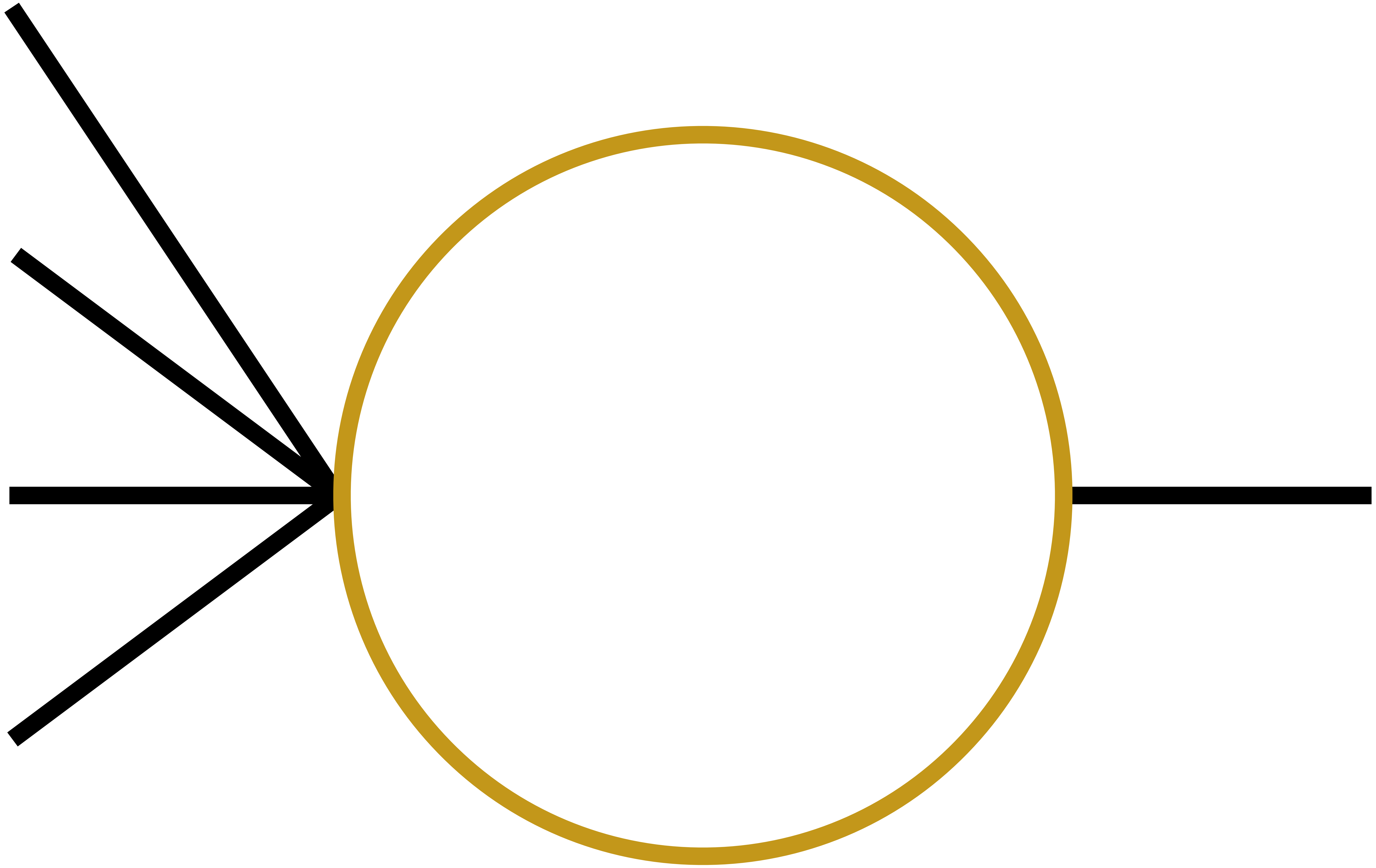


input **layer**

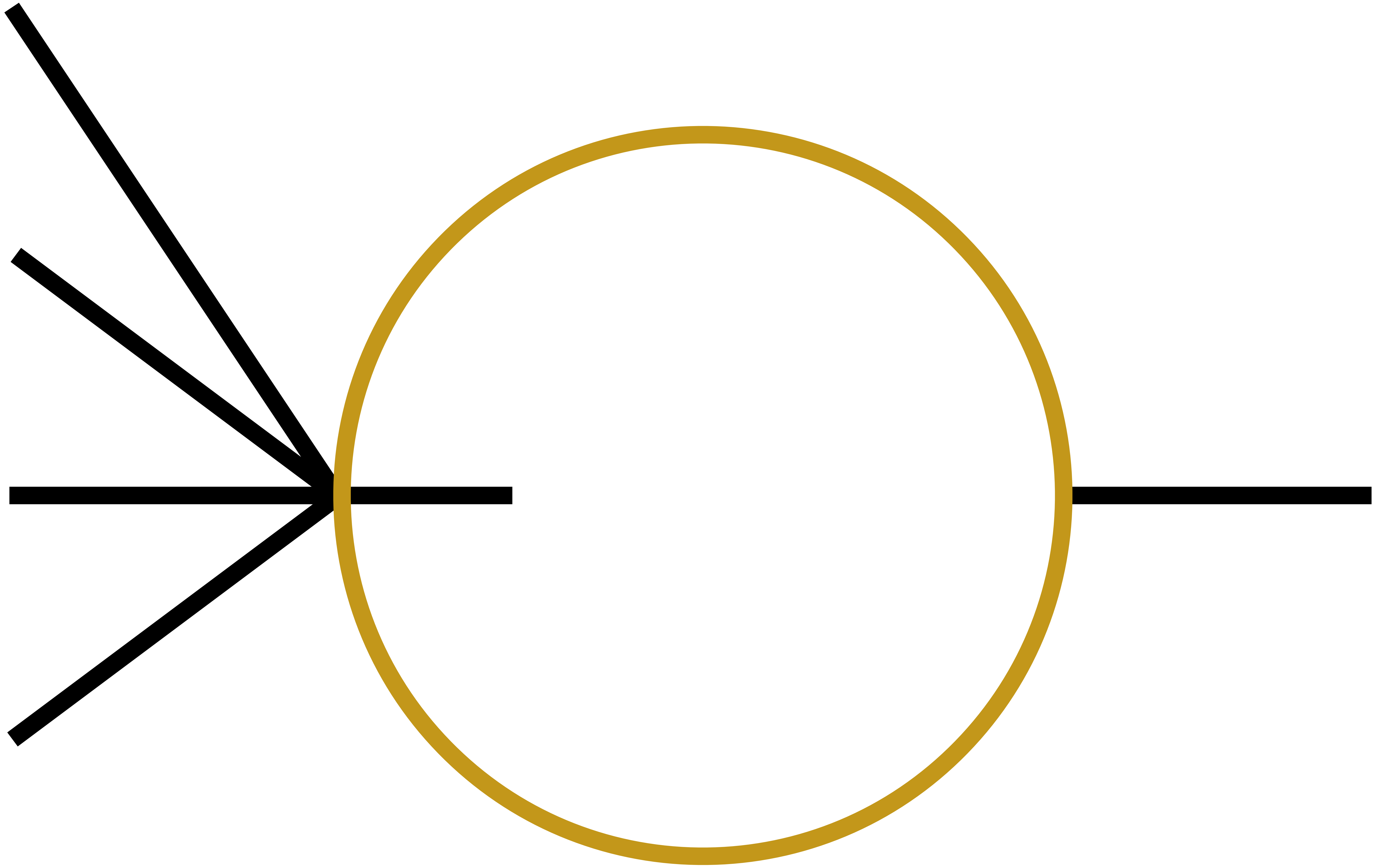
output **layer**

information flows in this direction

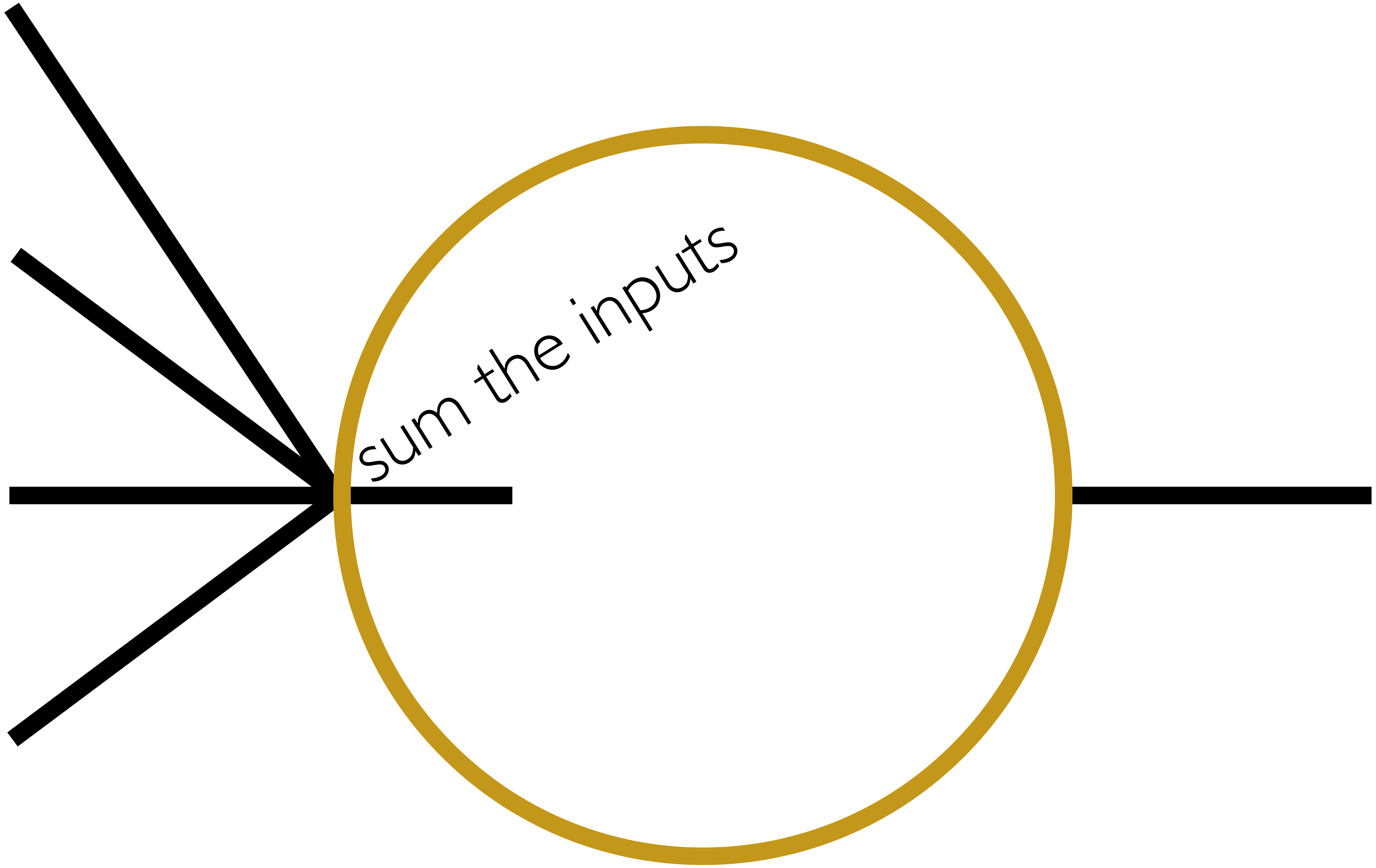
What is a unit, and what does it do?



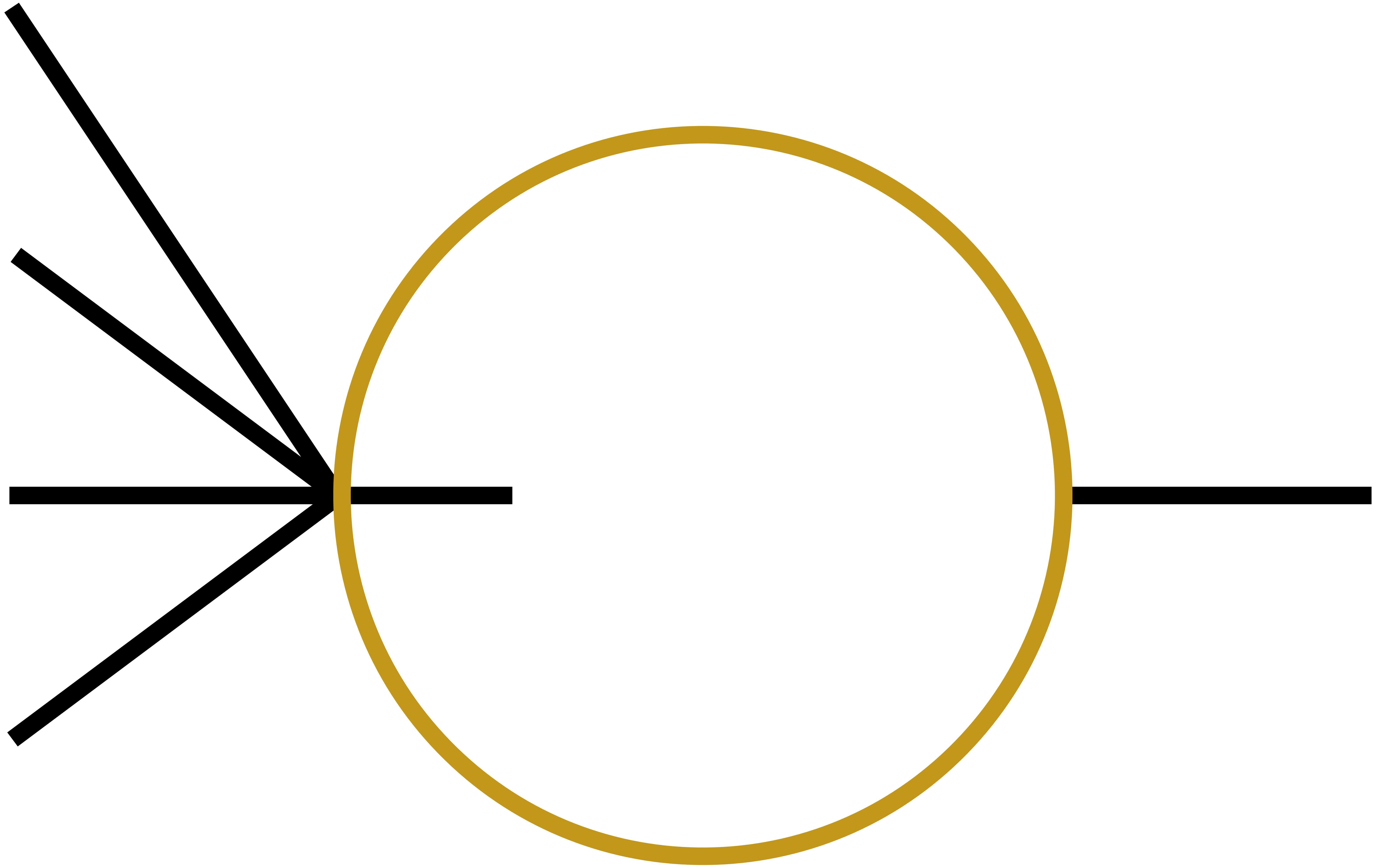
What is a unit, and what does it do?



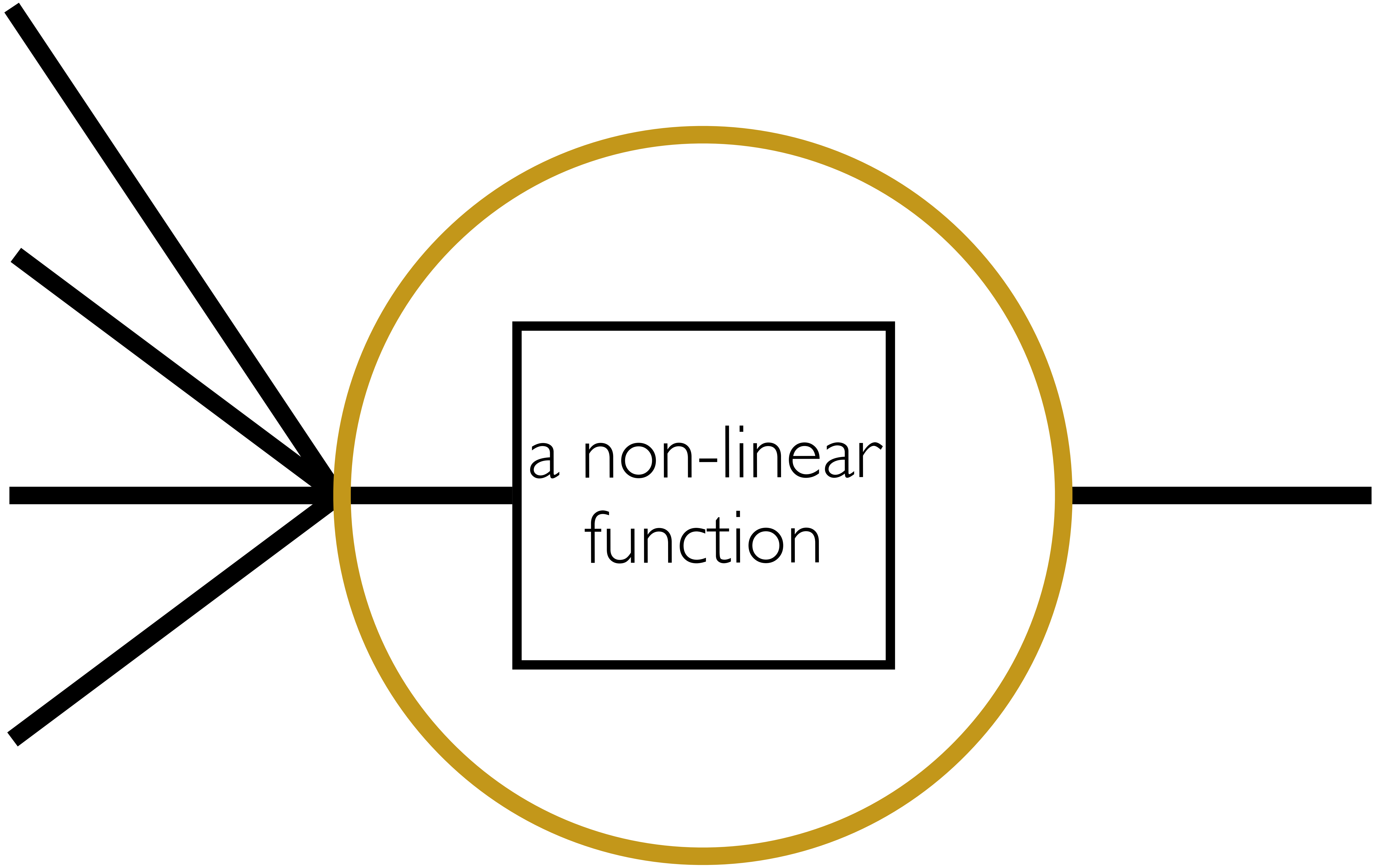
What is a unit, and what does it do?



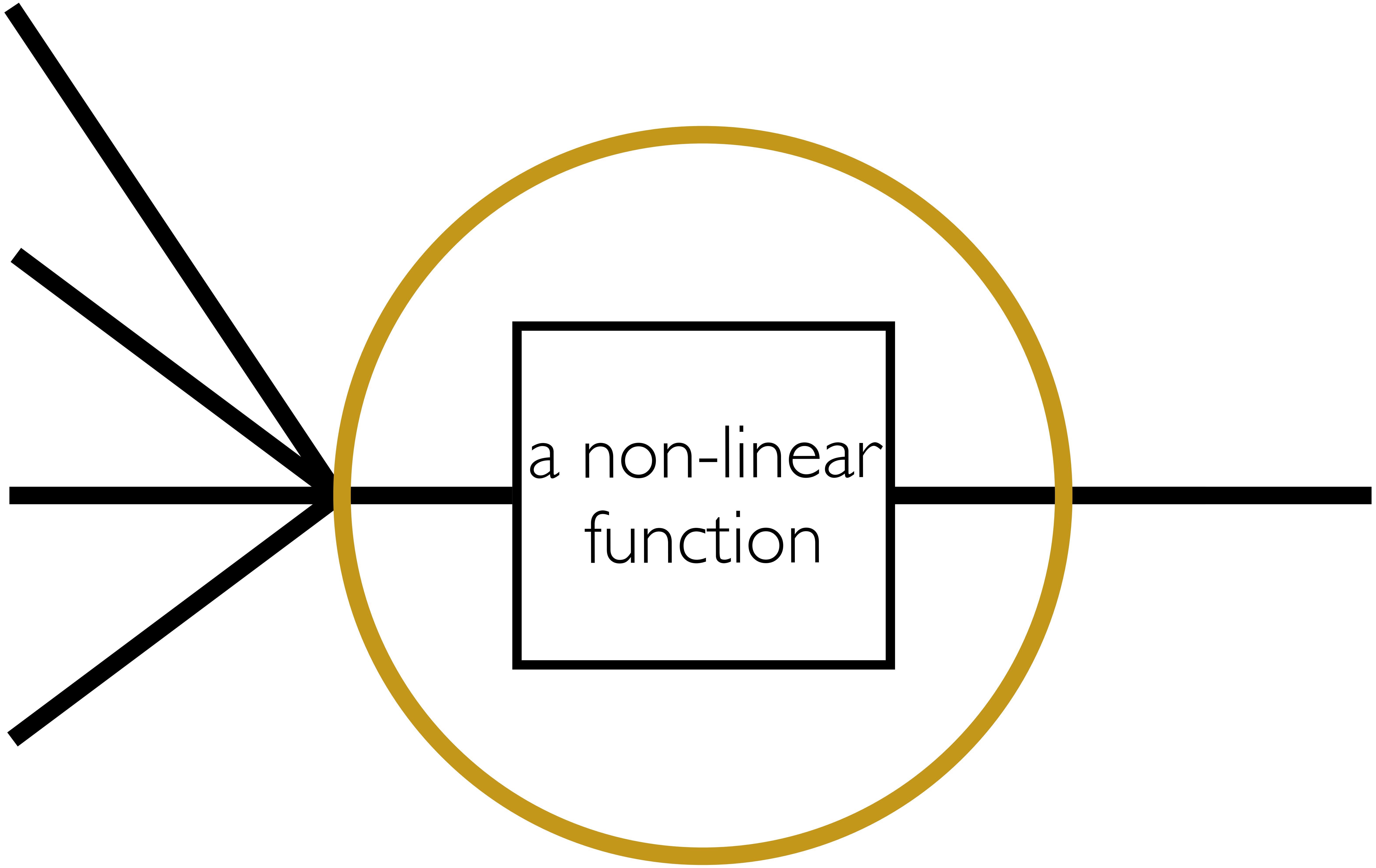
What is a unit, and what does it do?



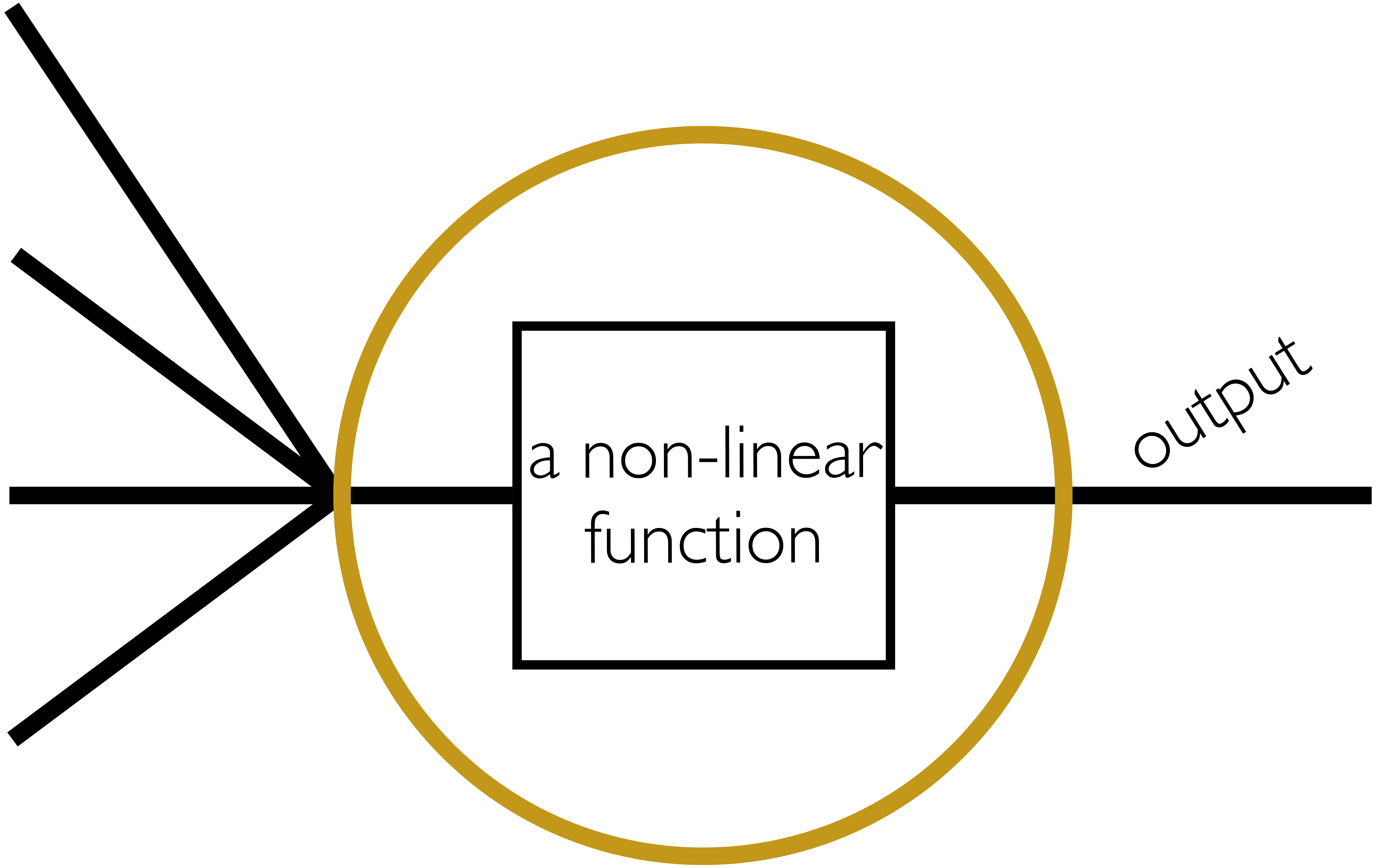
What is a unit, and what does it do?



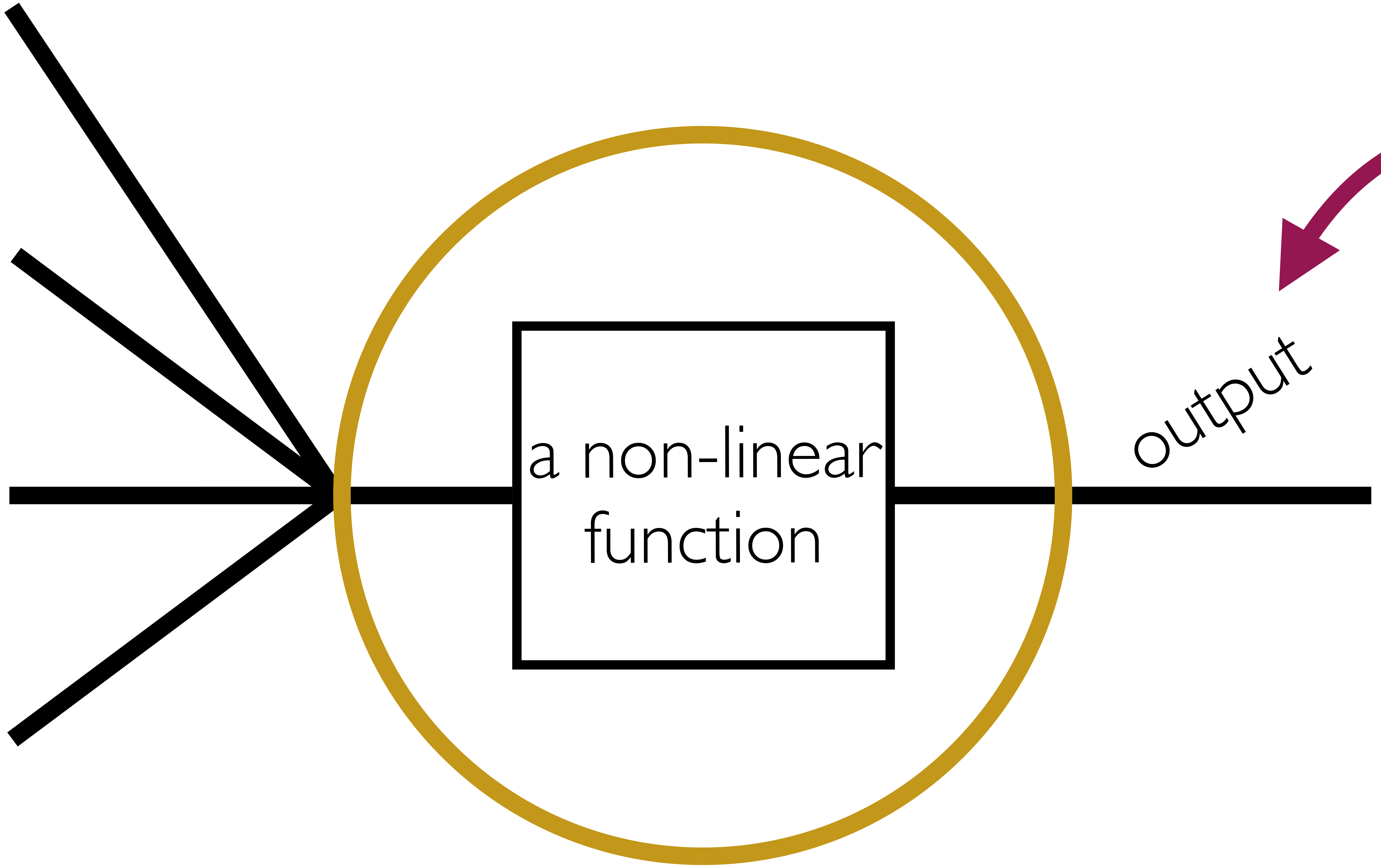
What is a unit, and what does it do?



What is a unit, and what does it do?

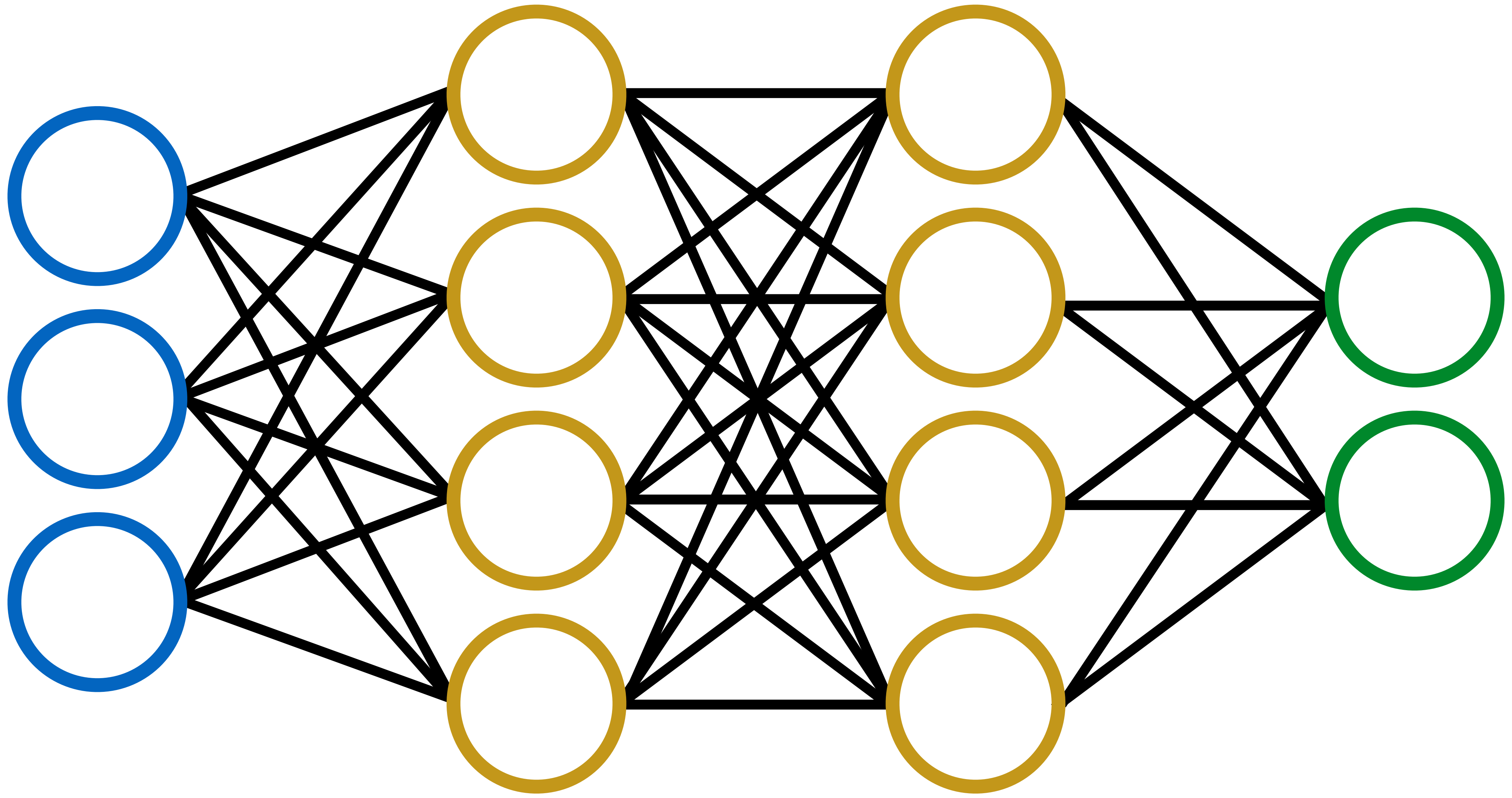


What is a unit, and what does it do?

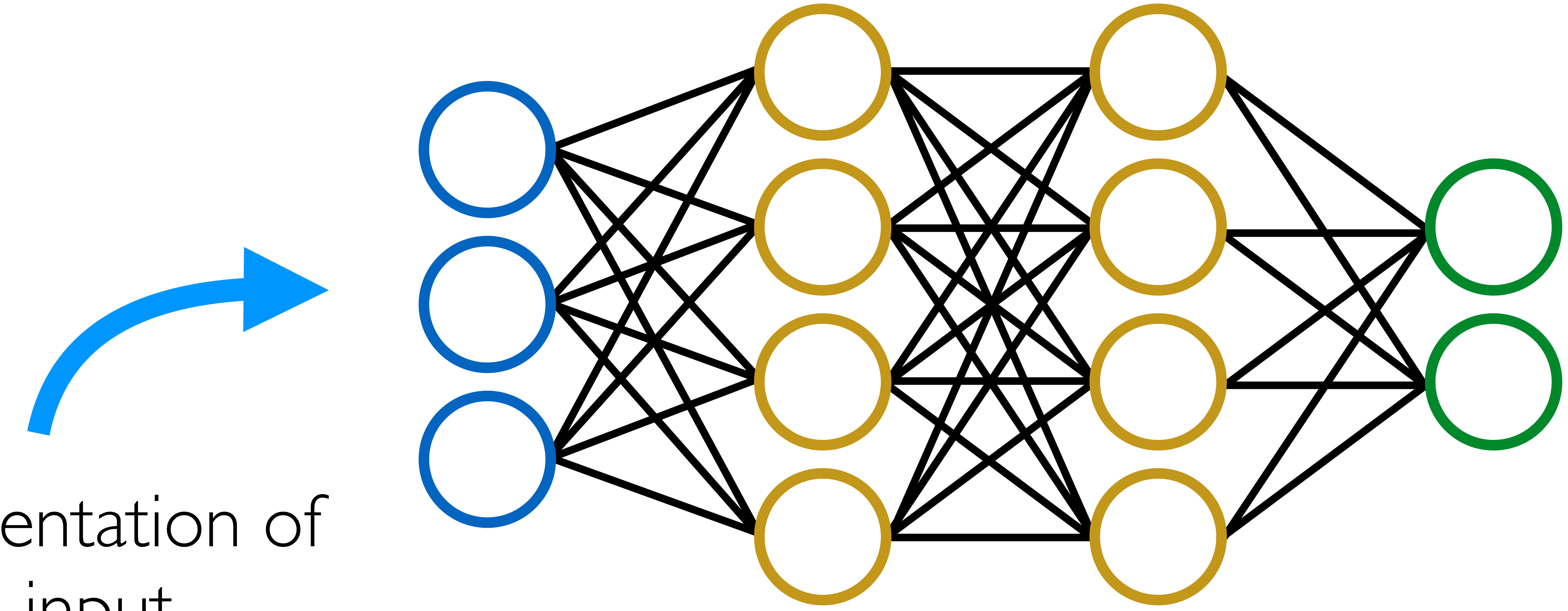


usually called the “activation”

What are all those layers for?

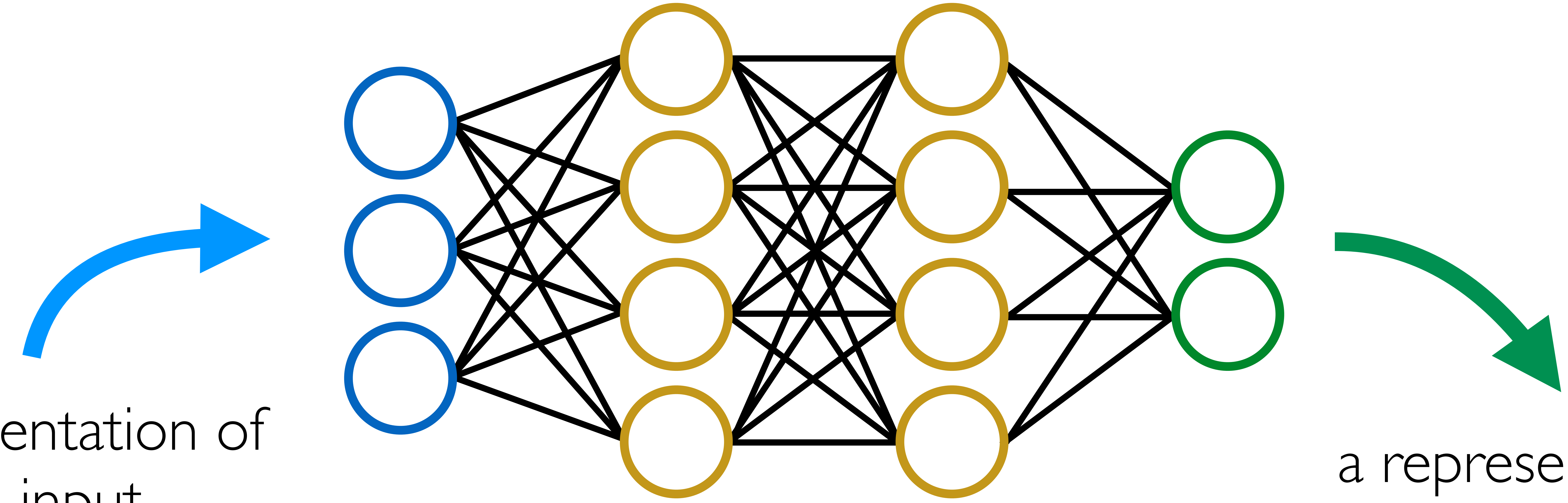


What are all those layers for?



a representation of
the input

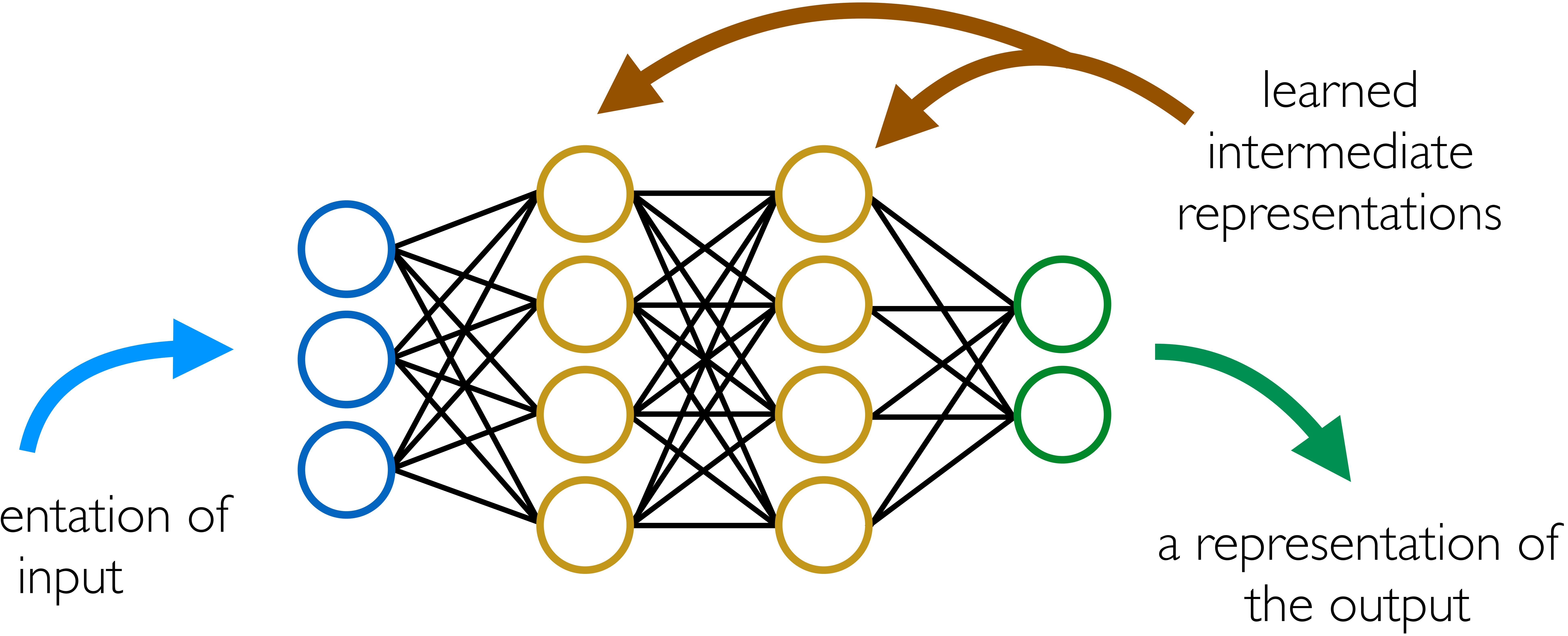
What are all those layers for?



a representation of
the input

a representation of
the output

What are all those layers for?

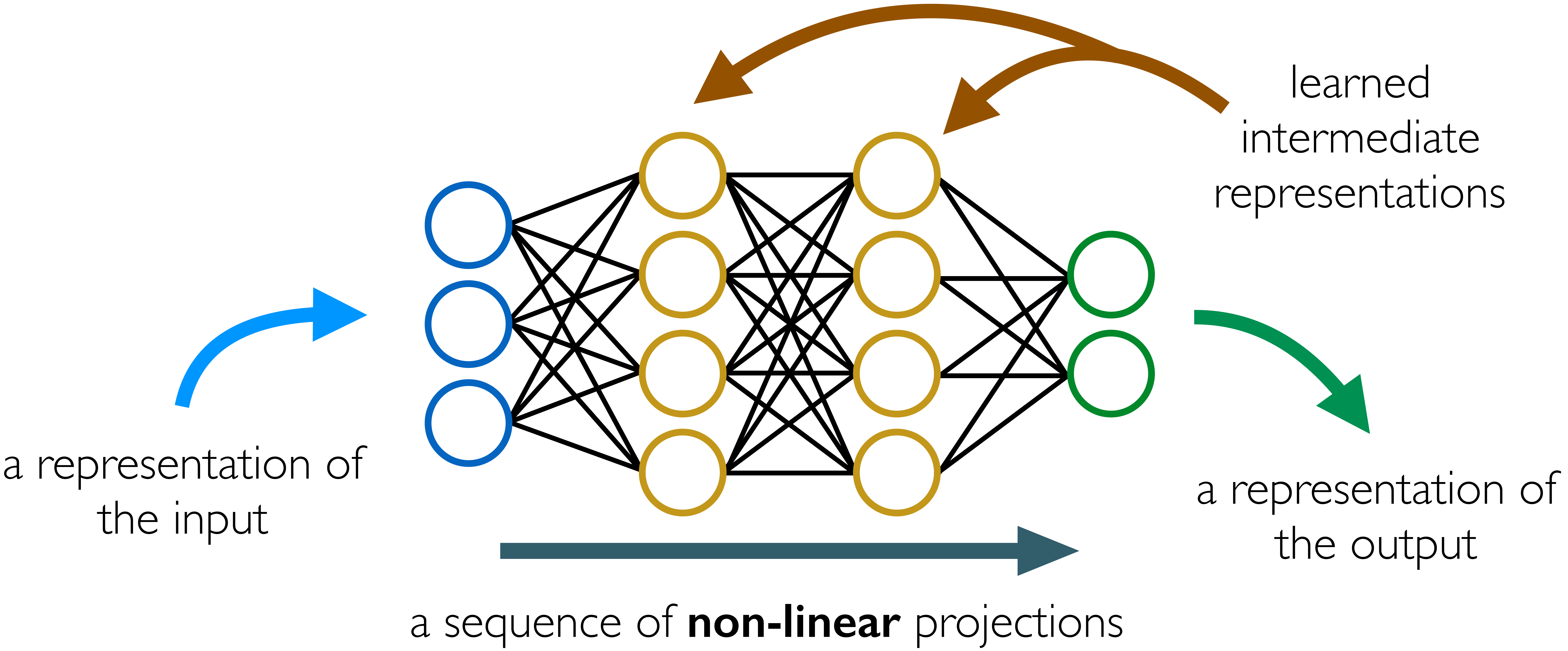


learned intermediate representations

a representation of the input

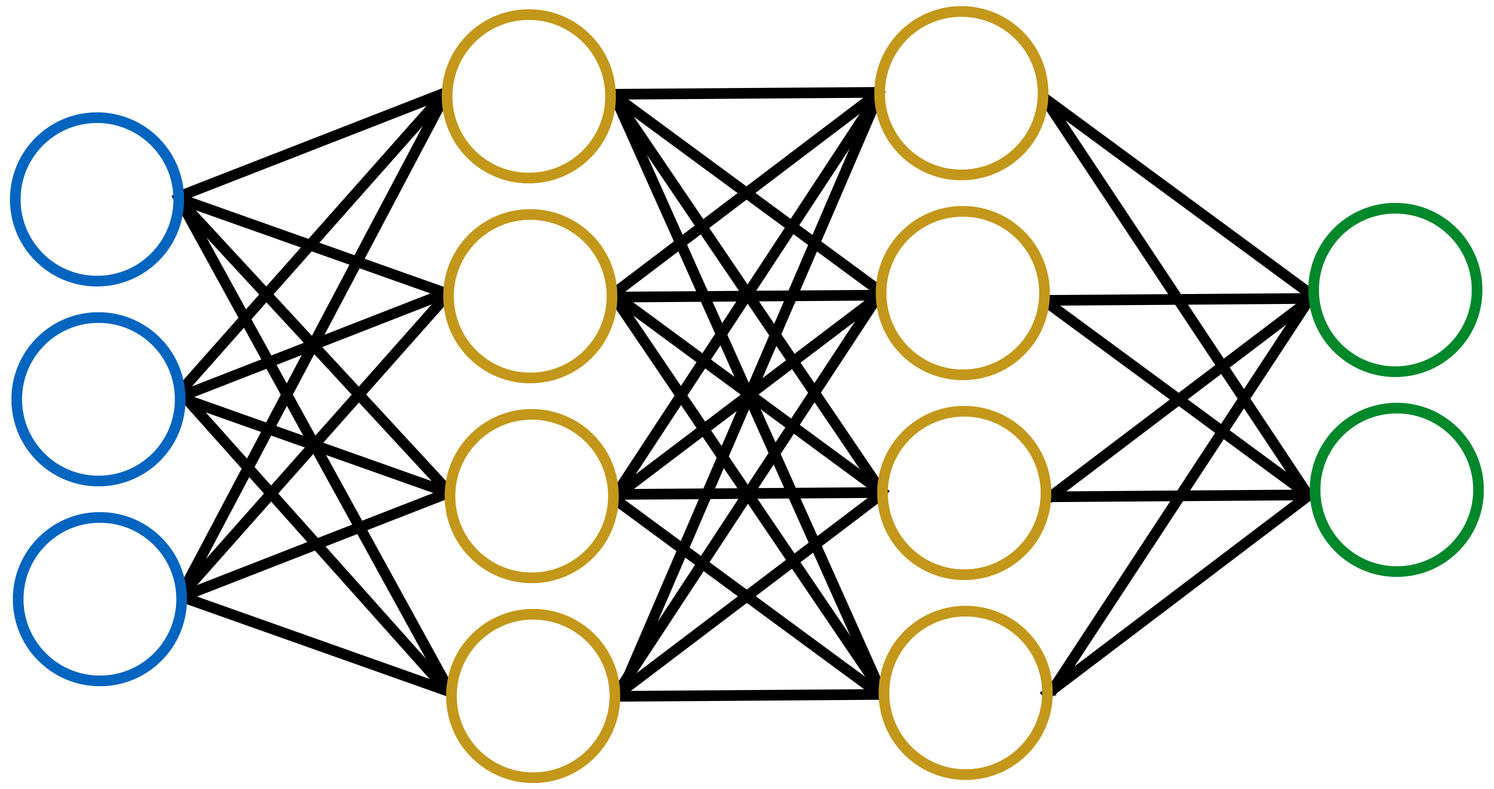
a representation of the output

What are all those layers for?


learned intermediate representations
a representation of the input
a representation of the output
a sequence of **non-linear** projections

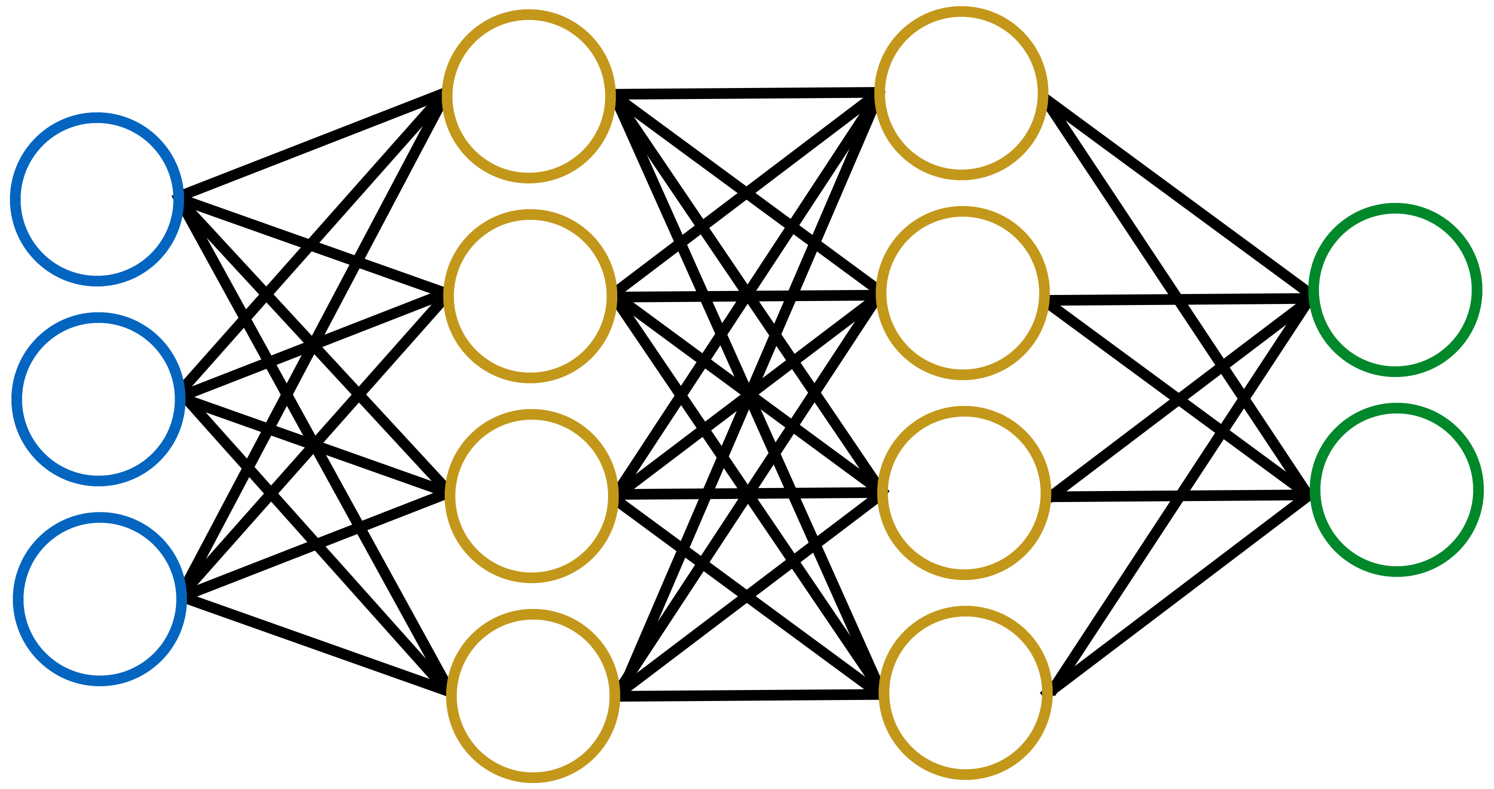
Synthesis with a neural network

```
[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]
…
```



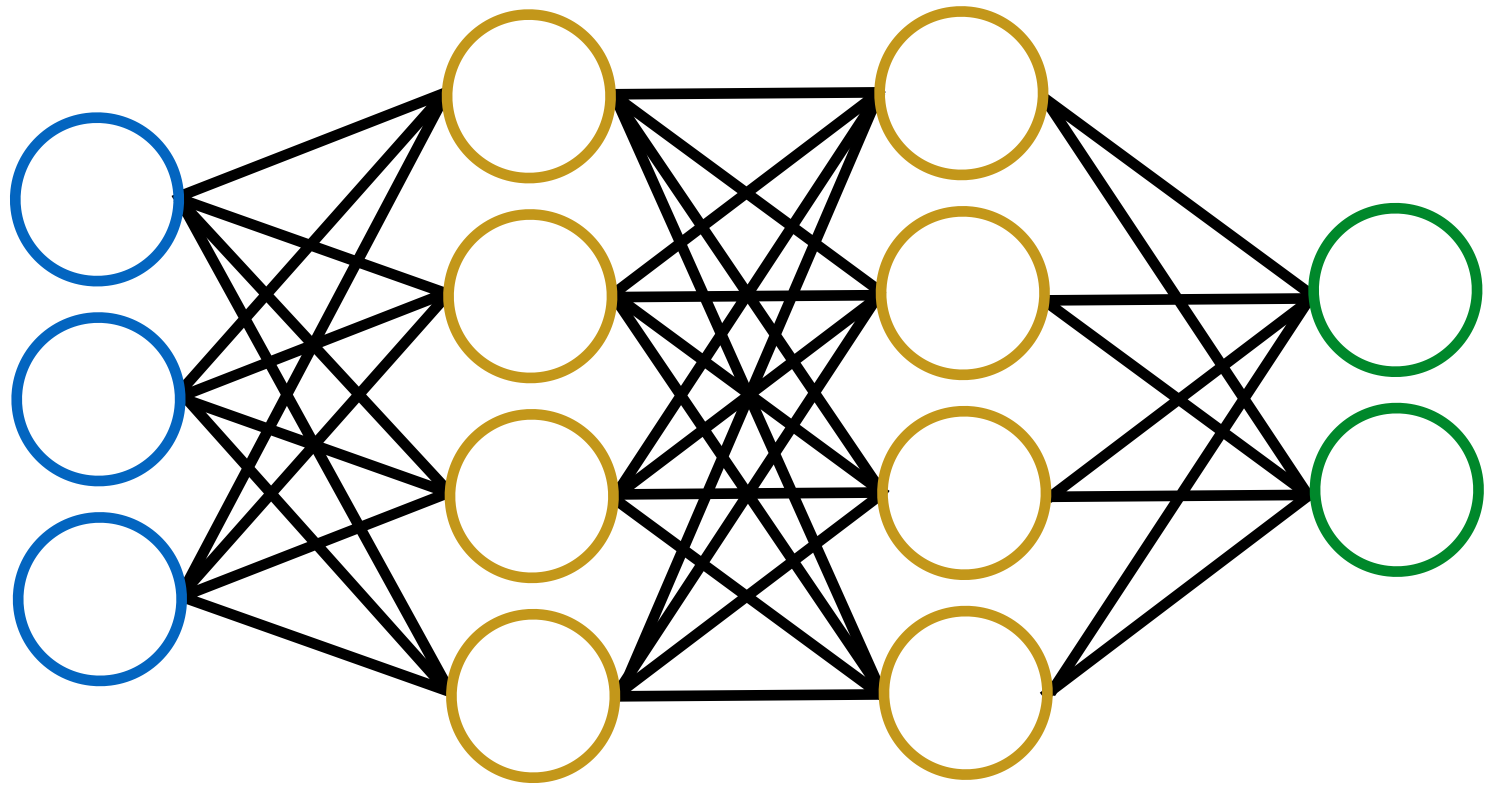
Synthesis with a neural network

```
[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]
...
```



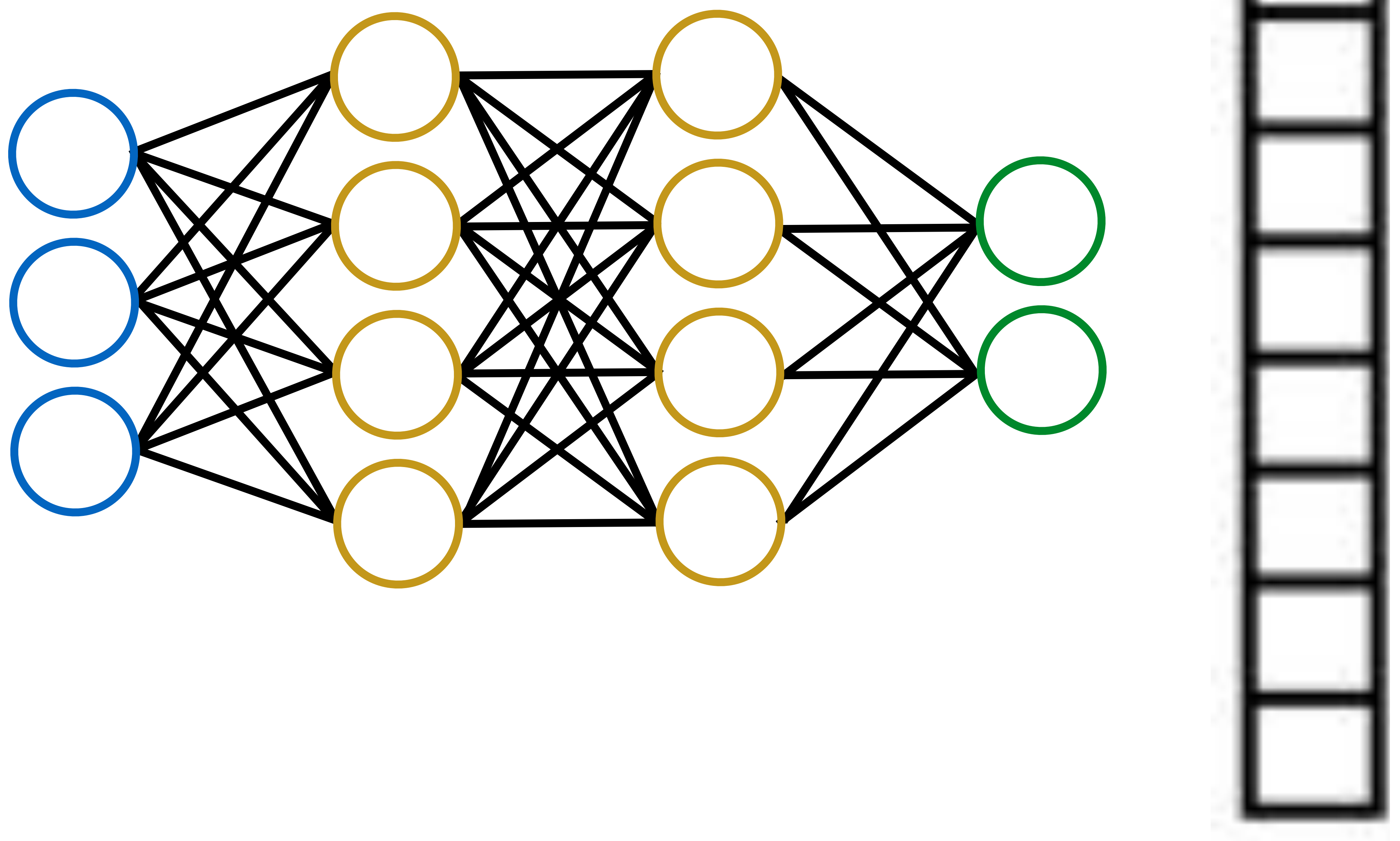
Synthesis with a neural network

```
[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]
...
```



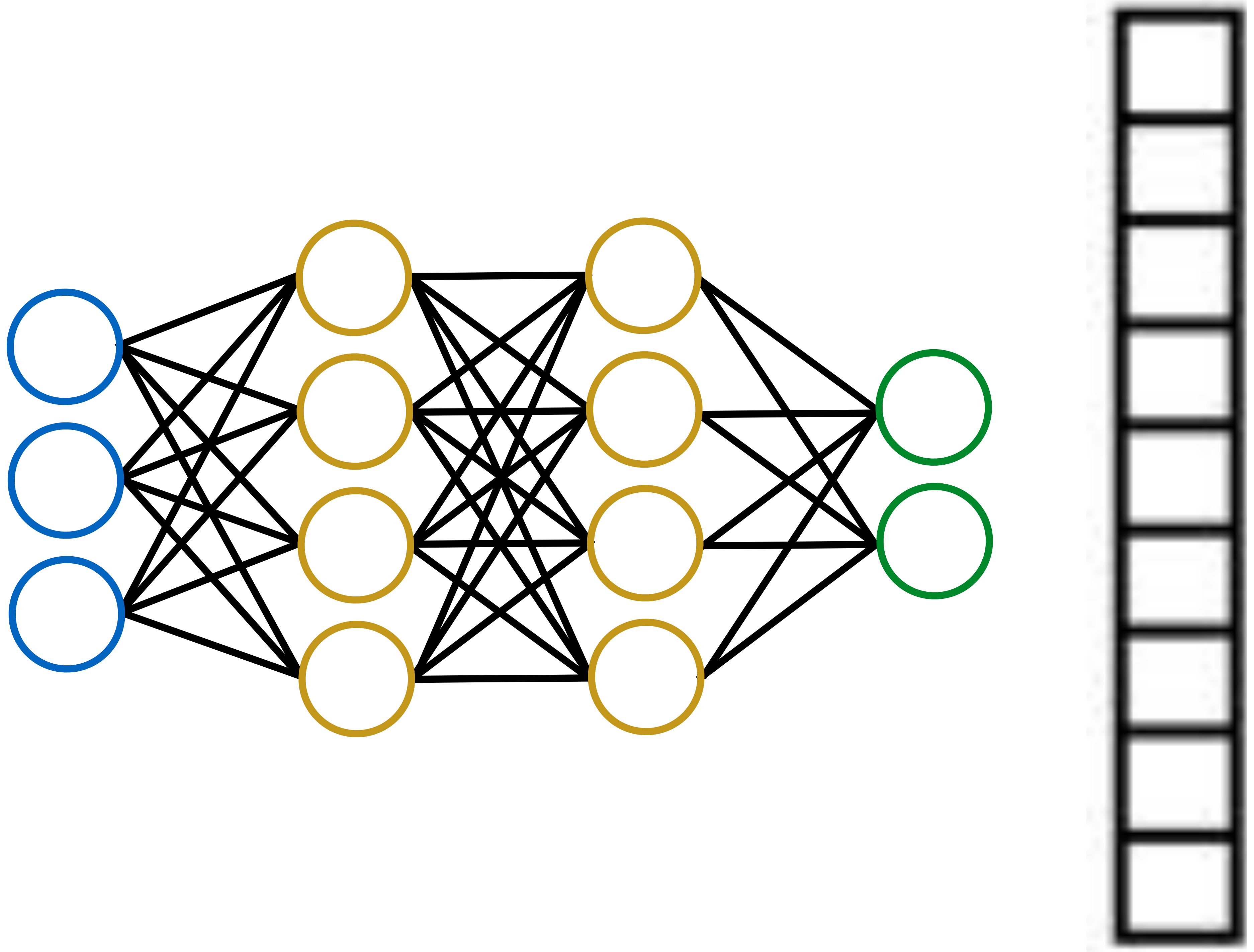
Synthesis with a neural network

```
[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]
...
```



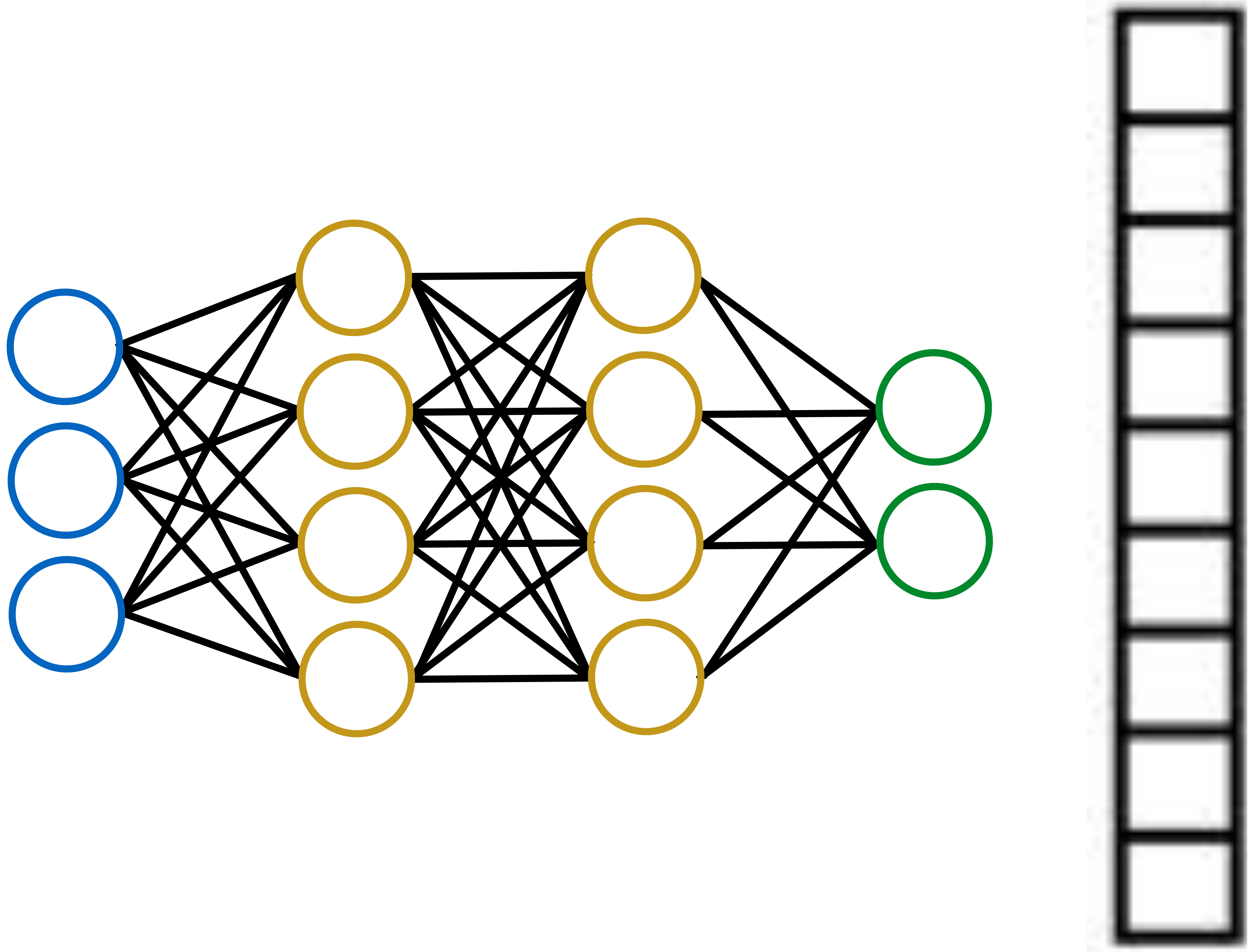
Synthesis with a neural network

```
...
[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]
...
```



Synthesis with a neural network

```
...
[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]
...
```



Synthesis with a neural network

...

[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]

...

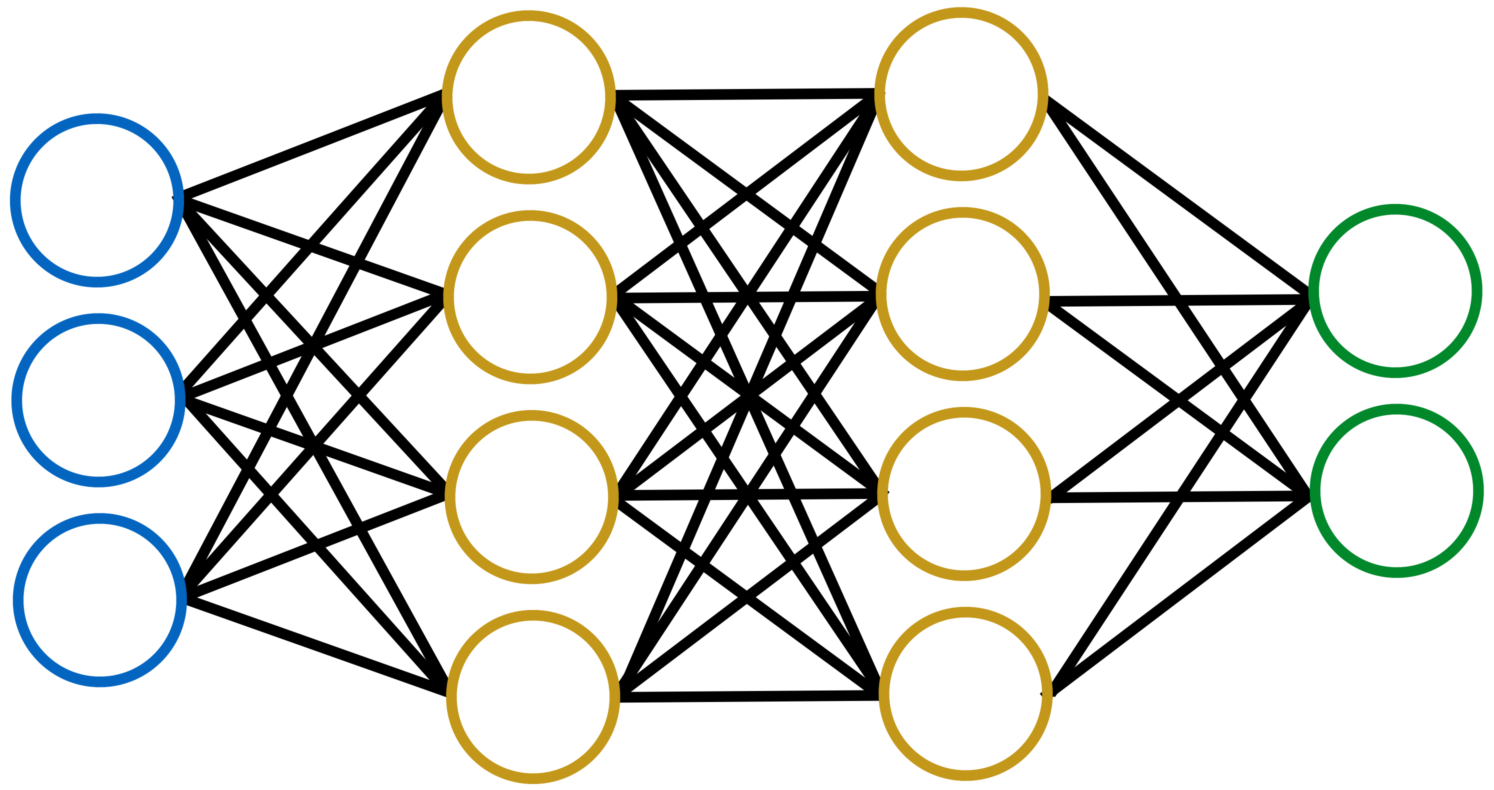
[0 0 1 0 0 1 0 1 1 0 ... 1.0 1.0]

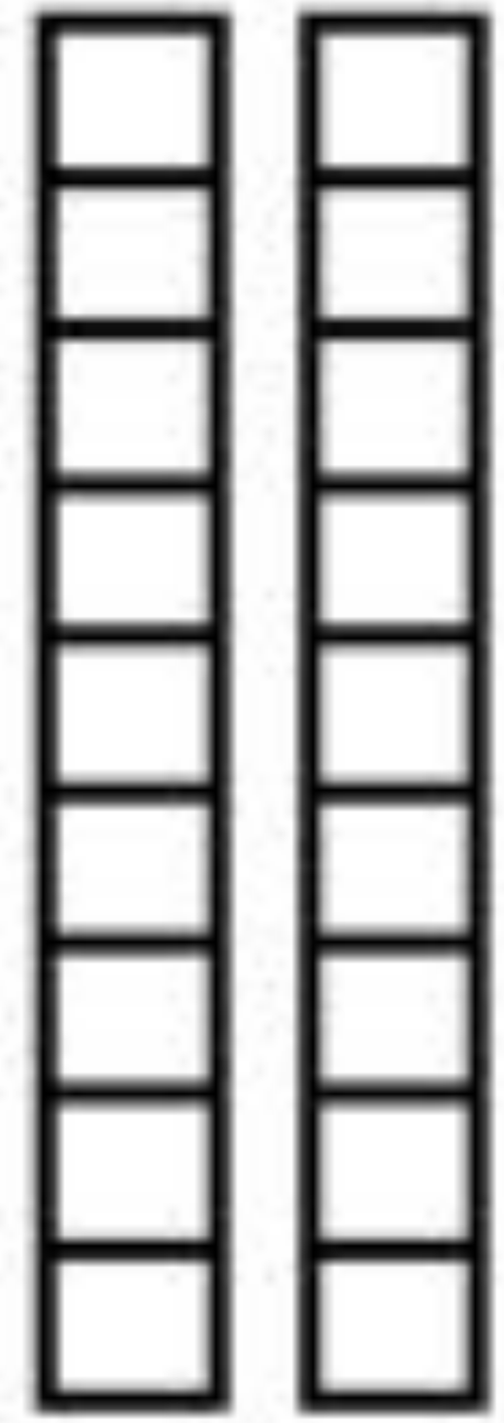
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.0]

[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]

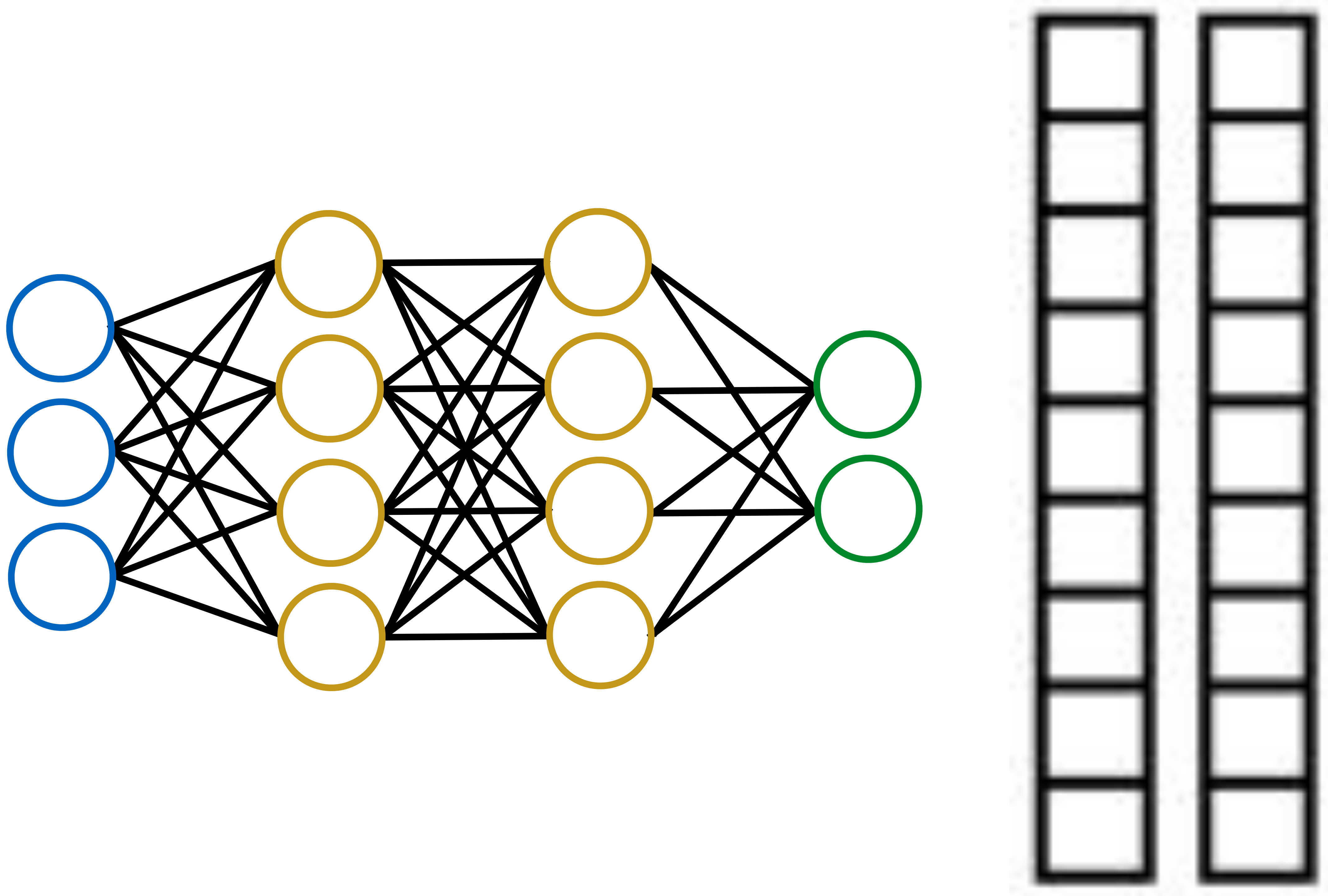
...



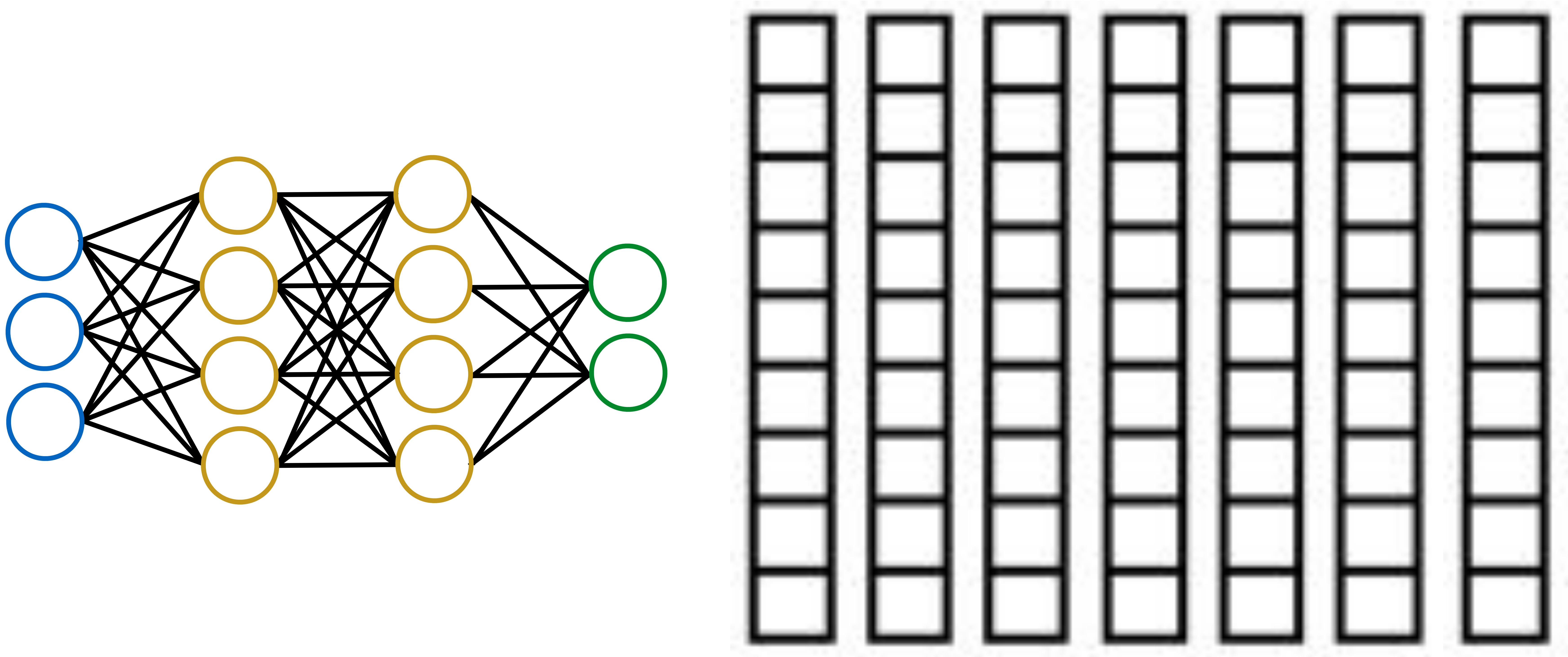


Synthesis with a neural network

```
...
[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]
...
```

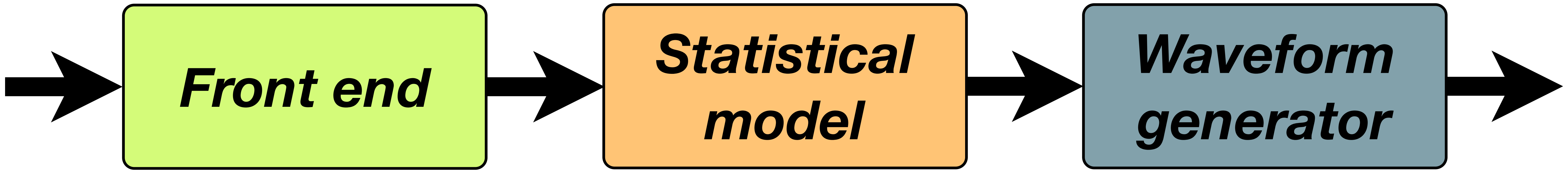


Synthesis with a neural network



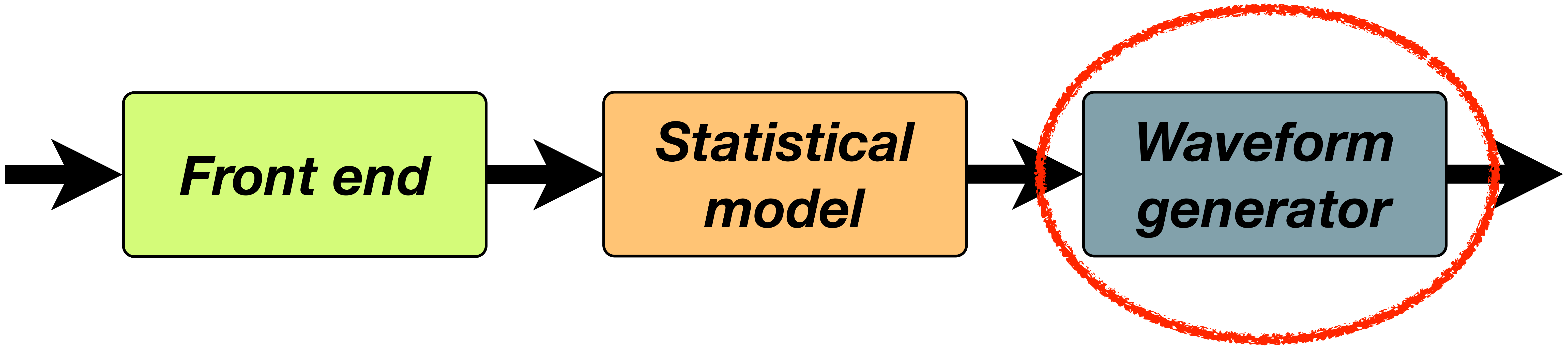
From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing



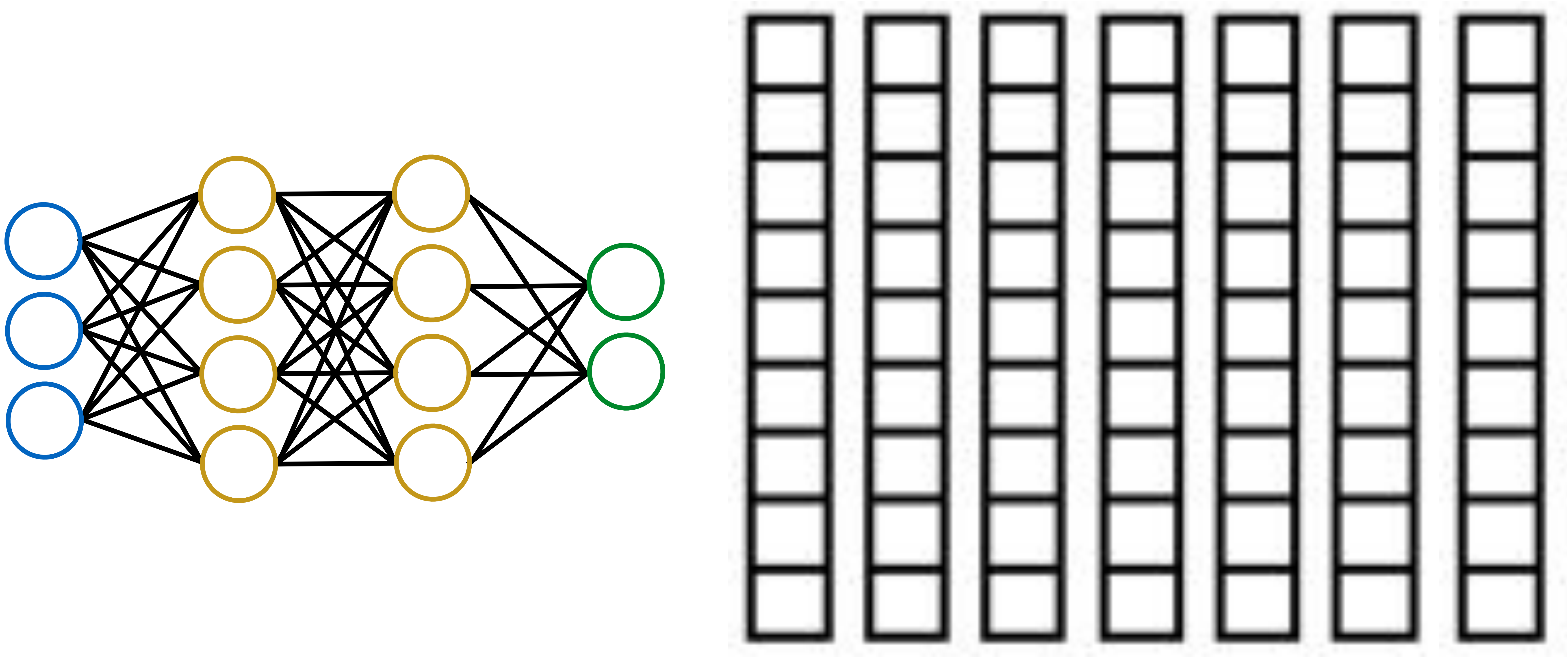
From text to speech

- Text processing
  - pipeline architecture
  - linguistic specification
- Regression
  - duration model
  - acoustic model
- Waveform generation
  - acoustic features
  - signal processing

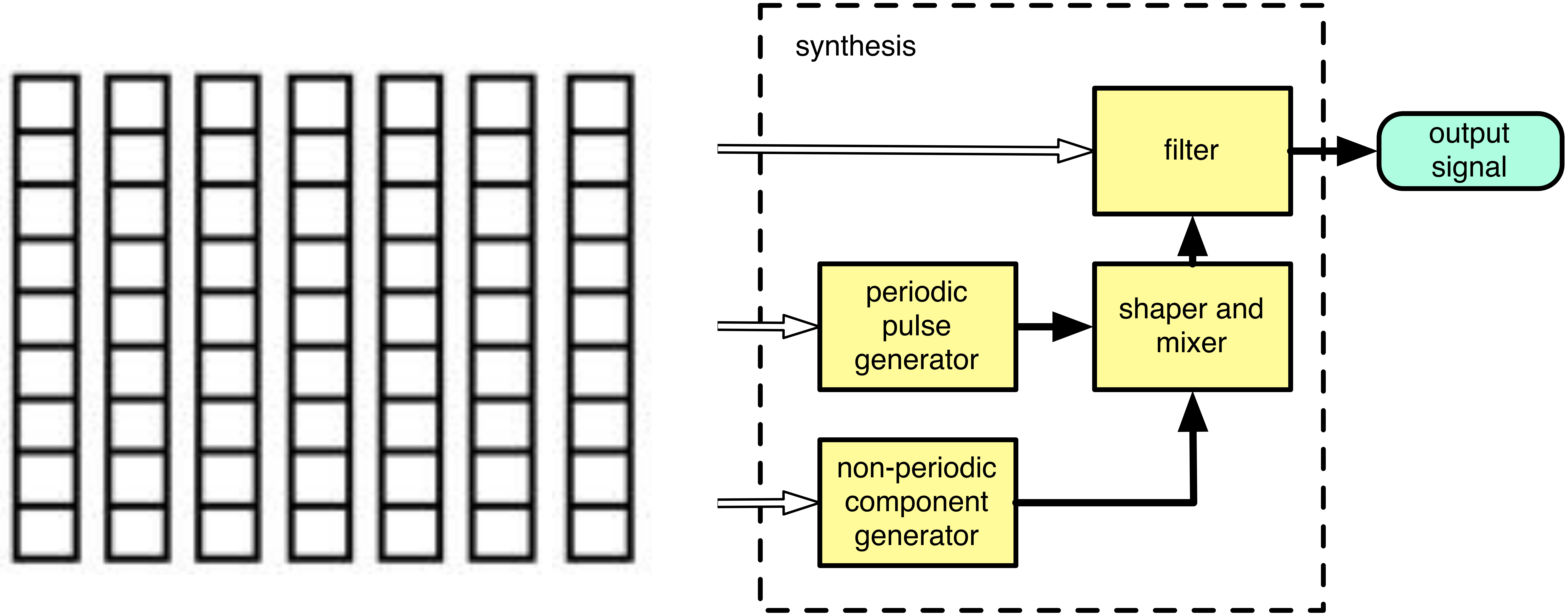


What are the acoustic features?

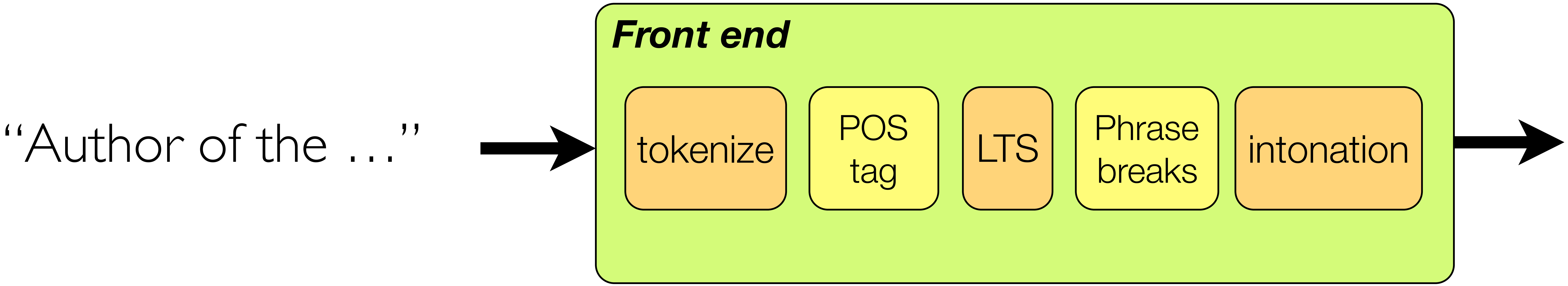
```
[0 0 1 0 0 1 0 1 1 0 ... 0.2 0.0]
[0 0 1 0 0 1 0 1 1 0 ... 0.2 0.1]
...
[0 0 1 0 0 1 0 1 1 0 ... 0.2 1.0]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.0]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.5]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 1.0]
...
[0 0 1 0 0 1 0 1 1 0 ... 1.0 1.0]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.0]
[0 0 0 1 1 1 0 1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



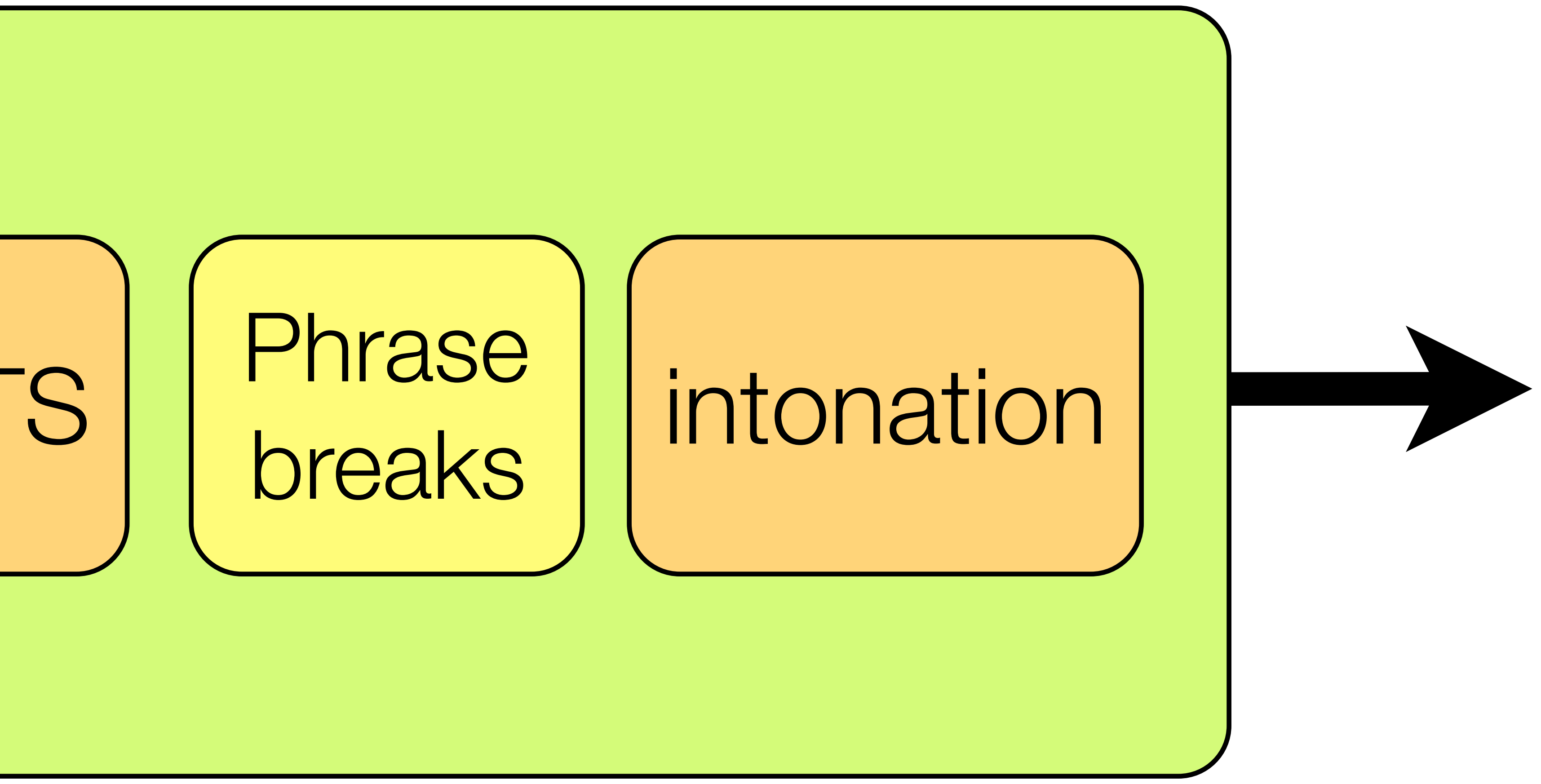
What are the acoustic features?



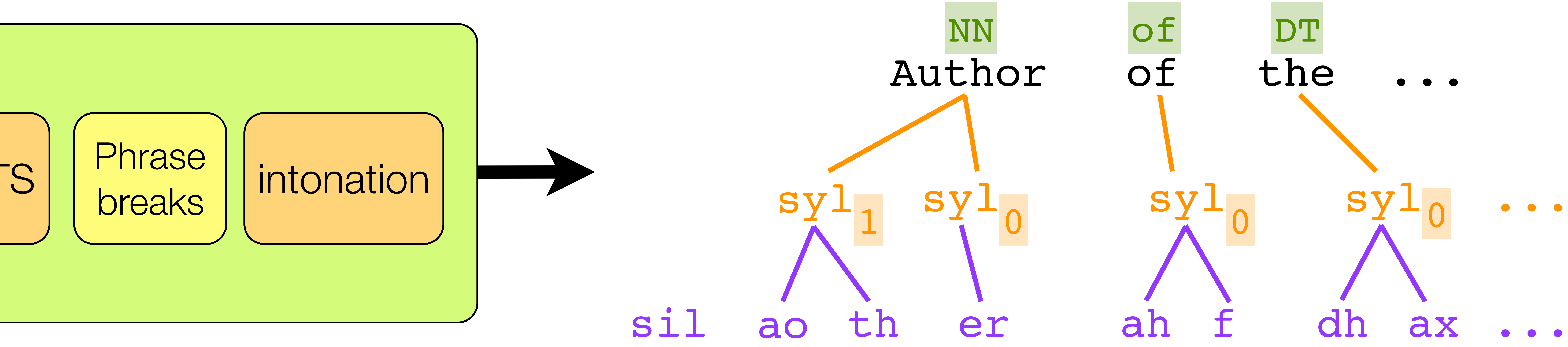
Putting it all together: text-to-speech with a neural network



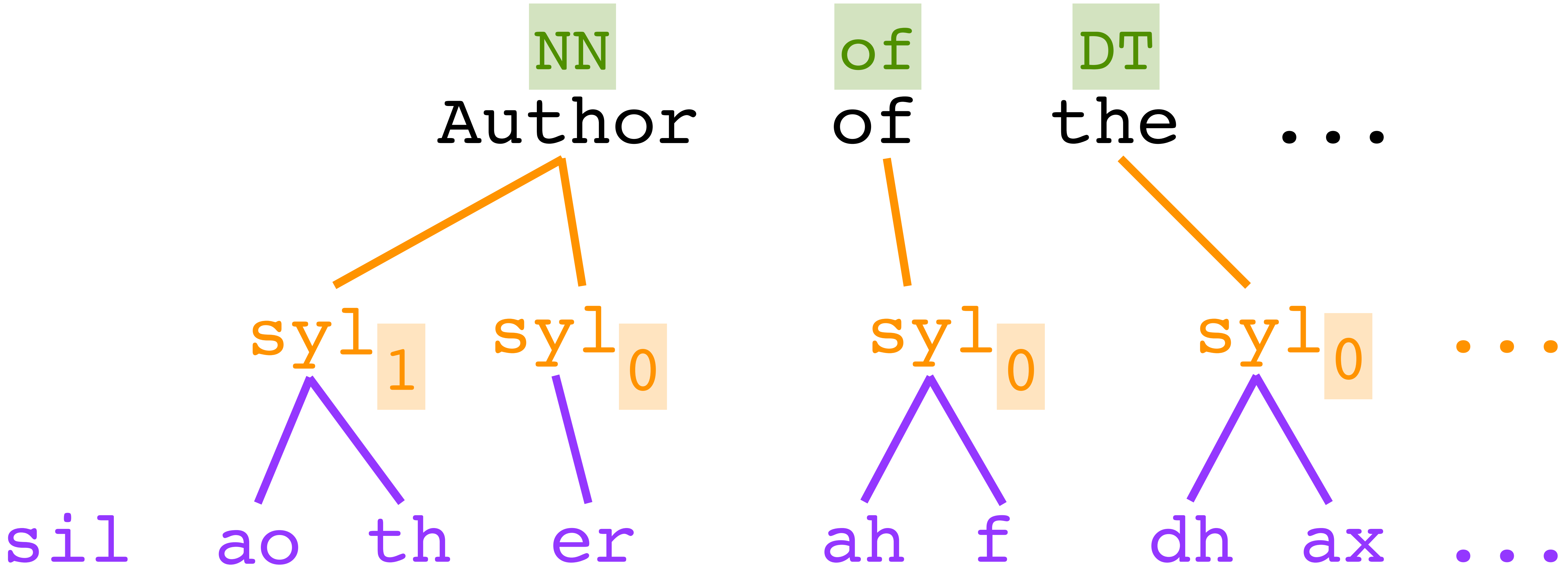
Putting it all together: text-to-speech with a neural network



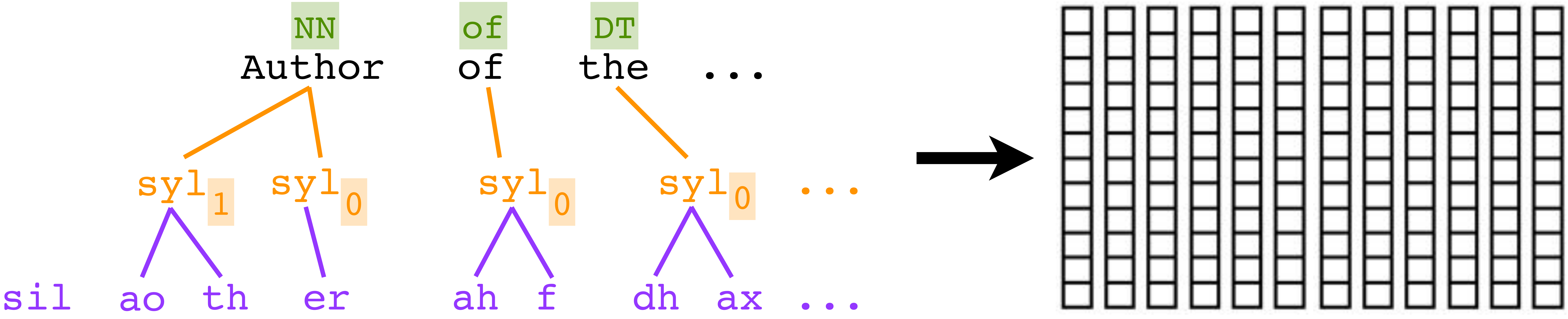
Putting it all together: text-to-speech with a neural network



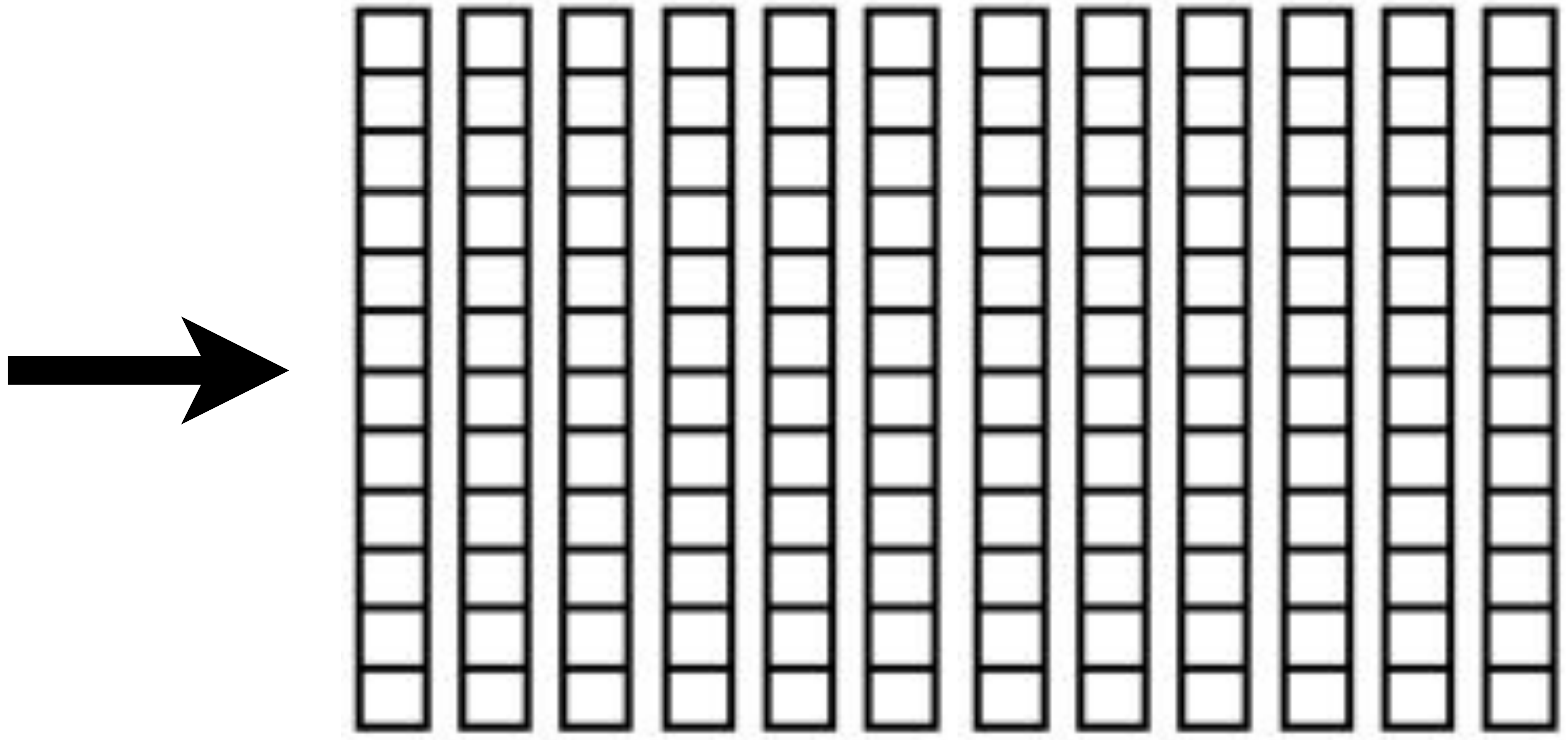
Putting it all together: text-to-speech with a neural network



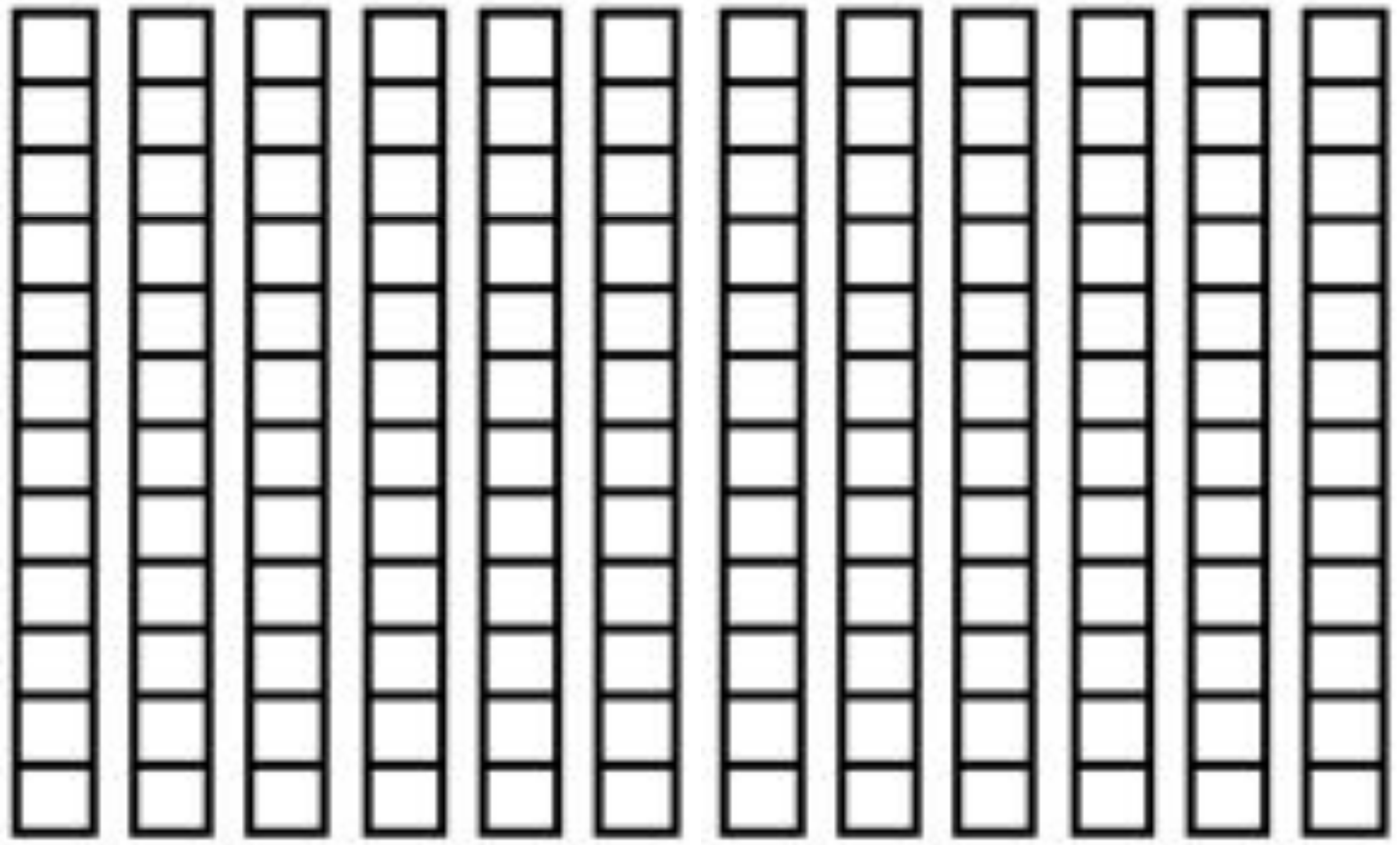
Putting it all together: text-to-speech with a neural network



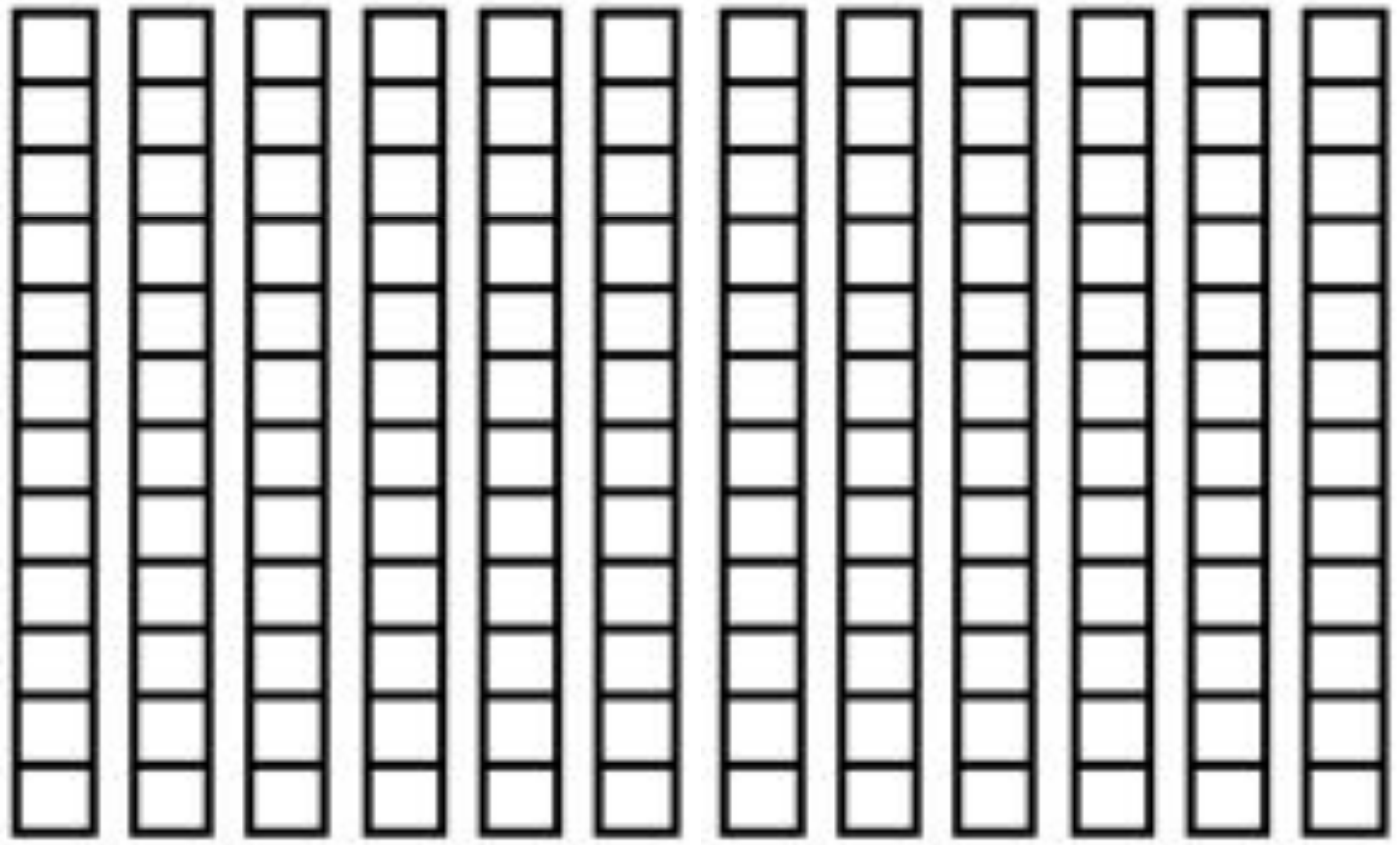
Putting it all together: text-to-speech with a neural network



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



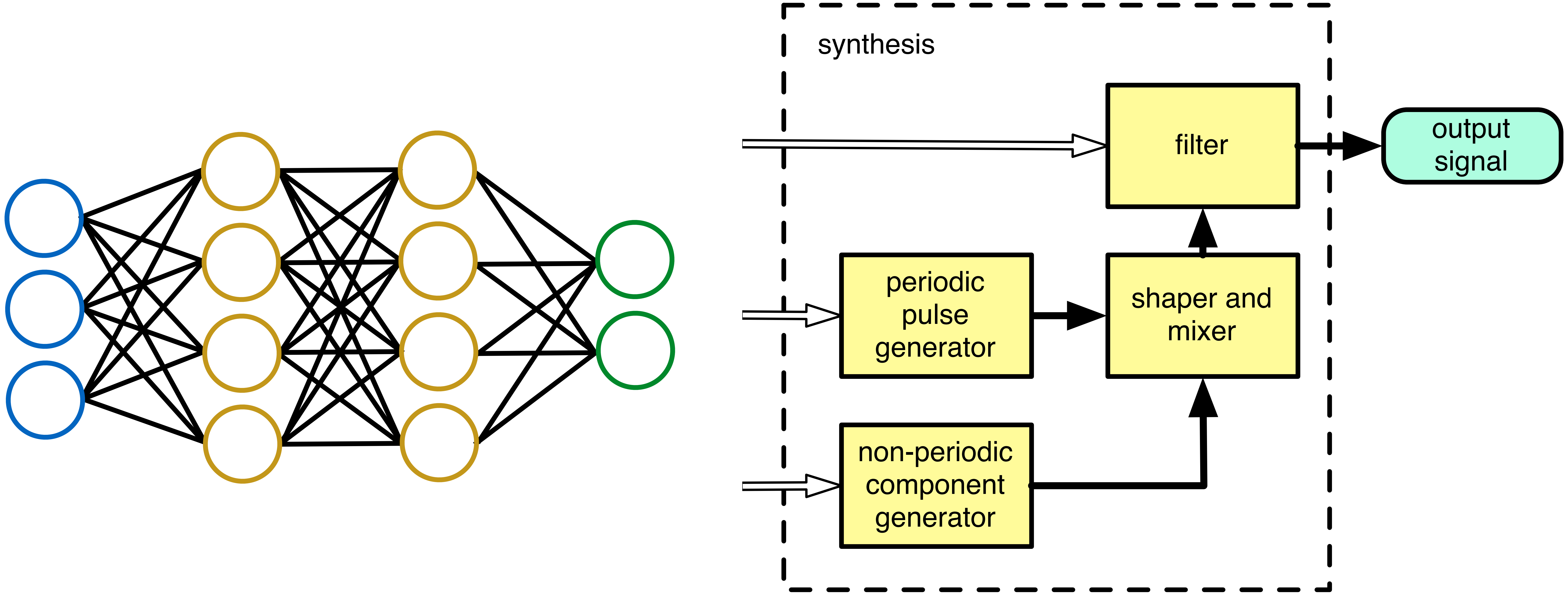
Putting it all together: text-to-speech with a neural network



```
[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]
…
```

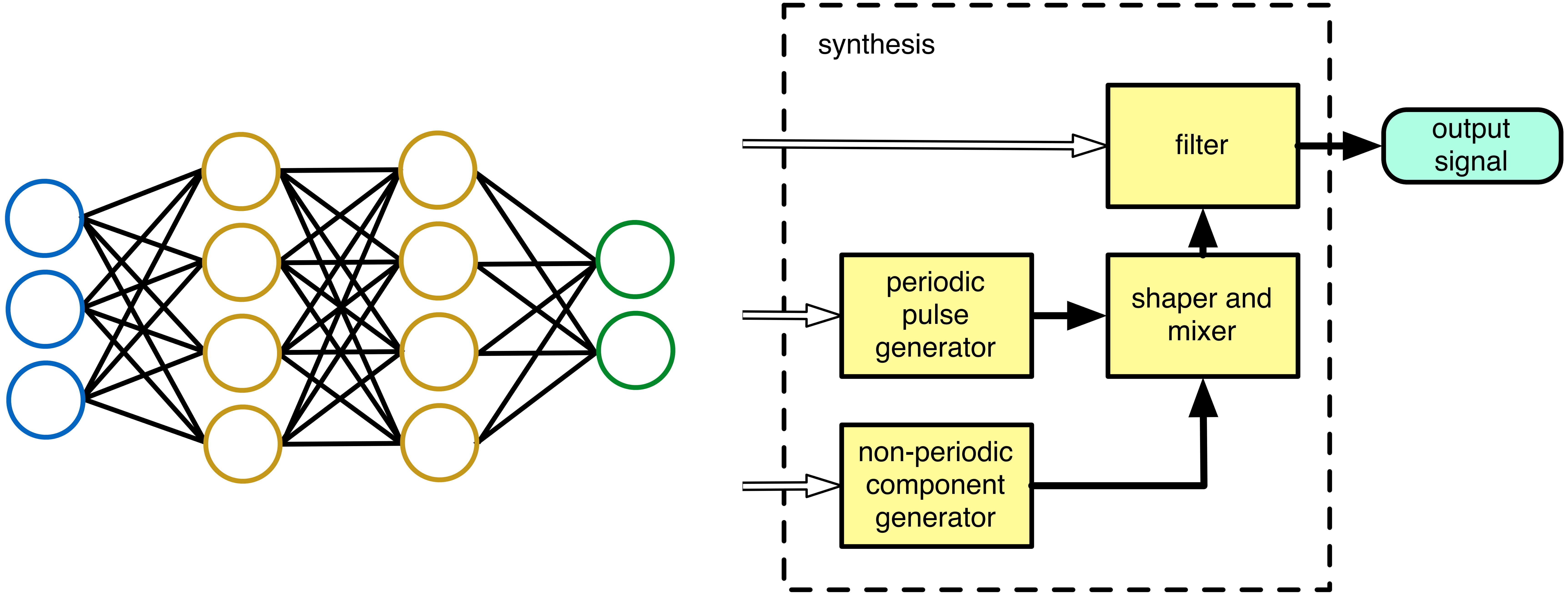
Putting it all together: text-to-speech with a neural network

```
[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]
...
```



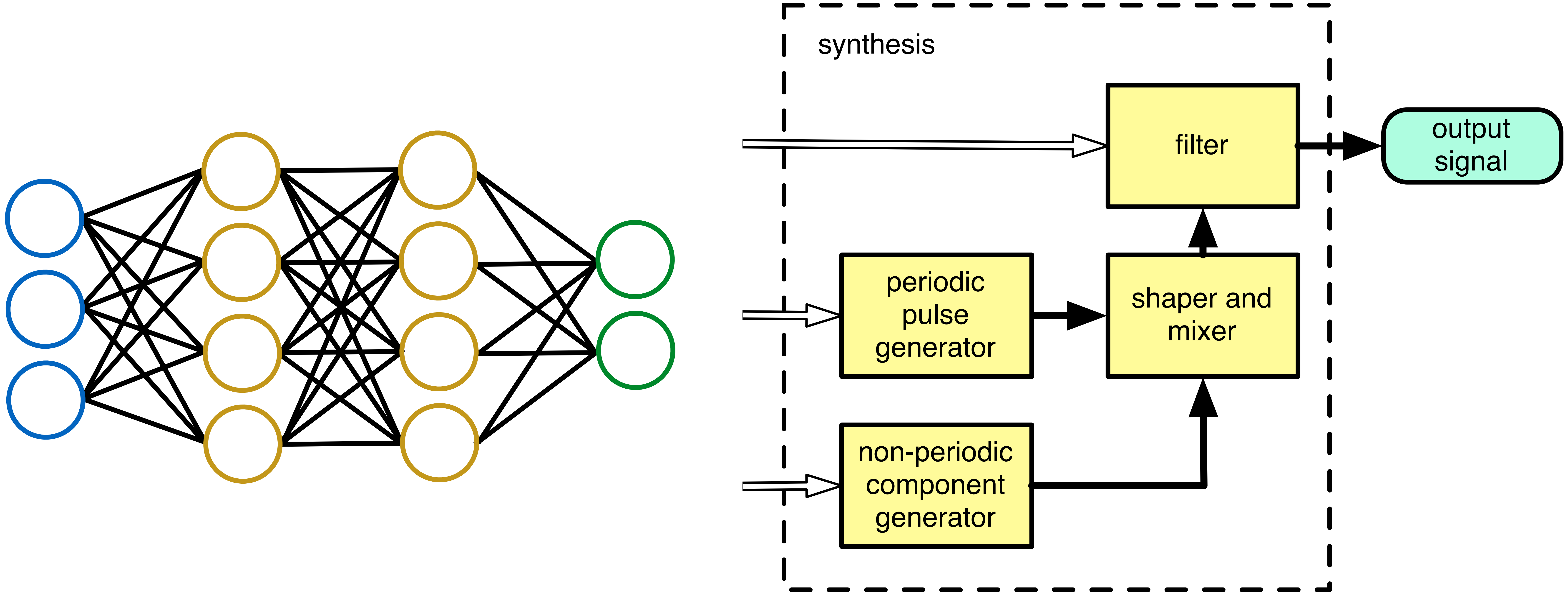
Putting it all together: text-to-speech with a neural network

```
[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]
…
```



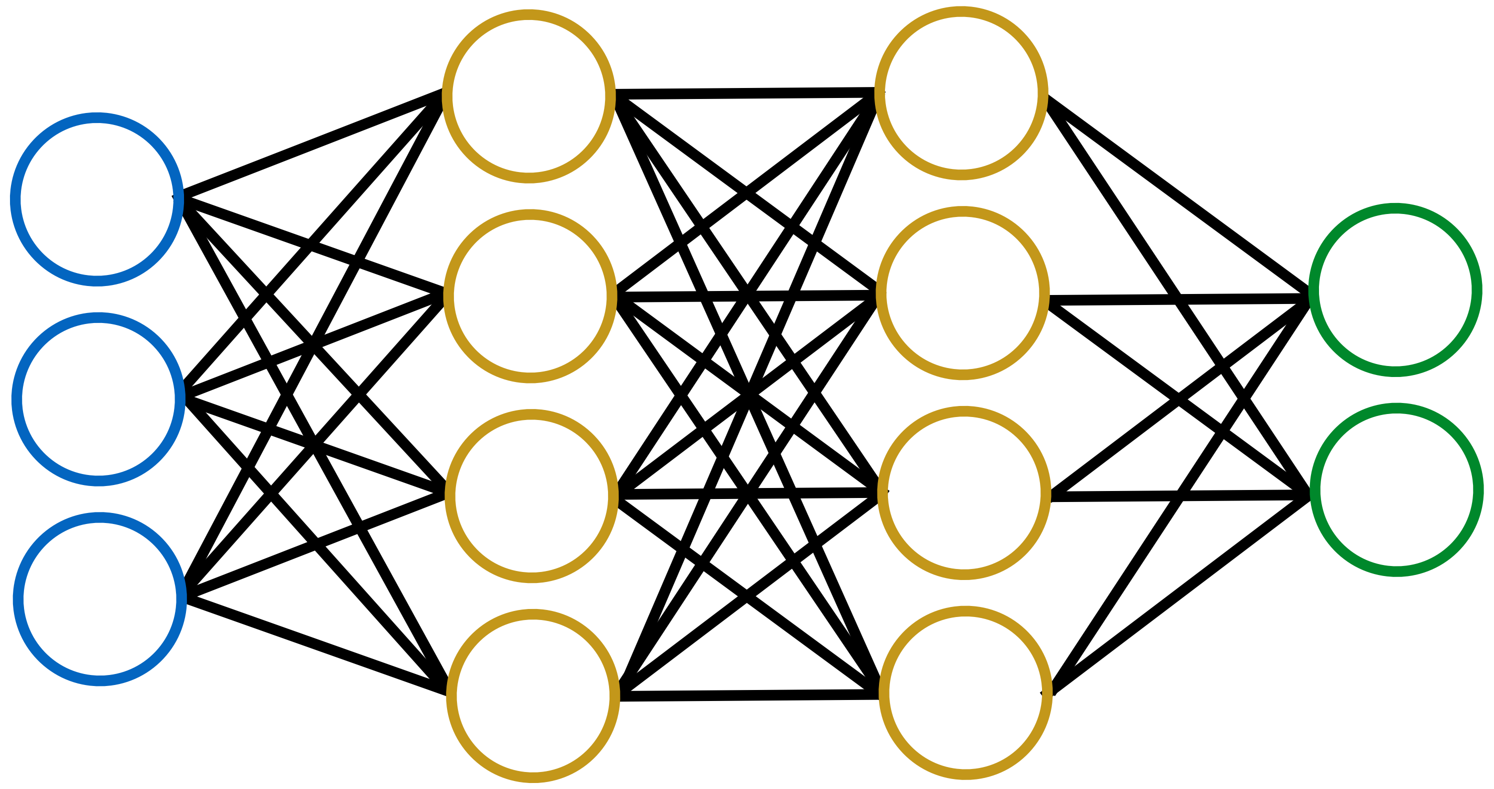
Putting it all together: text-to-speech with a neural network

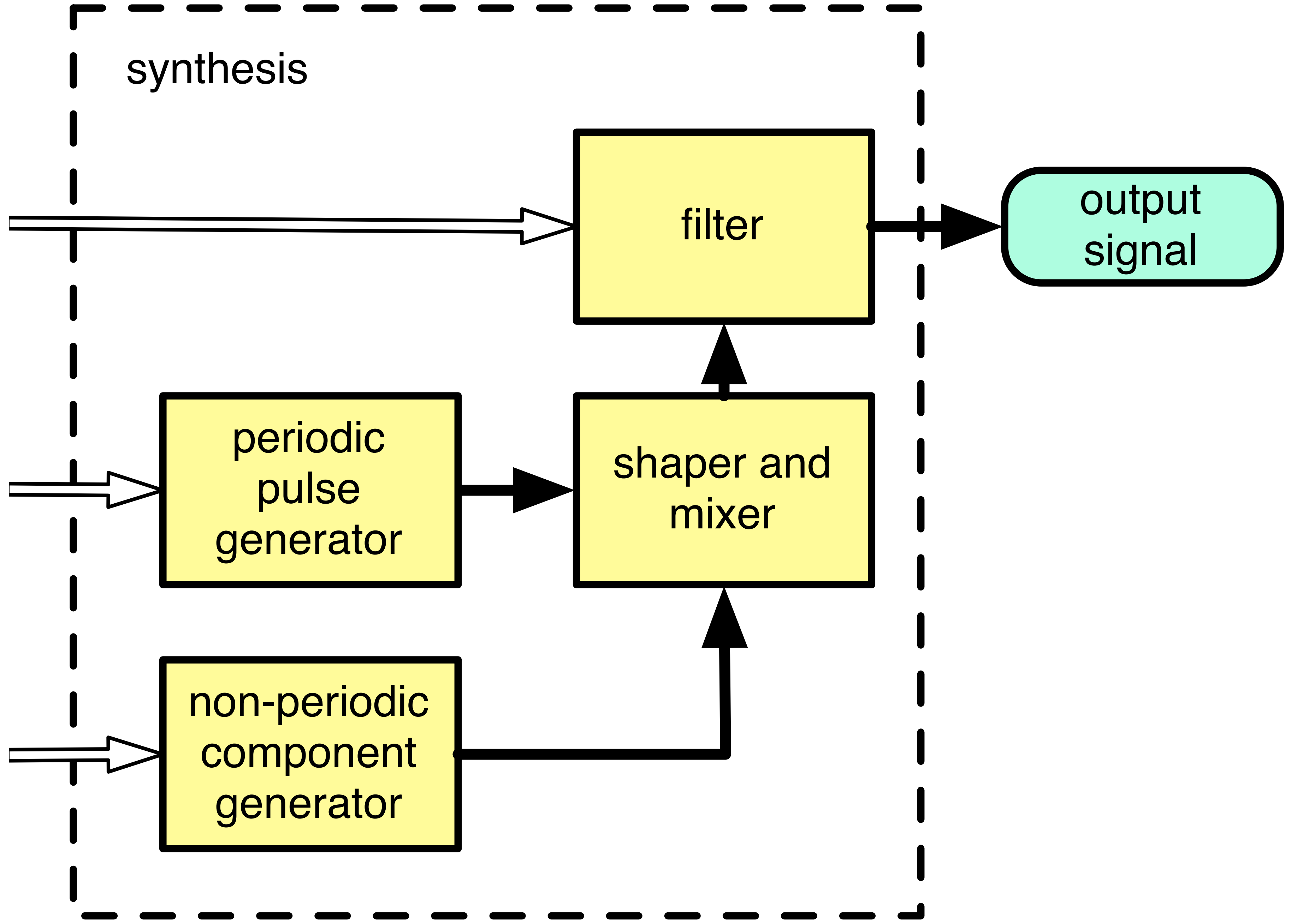
```
[0 0 1 0 0 1 0 1 1 0 ... 0.2 0.1]
...
[0 0 1 0 0 1 0 1 1 0 ... 0.2 1.0]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.0]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 0.5]
[0 0 1 0 0 1 0 1 1 0 ... 0.4 1.0]
...
[0 0 1 0 0 1 0 1 1 0 ... 1.0 1.0]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.0]
[0 0 0 1 1 1 0 1 0 0 ... 0.2 0.2]
[0 0 0 1 1 1 0 1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



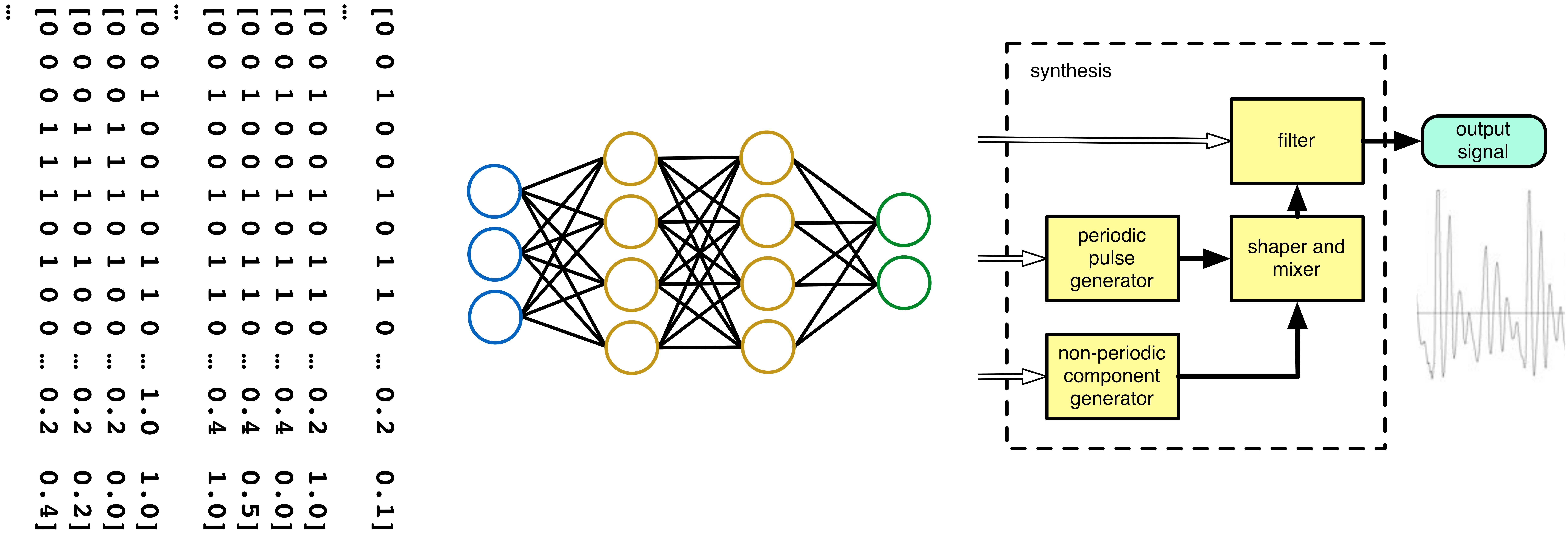
Putting it all together: text-to-speech with a neural network

```
[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



```
[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]
...
```

What next?

- How to build the system
  - A front end for a new language
  - Linguistic feature extraction & engineering
  - Acoustic feature extraction & engineering
  - Regression
    - including duration modelling
  - Waveform generation

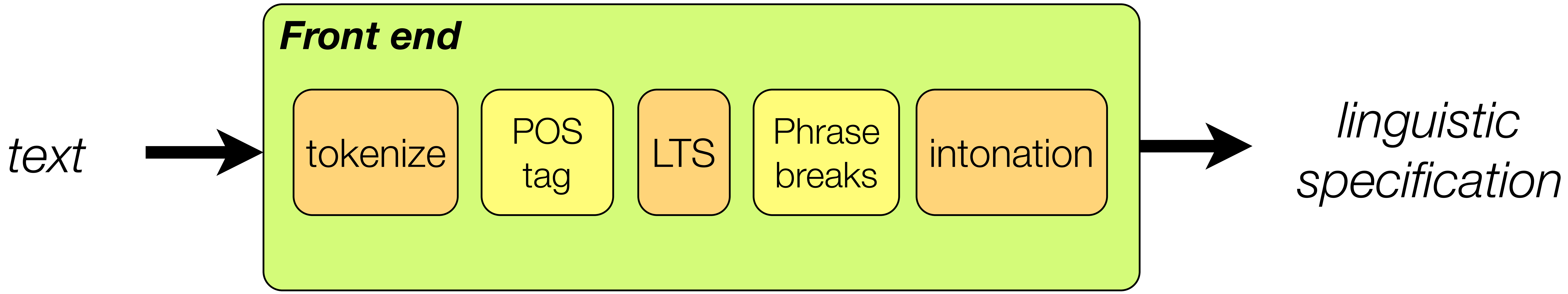
Agenda

	Topic	Presenter
PART 1	From text to speech	Simon King
PART 2	**The front end**	**Oliver Watts**
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

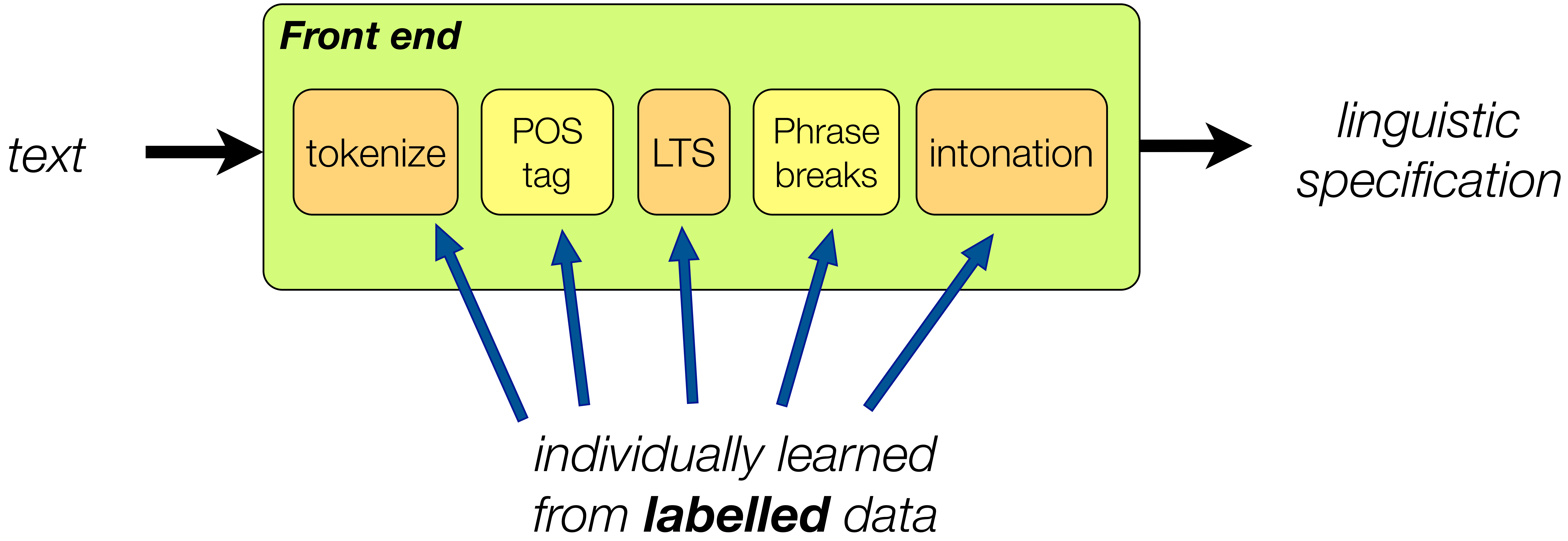
The front end

Oliver Watts

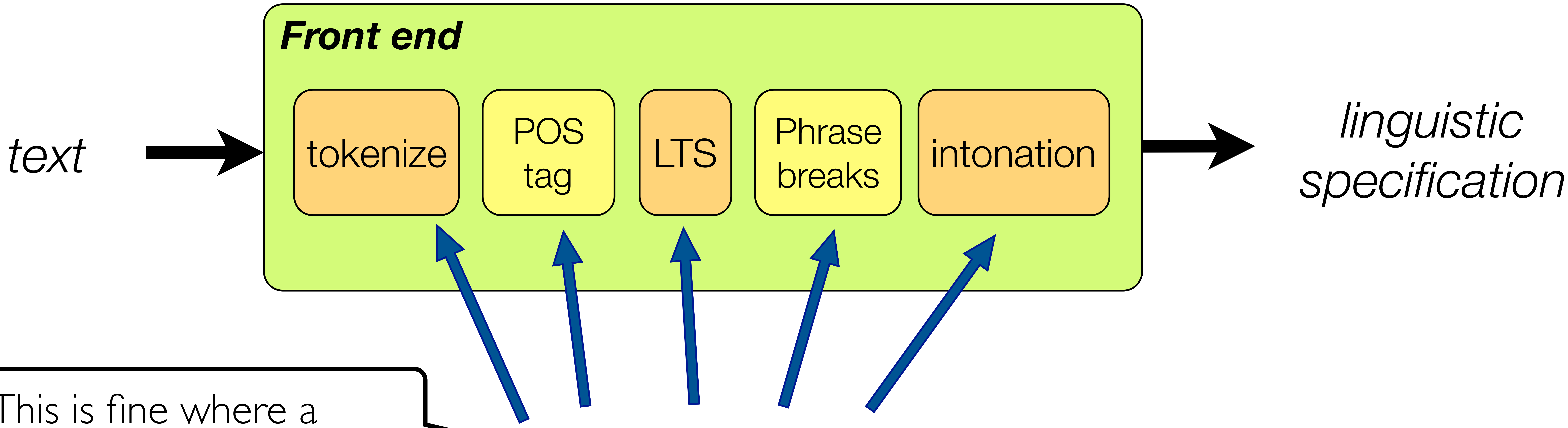
Front end


text
Front end
tokenize
POS tag
LTS
Phrase breaks
intonation
linguistic specification

Front end


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

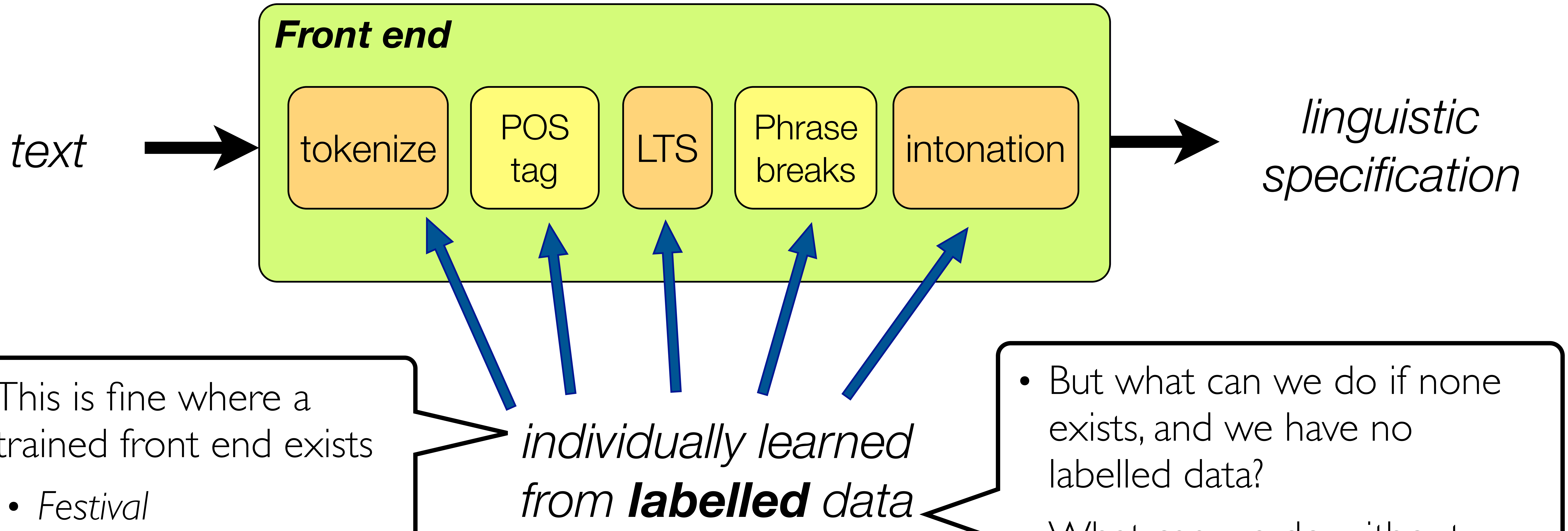
Front end



individually learned from ***labelled*** *data*

- This is fine where a trained front end exists
  - *Festival*
  - *MaryTTS*
  - *eSpeak*

Front end



- This is fine where a trained front end exists
  - *Festival*
  - *MaryTTS*
  - *eSpeak*
  -

- But what can we do if none exists, and we have no labelled data?
- What can we do without labelled data?
  - *Ossian*

simple4all.org

Ossian toolkit

- uses **training data**, which can be as minimal as speech + text
  - sentence- *or* paragraph-aligned

- exploits any **additional resources** a user can find

- provides **front-end modules** and the ‘**glue**’ for combining them with Merlin DNNs

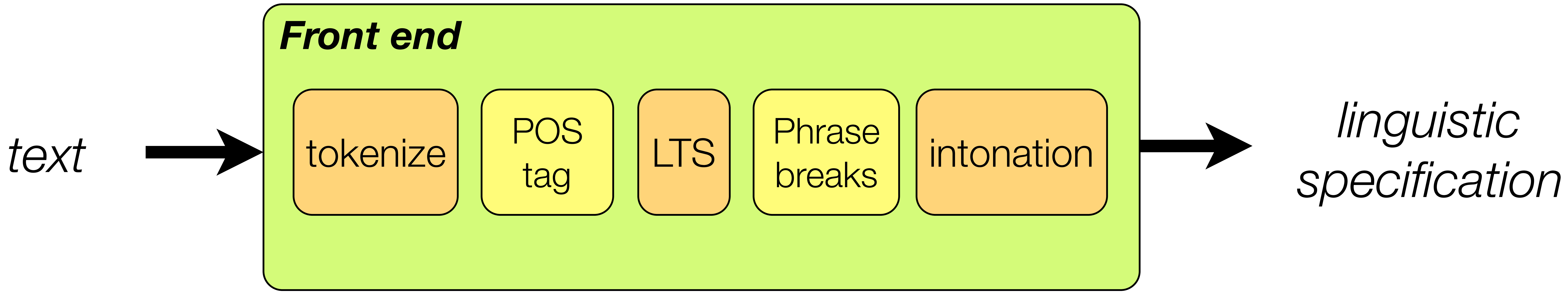
Home \ Outputs \ Tools \ Ossian

Ossian

Ossian is a python based tool for automatic
ends. The framework consists of a sequenc
accepts and enriches an XML representatio
designed to be flexible so that the nature a
can be configured by a user. A range of pro
it is also relatively easy to code new ones.

[Download]

Ossian toolkit



- In this section we will:
  - Show how Ossian can be used with Merlin to build a **Swahili** voice without any language-specific expertise, only transcribed speech
  - Introduce some of the ideas used by Ossian to manage **without annotation**

Ossian naive recipe: training data

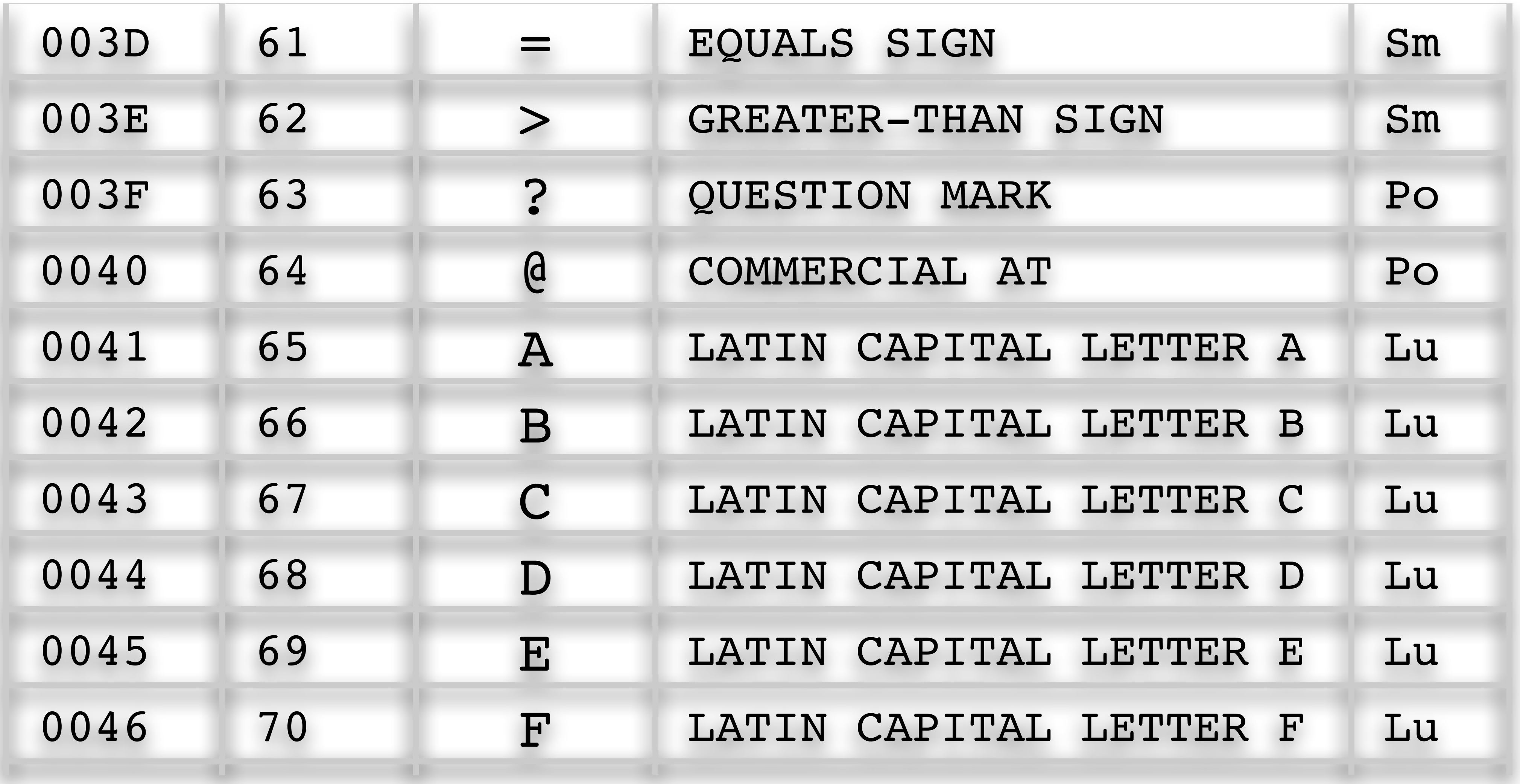
```
Khartoum imejitenga na mzozo huo.
```

The only required input is UTF8 text and speech, in matched sentence/paragraph size chunks

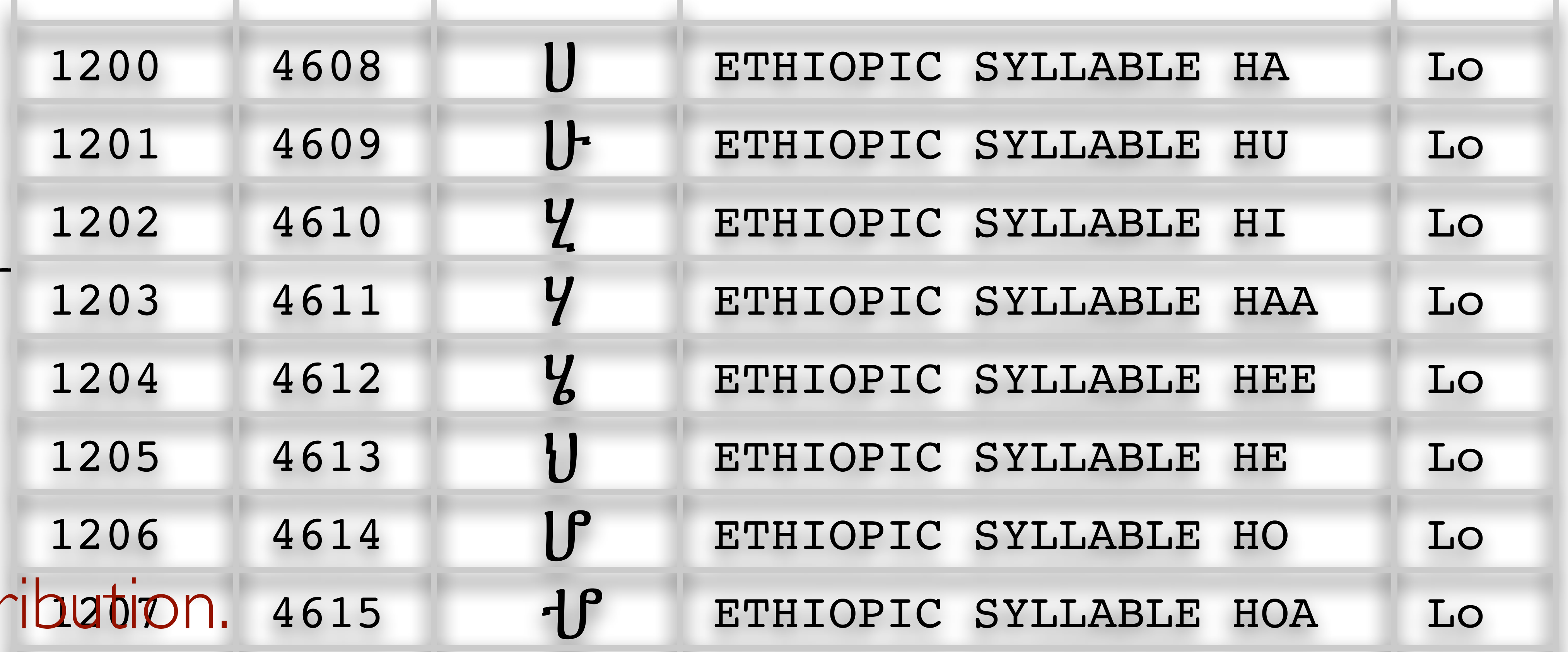
Ossian naive recipe: training data

Khartoum imejitenga na mzozo huo.

The only required input is UTF8 text and speech, in matched sentence/paragraph size chunks



003D	61	=	EQUALS SIGN	Sm
003E	62	>	GREATER-THAN SIGN	Sm
003F	63	?	QUESTION MARK	Po
0040	64	@	COMMERCIAL AT	Po
0041	65	A	LATIN CAPITAL LETTER A	Lu
0042	66	B	LATIN CAPITAL LETTER B	Lu
0043	67	C	LATIN CAPITAL LETTER C	Lu
0044	68	D	LATIN CAPITAL LETTER D	Lu
0045	69	E	LATIN CAPITAL LETTER E	Lu
0046	70	F	LATIN CAPITAL LETTER F	Lu



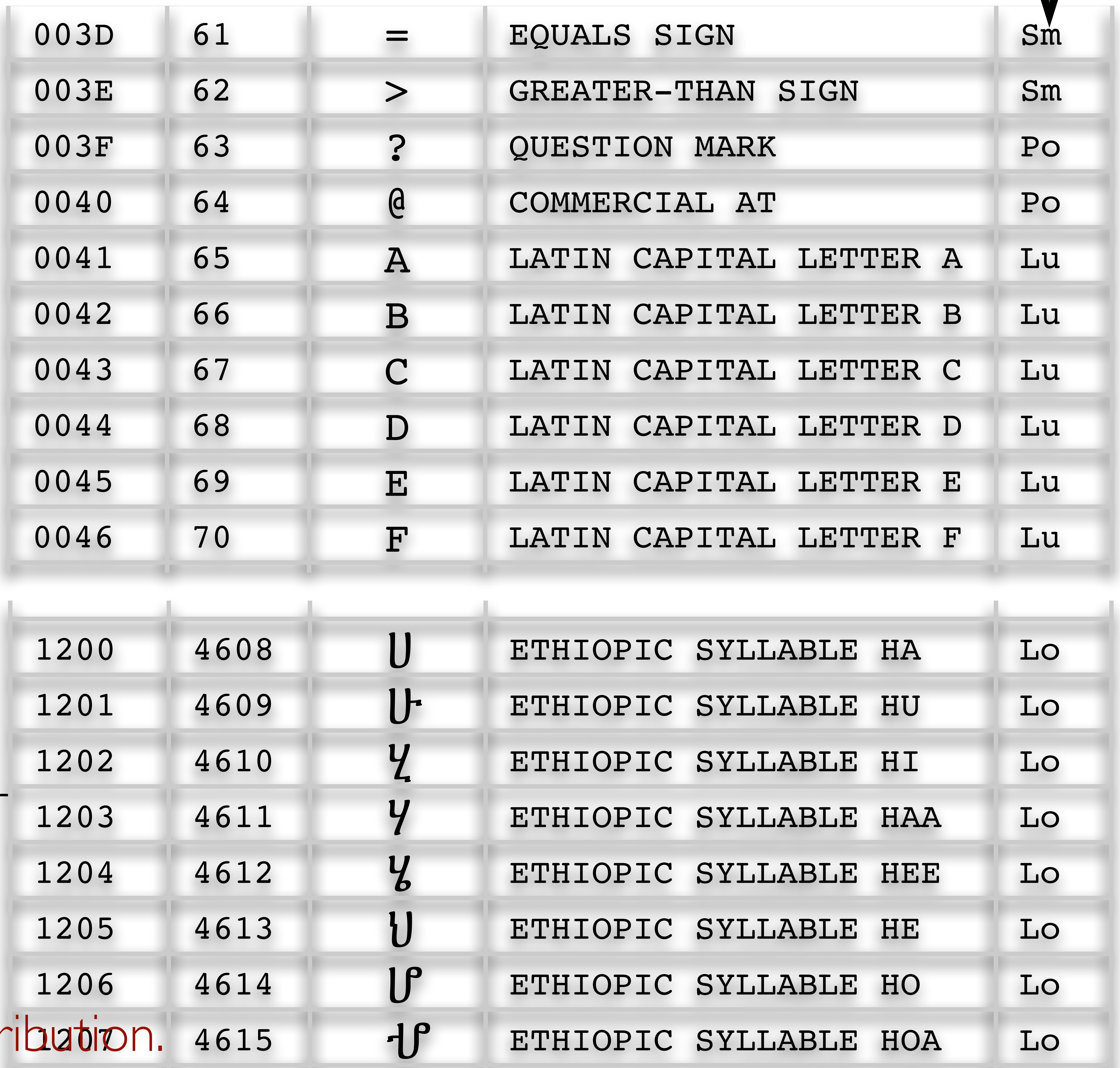
1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
1202	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
1203	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
1204	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
1205	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo

Ossian naive recipe: training data

Khartoum imejitenga na mzozo huo.

The only required input is UTF8 text and speech, in matched sentence/paragraph size chunks

character classes



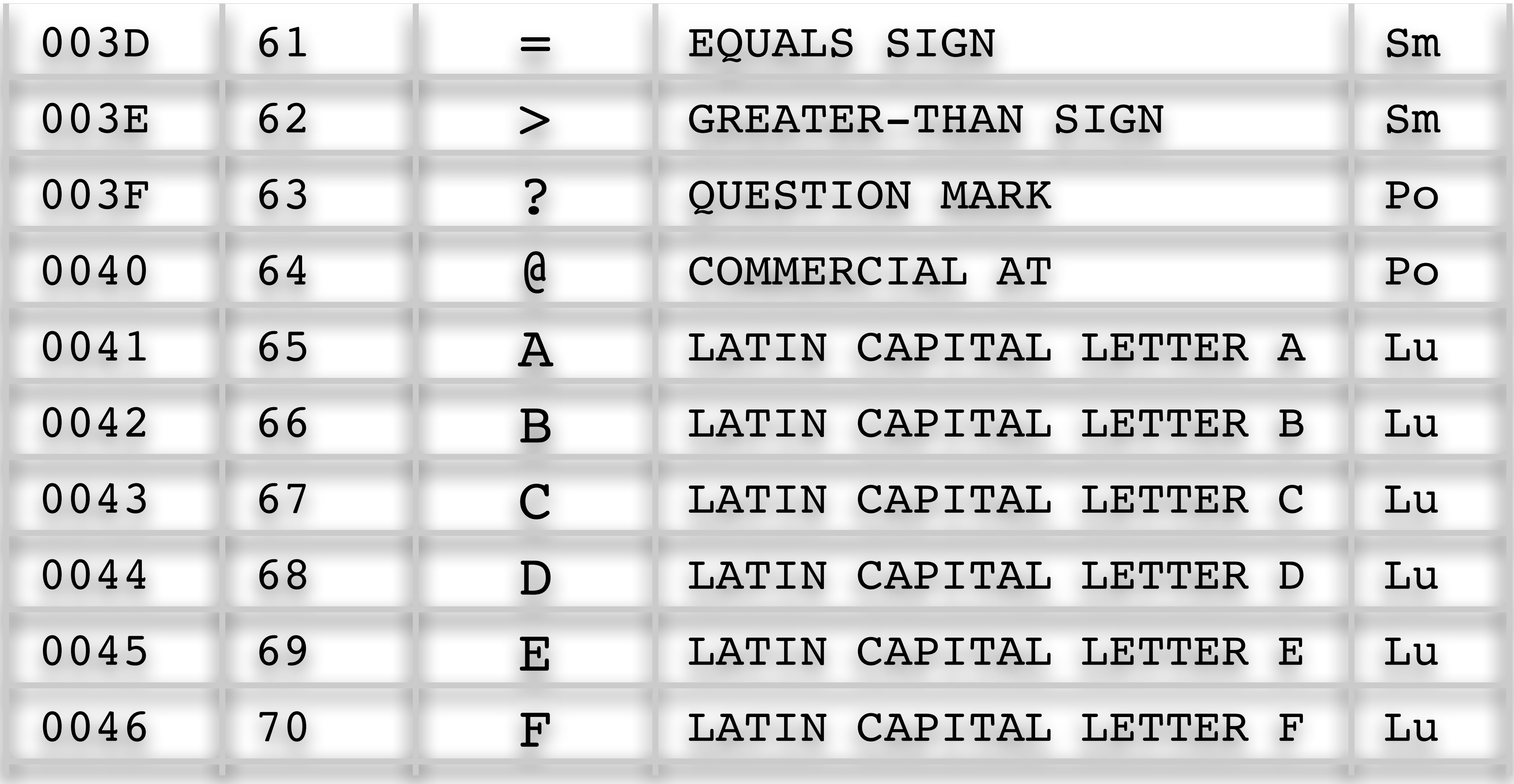
003D	61	=	EQUALS SIGN	Sm
003E	62	>	GREATER-THAN SIGN	Sm
003F	63	?	QUESTION MARK	Po
0040	64	@	COMMERCIAL AT	Po
0041	65	A	LATIN CAPITAL LETTER A	Lu
0042	66	B	LATIN CAPITAL LETTER B	Lu
0043	67	C	LATIN CAPITAL LETTER C	Lu
0044	68	D	LATIN CAPITAL LETTER D	Lu
0045	69	E	LATIN CAPITAL LETTER E	Lu
0046	70	F	LATIN CAPITAL LETTER F	Lu

1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
1202	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
1203	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
1204	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
1205	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo

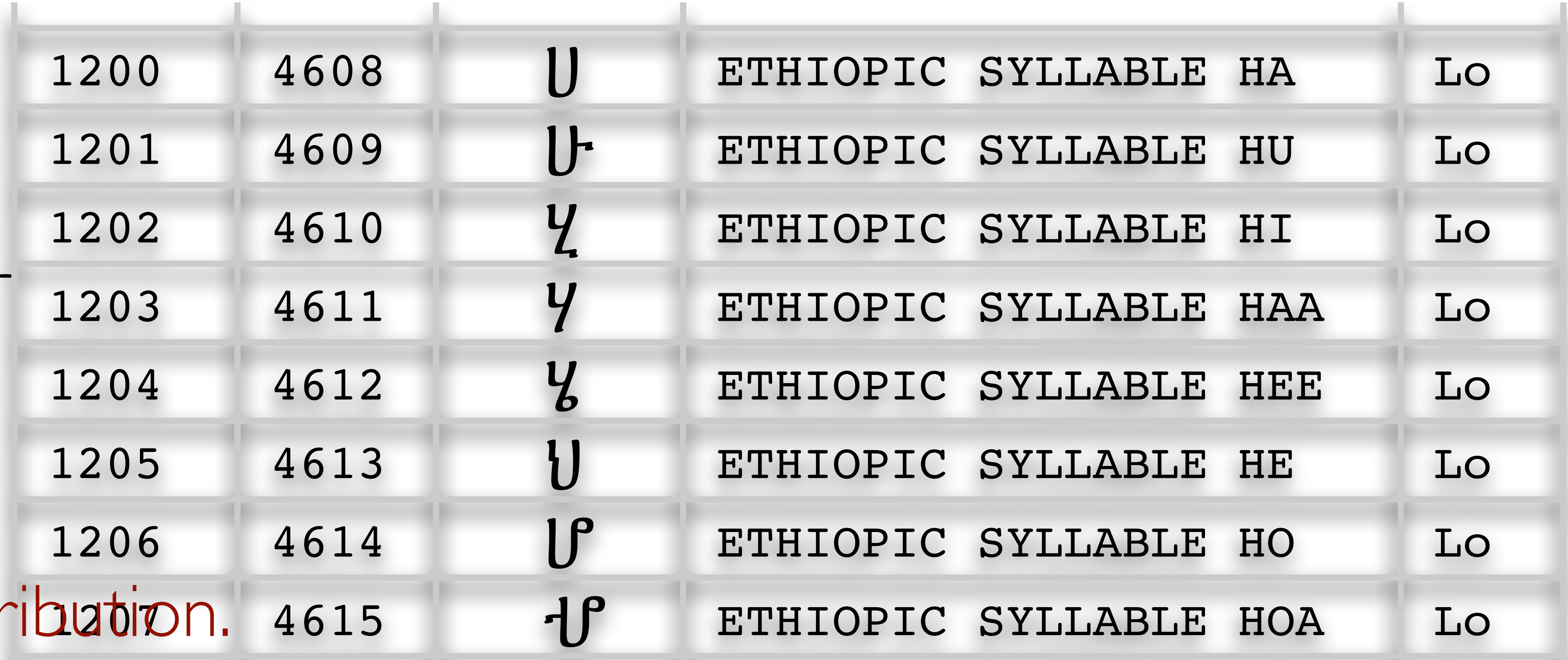
Ossian naive recipe: training data

Khartoum imejitenga na mzozo huo.

The only required input is UTF8 text and speech, in matched sentence/paragraph size chunks



003D	61	=	EQUALS SIGN	Sm
003E	62	>	GREATER-THAN SIGN	Sm
003F	63	?	QUESTION MARK	Po
0040	64	@	COMMERCIAL AT	Po
0041	65	A	LATIN CAPITAL LETTER A	Lu
0042	66	B	LATIN CAPITAL LETTER B	Lu
0043	67	C	LATIN CAPITAL LETTER C	Lu
0044	68	D	LATIN CAPITAL LETTER D	Lu
0045	69	E	LATIN CAPITAL LETTER E	Lu
0046	70	F	LATIN CAPITAL LETTER F	Lu



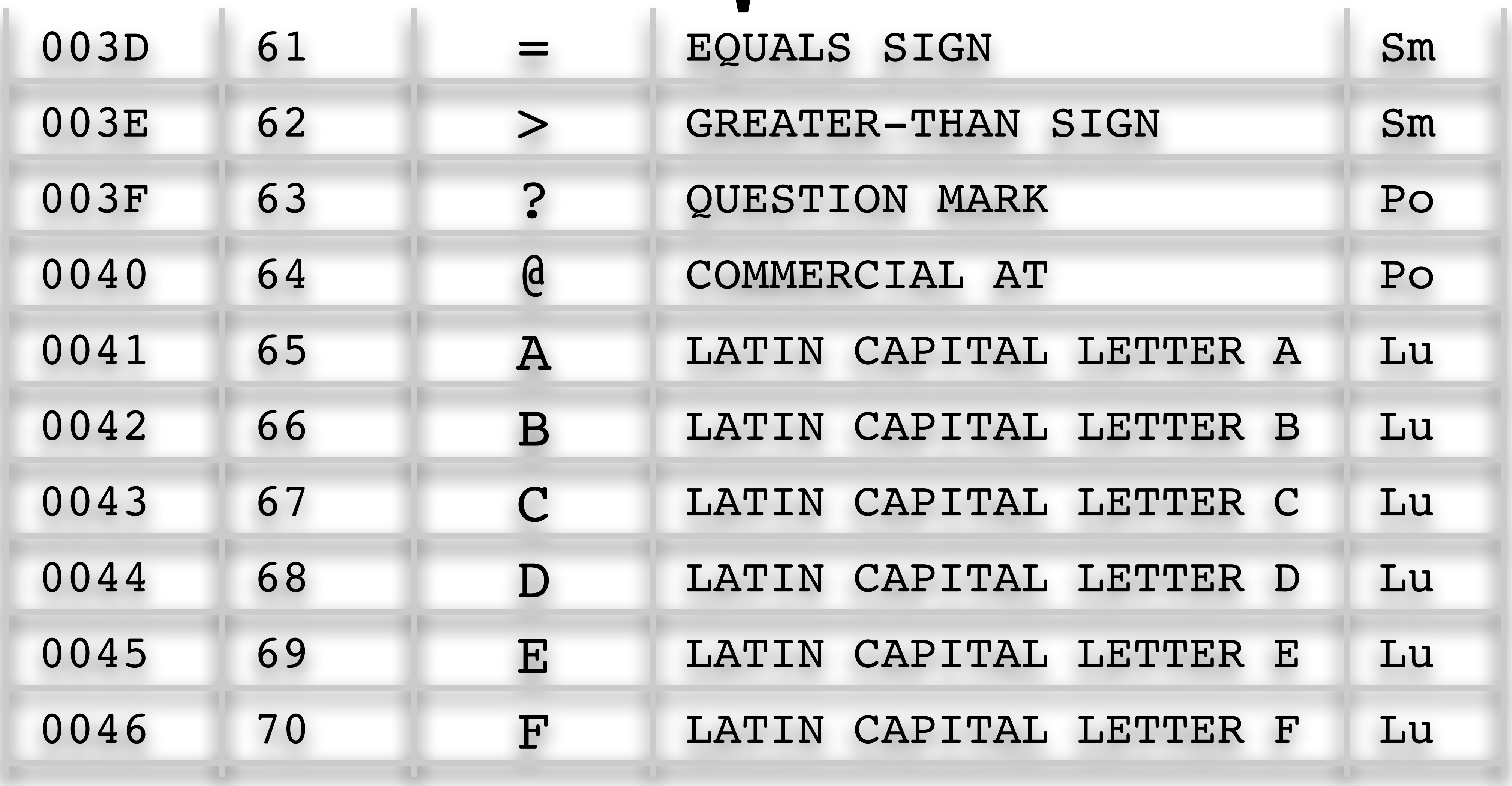
1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
1202	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
1203	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
1204	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
1205	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo

Ossian naive recipe: training data

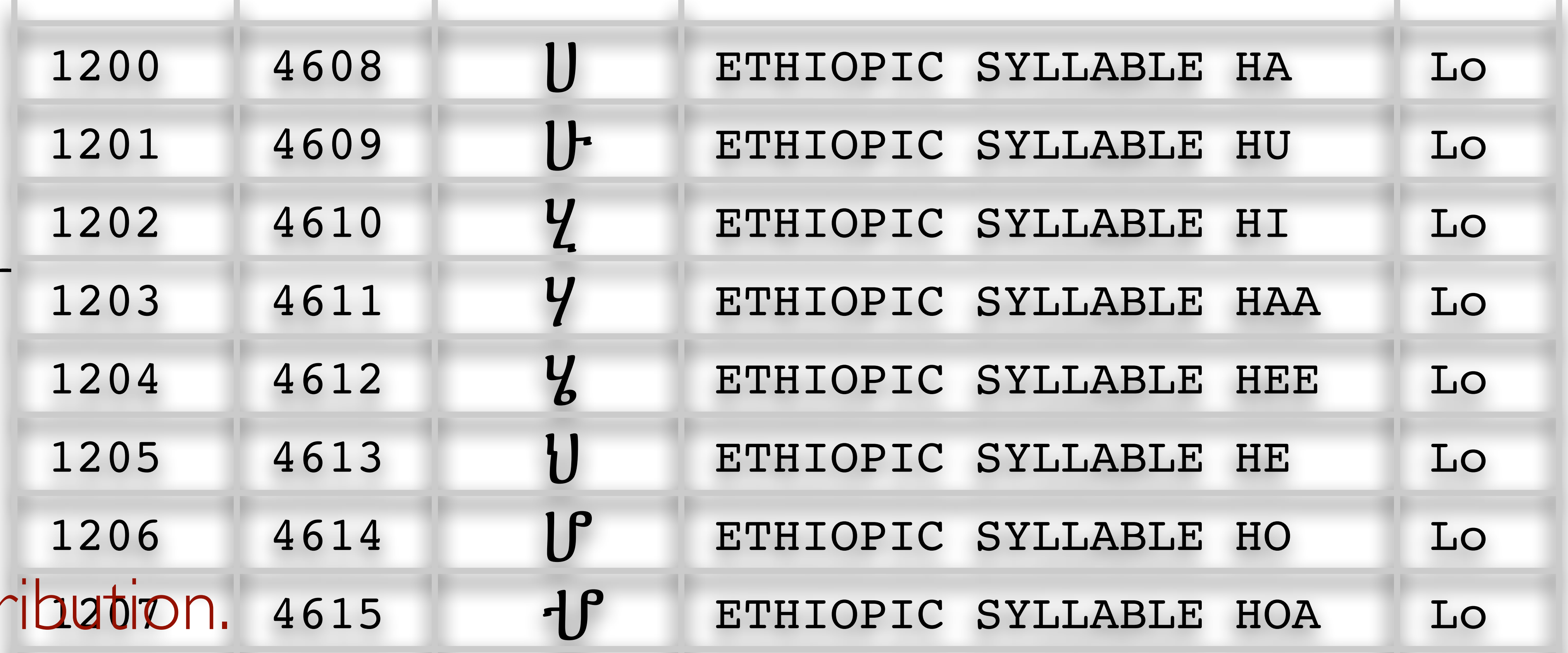
ASCII names

```
Khartoum imejitenga na mzozo huo.
```

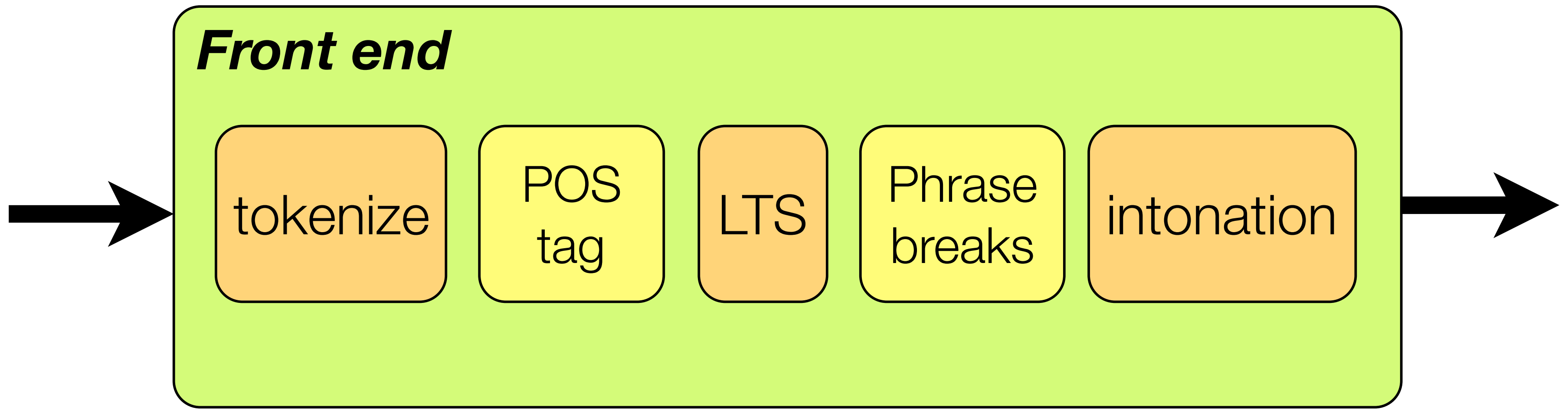
The only required input is UTF8 text and speech, in matched sentence/paragraph size chunks



003D	61	=	EQUALS SIGN	Sm
003E	62	>	GREATER-THAN SIGN	Sm
003F	63	?	QUESTION MARK	Po
0040	64	@	COMMERCIAL AT	Po
0041	65	A	LATIN CAPITAL LETTER A	Lu
0042	66	B	LATIN CAPITAL LETTER B	Lu
0043	67	C	LATIN CAPITAL LETTER C	Lu
0044	68	D	LATIN CAPITAL LETTER D	Lu
0045	69	E	LATIN CAPITAL LETTER E	Lu
0046	70	F	LATIN CAPITAL LETTER F	Lu



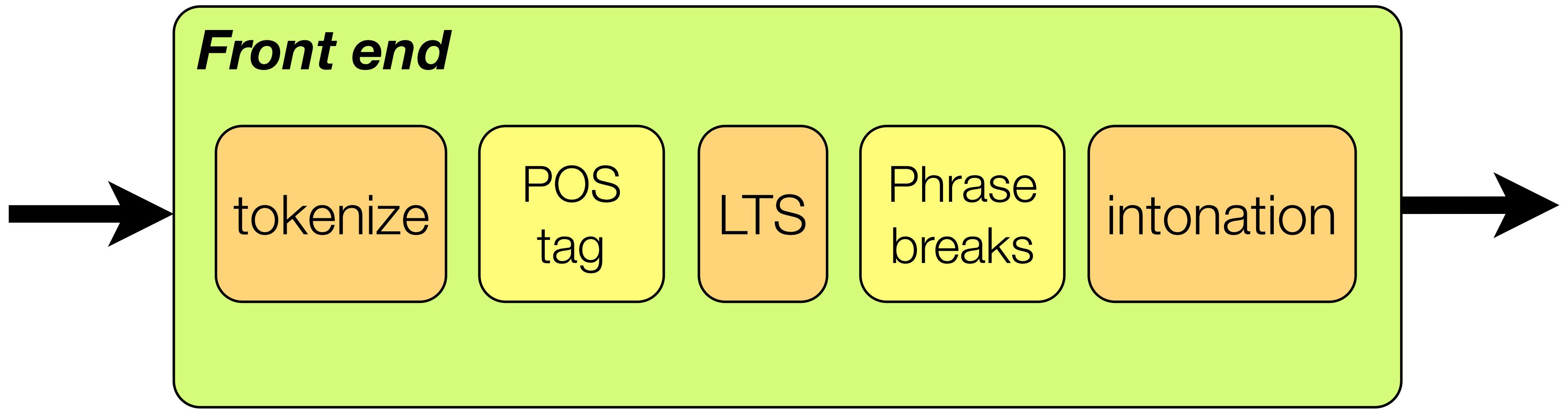
1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
1202	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
1203	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
1204	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
1205	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo



The front end

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"/>
```

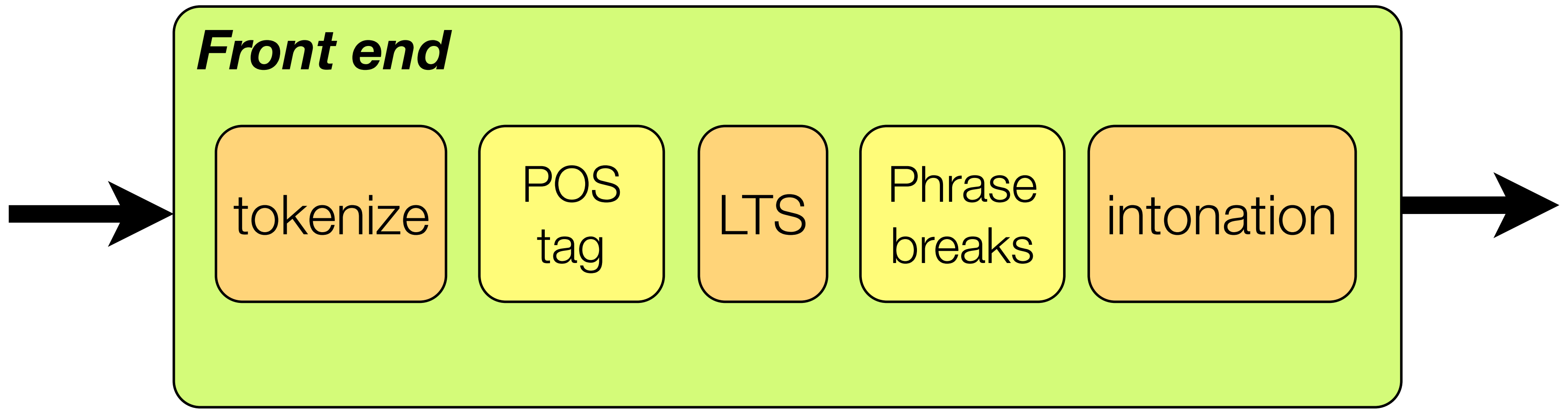
The front end



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"/>
```

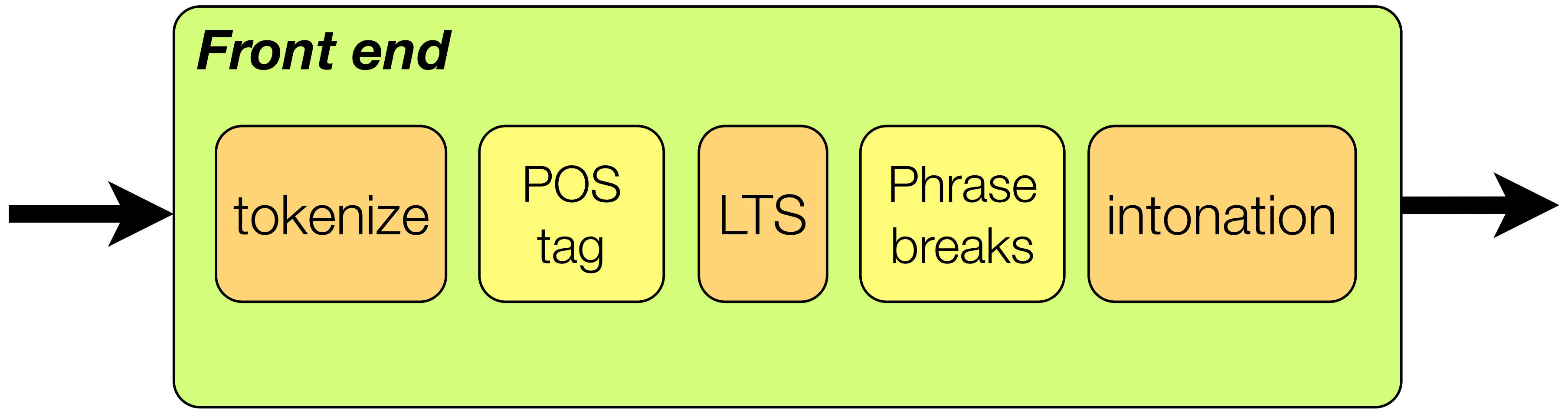
An XML utterance structure is created for each sentence in the training corpus

The front end

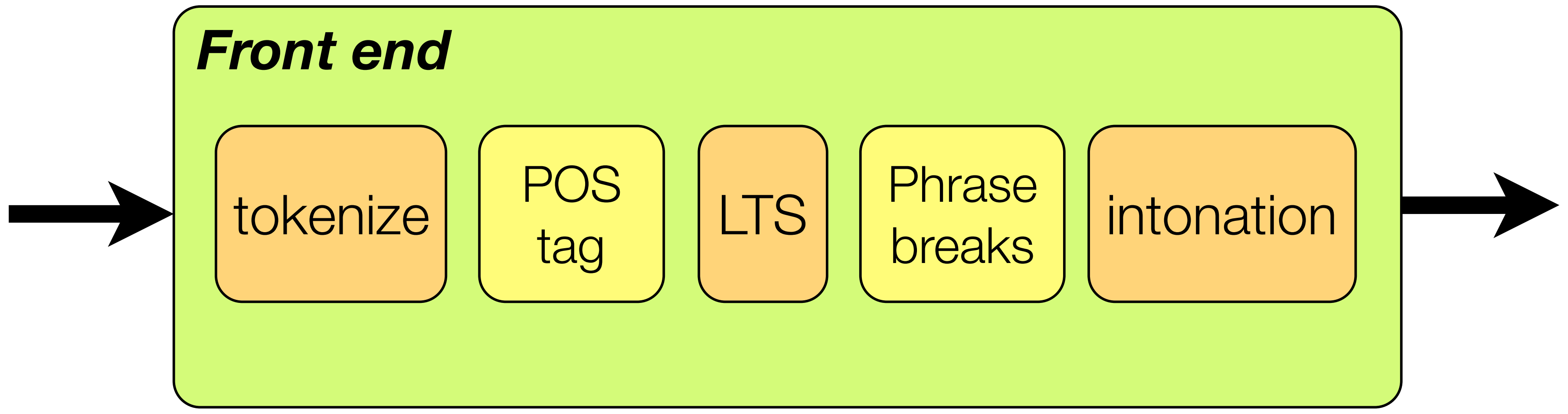


```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"/>
```

Tokeniser



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```



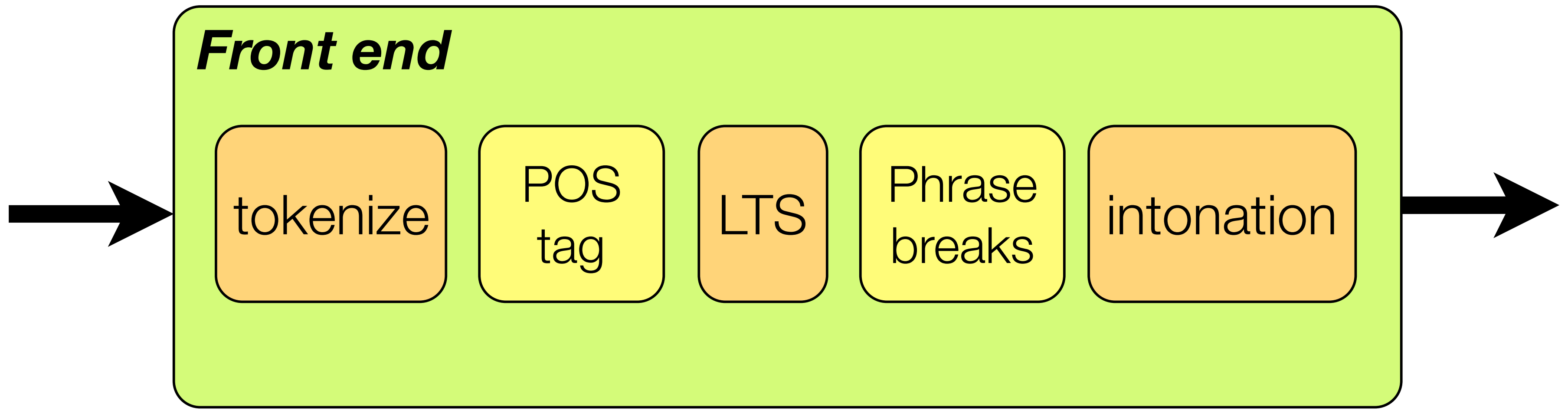
Tokeniser

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

Unicode character properties are used to tokenise the text with a language-independent regular expression

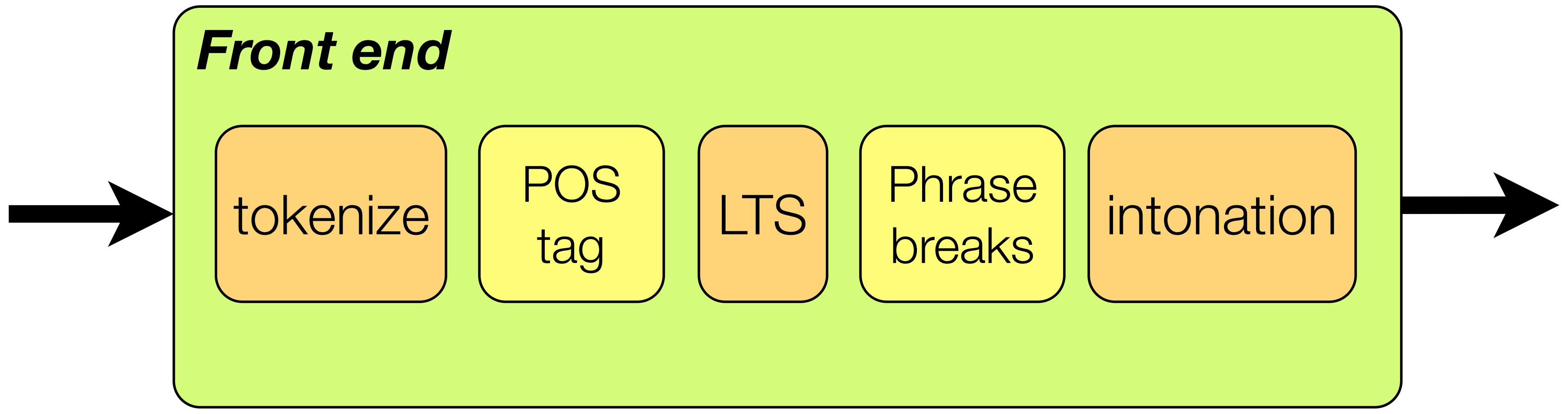
```
([\p{L}||\p{N}||\p{M}]+)
```

Tokeniser



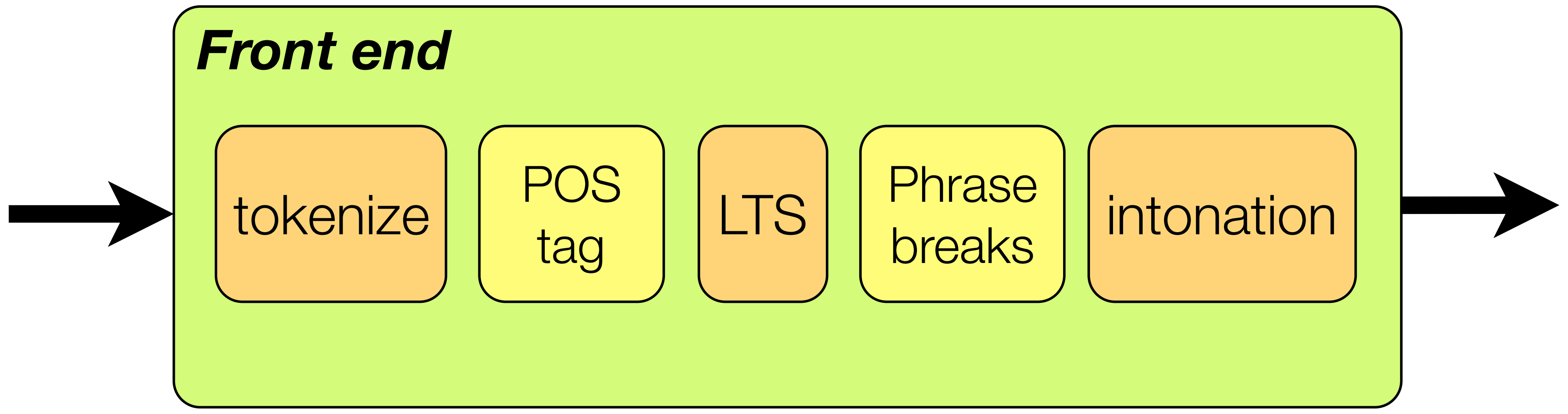
```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

Tokeniser



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

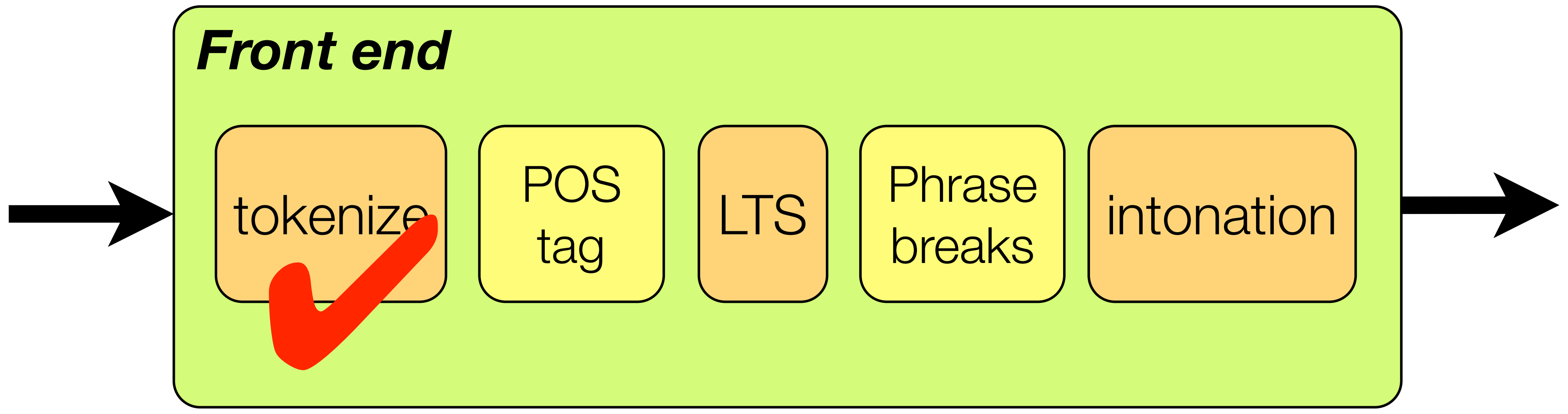
Unicode is also used to classify tokens as words, space, and punctuation



Tokeniser

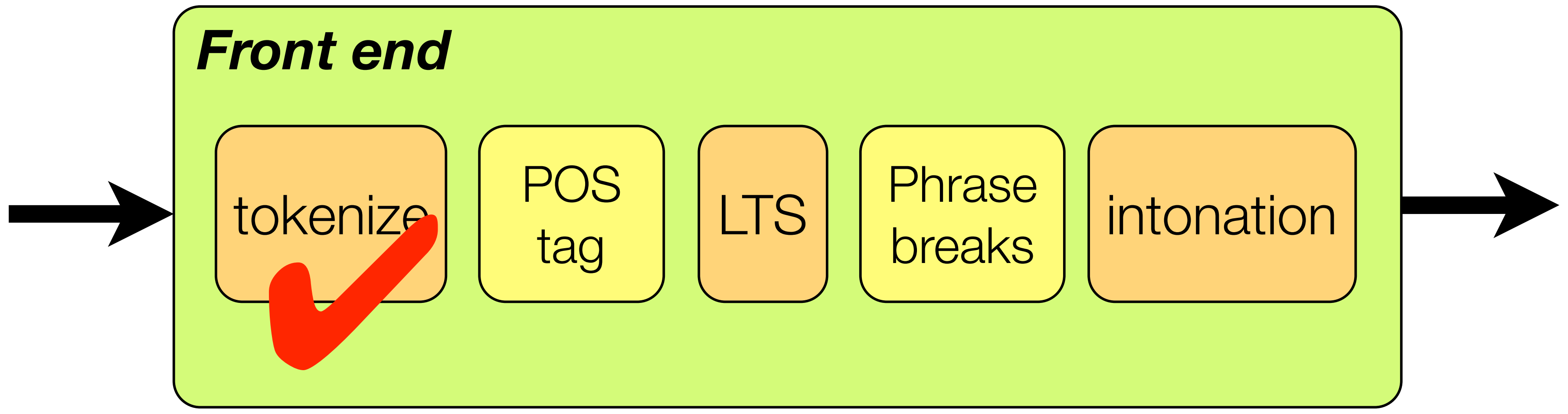
```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

Tokeniser



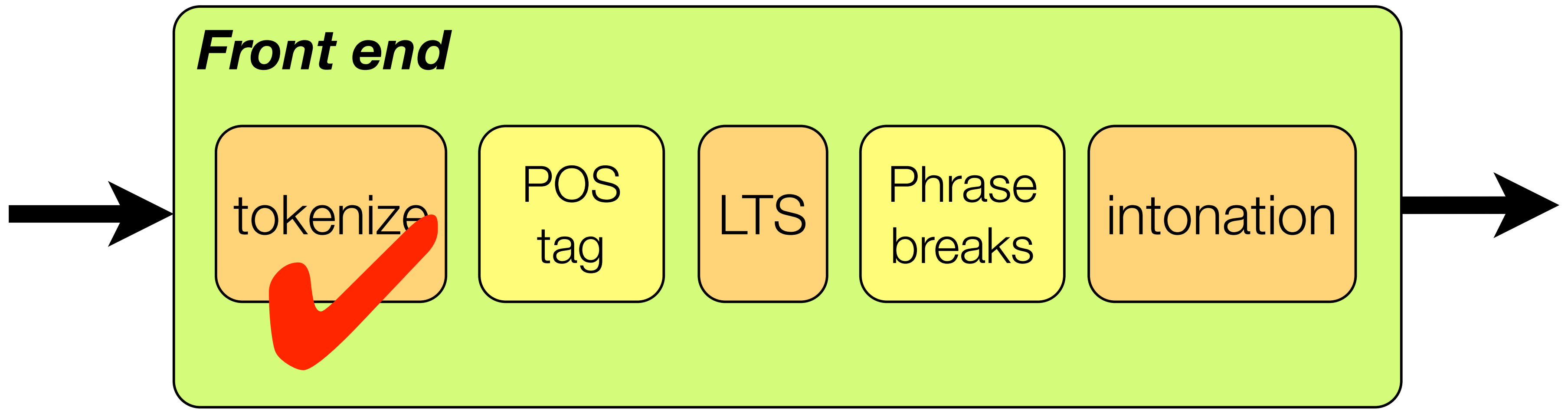
```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

POS tagging?



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

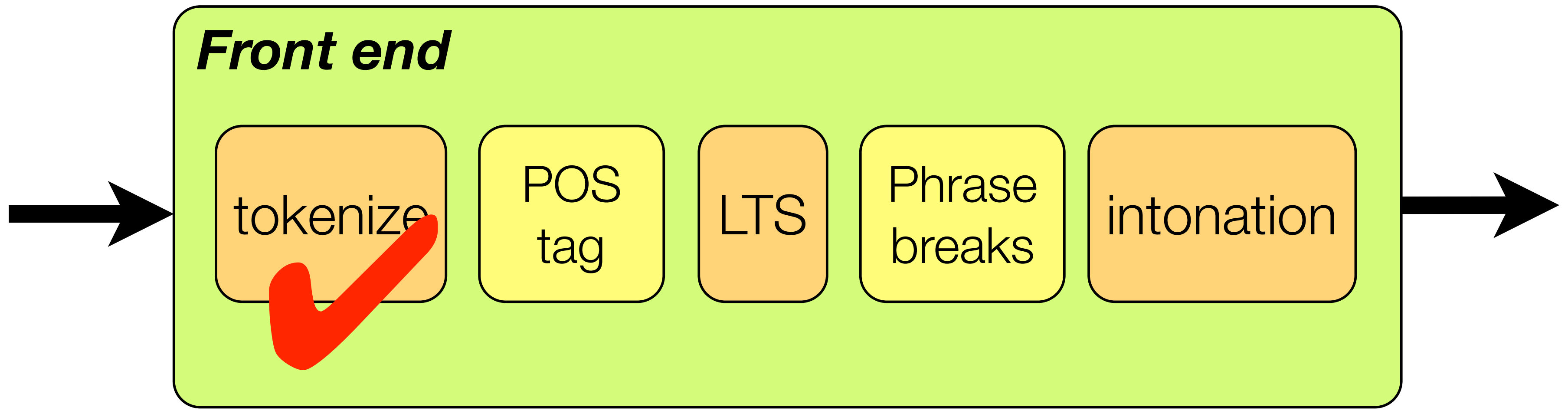
POS tagging?



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text=".   token_class="punctuation"/>
  <token text="   ND_" token_class="_END_"/>
</utt
```

High frequency word

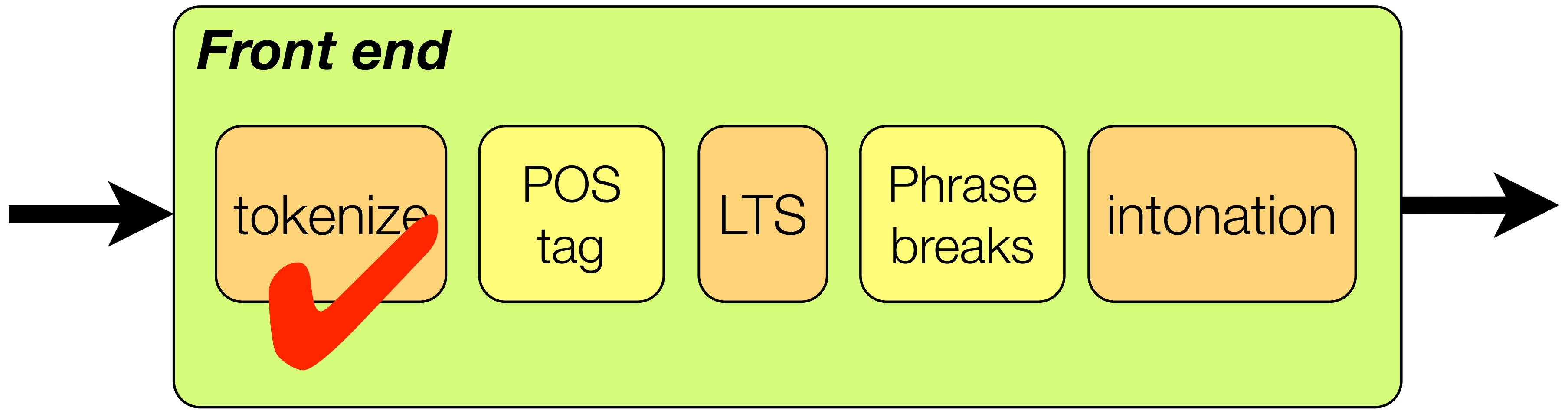
POS tagging?



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_cla
  <token text="mzozo"
  <token text=" " token_cla
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

Mid-frequency word

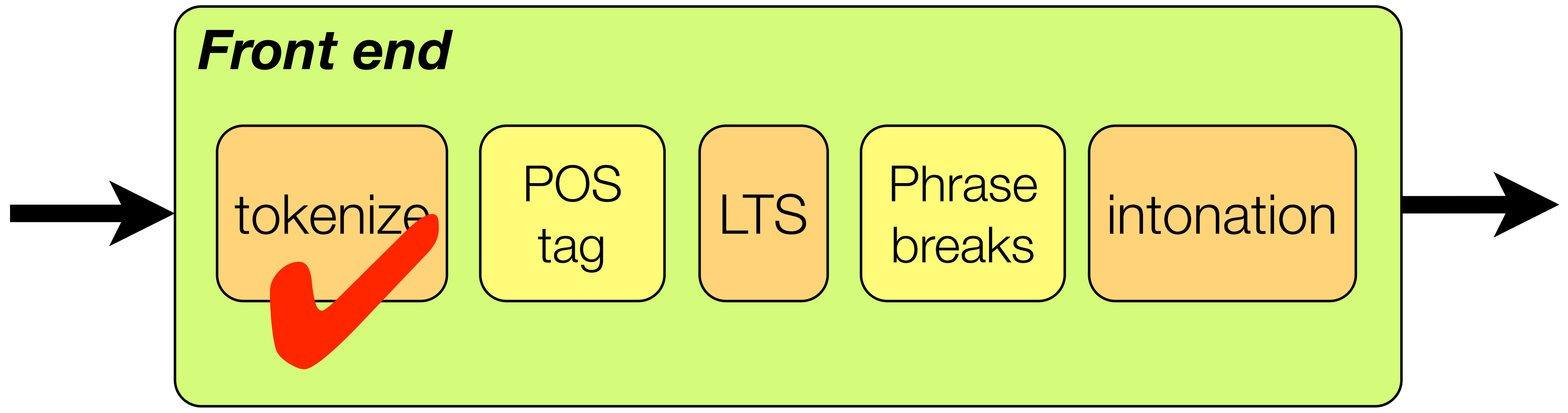
POS tagging?



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text                         D_"/>
  <token text                         word"/>
  <token text=    token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

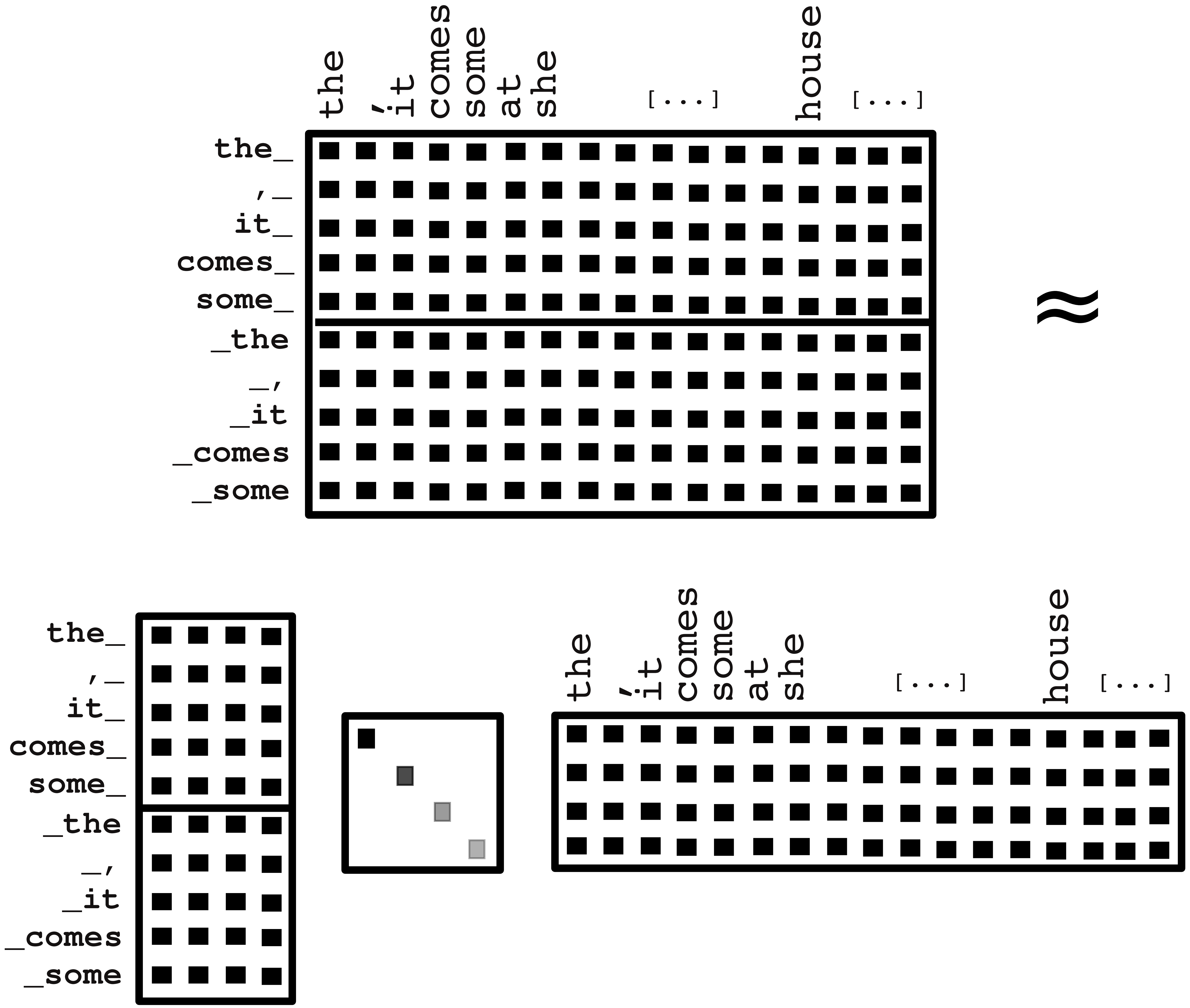
Mzozo often preceded by *na*

POS tagging?

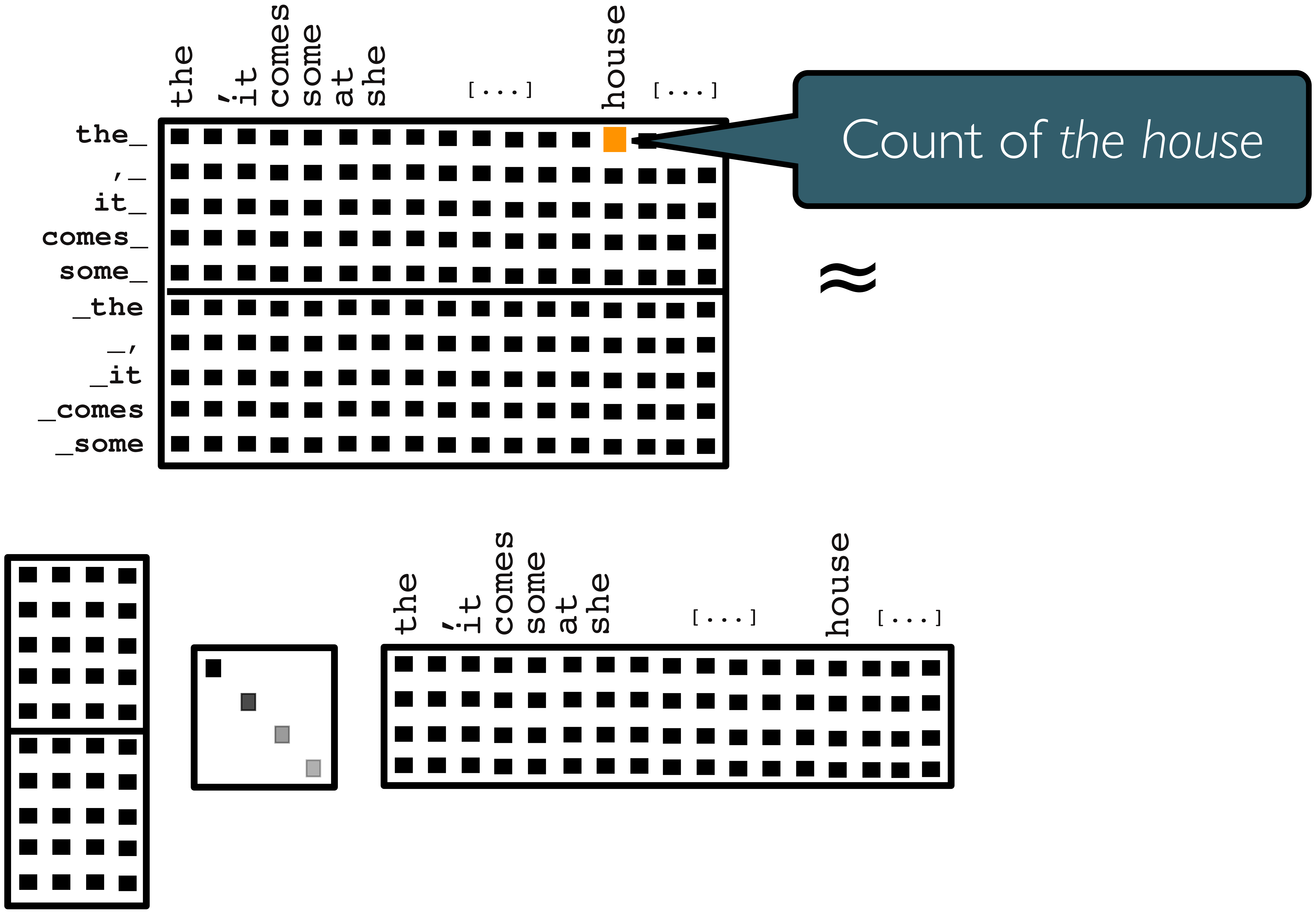


```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

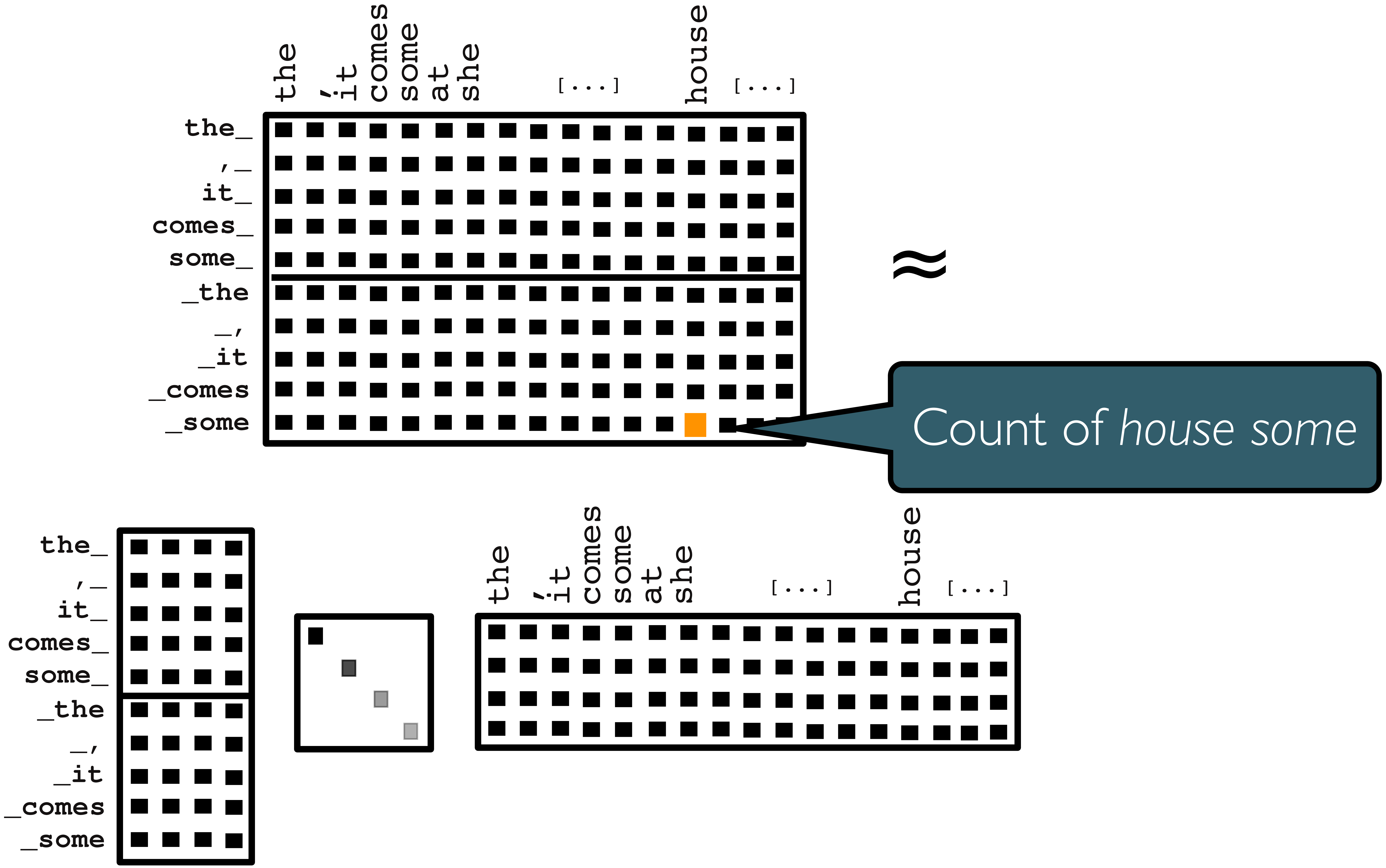
Distributional word vectors as Part Of Speech (POS) tag substitutes



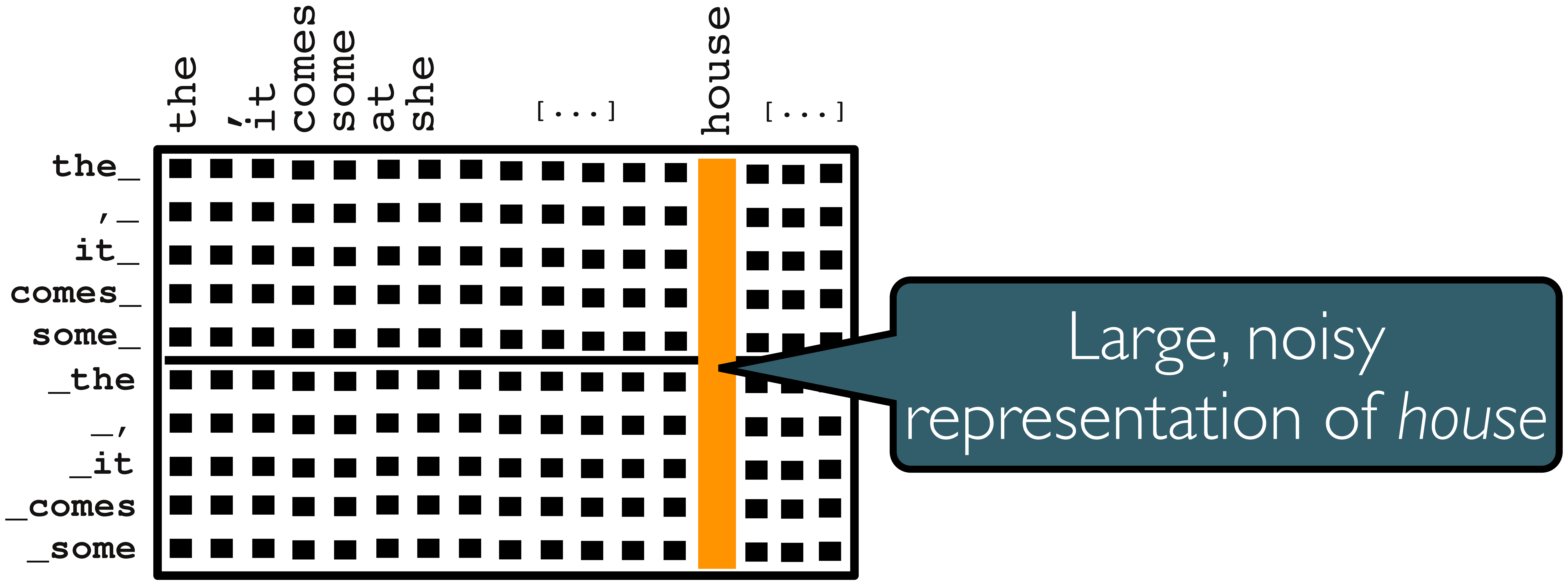
Distributional word vectors as Part Of Speech (POS) tag substitutes



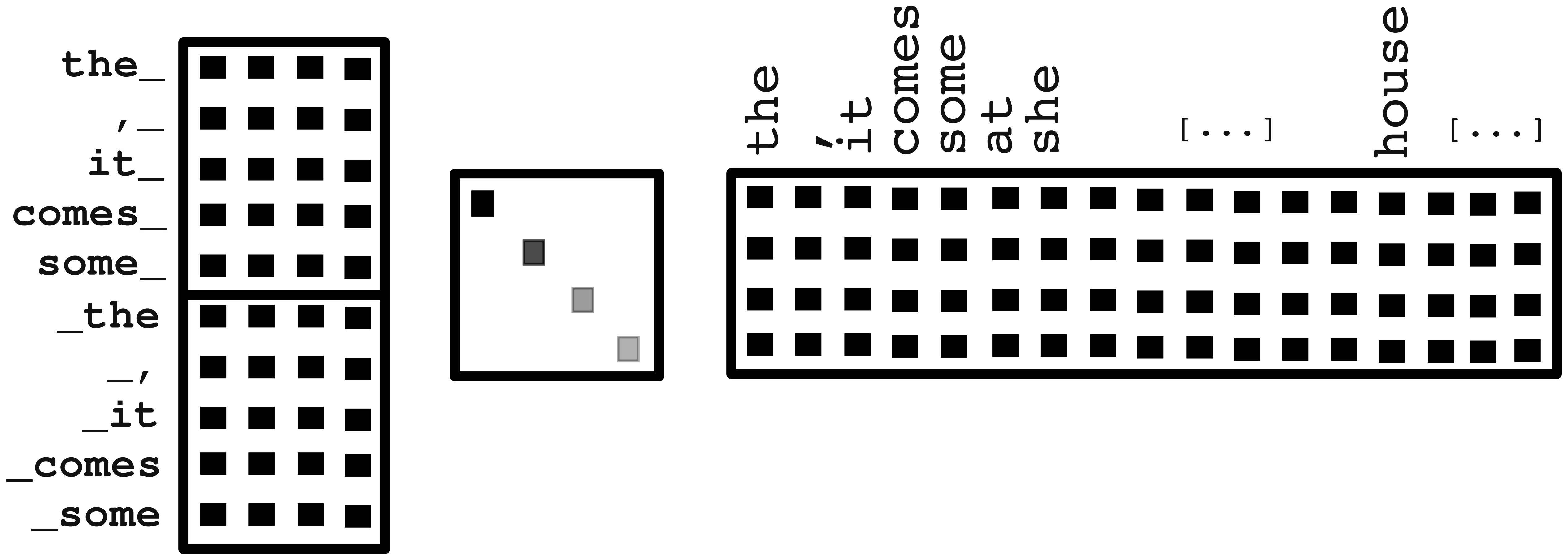
Distributional word vectors as Part Of Speech (POS) tag substitutes



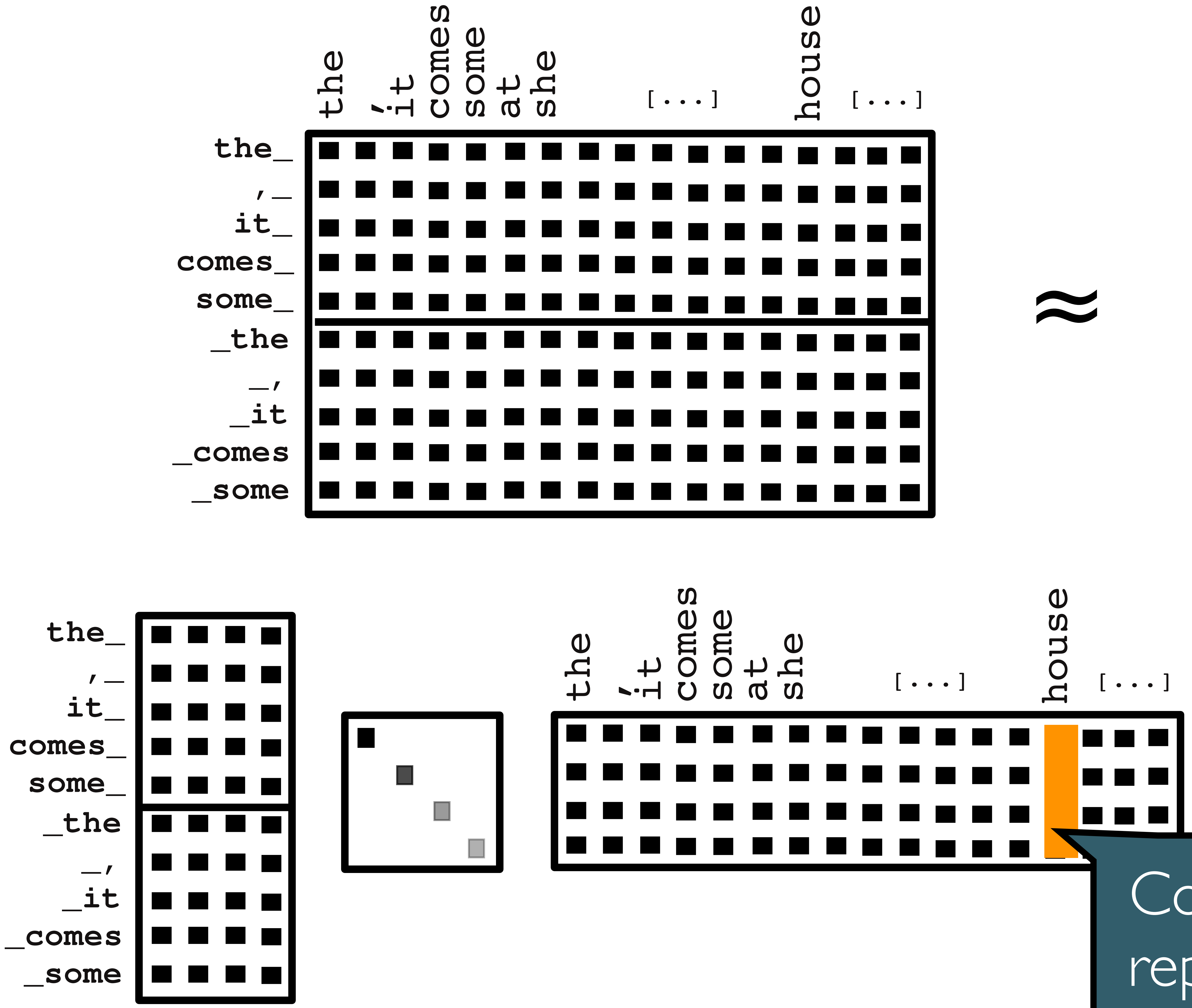
Distributional word vectors as Part Of Speech (POS) tag substitutes



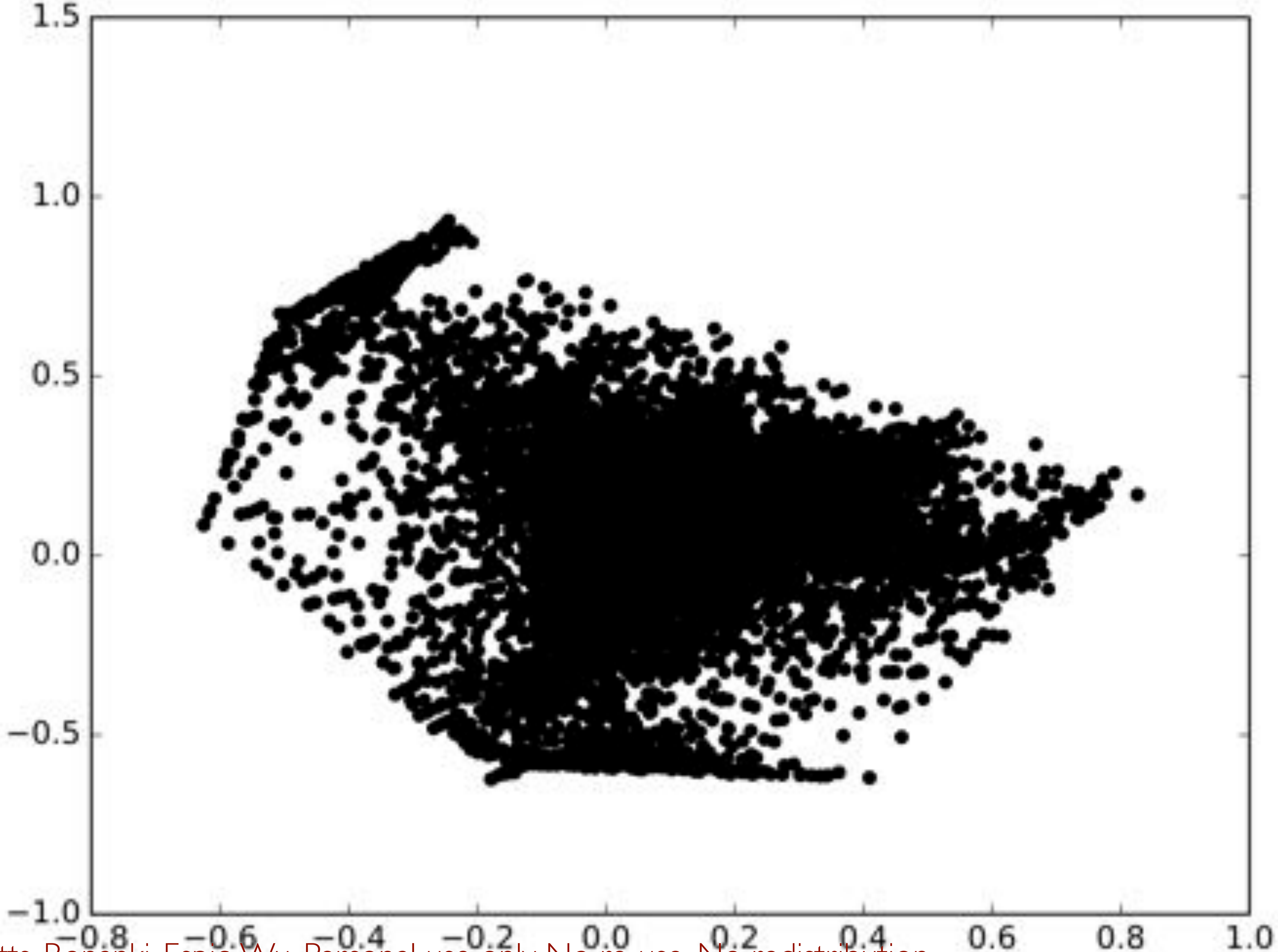
Large, noisy representation of *house*

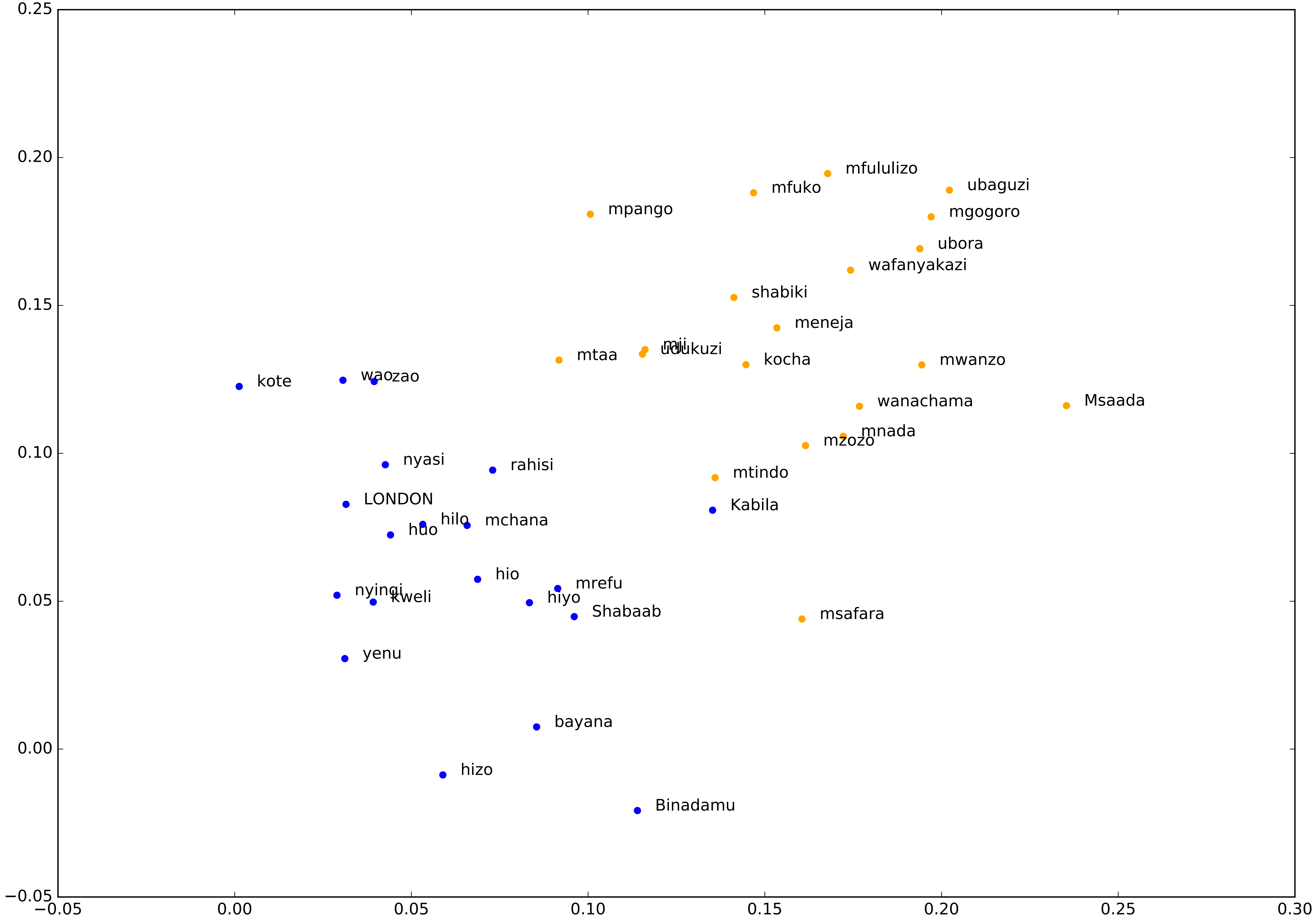


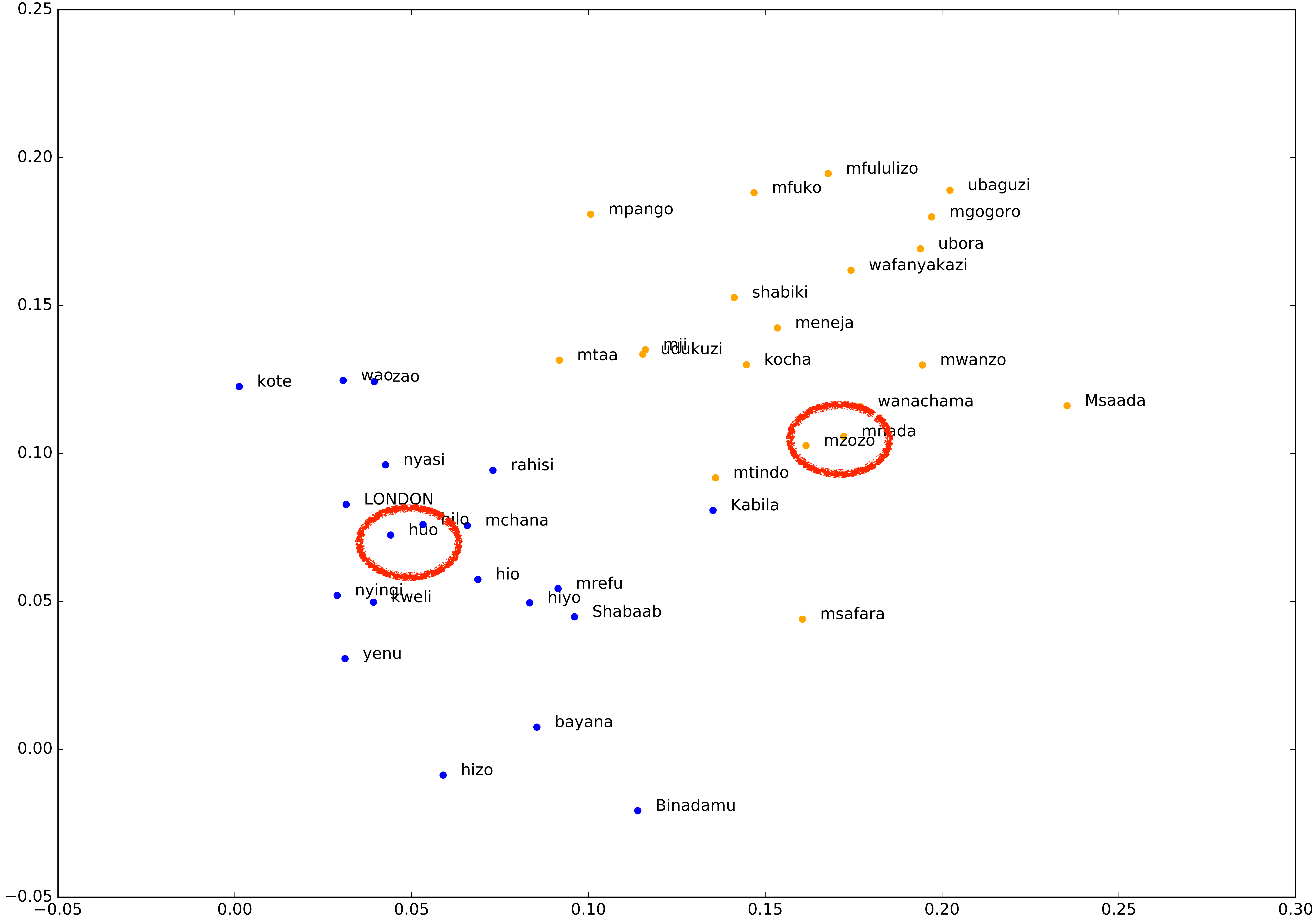
Distributional word vectors as Part Of Speech (POS) tag substitutes



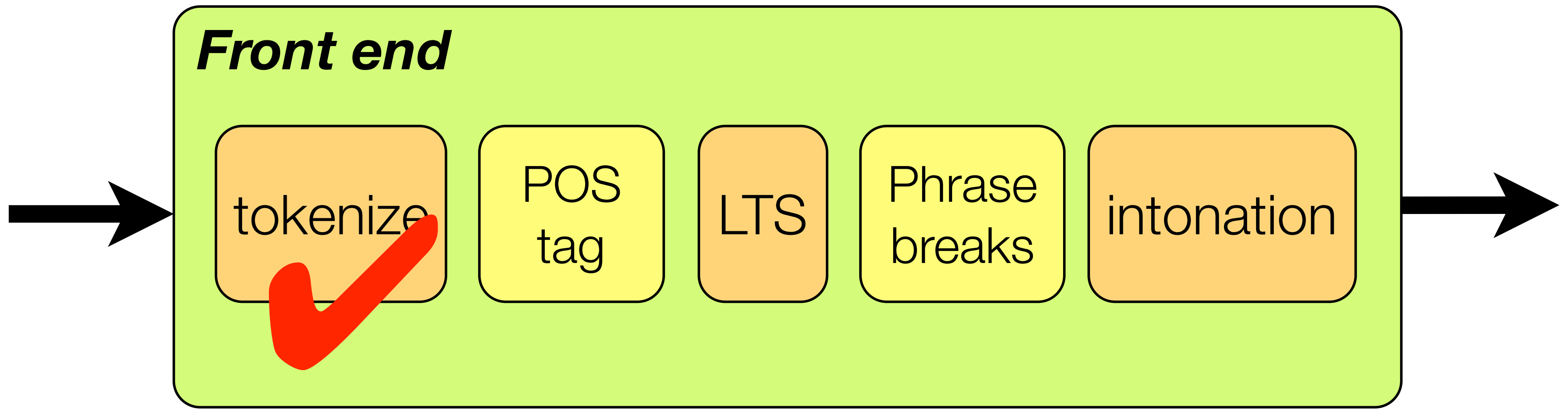
Compact and denoised representation of *house*


1.5
1.0
0.5
0.0
−0.5
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
1.0


0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30
mfululizo
ubaguzi
mfuko
mpango
mgogoro
ubora
wafanyakazi
shabiki
meneja
mji
udukuzi
mtaa
kocha
mwanzo
kote
wao
zao
wanachama
Msaada
mnada
mzozo
nyasi
rahisi
mtindo
LONDON
Kabila
hilo
mchana
huo
hio
mrefu
nyingi
kweli
hiyo
Shabaab
msafara
yenu
bayana
hizo
Binadamu

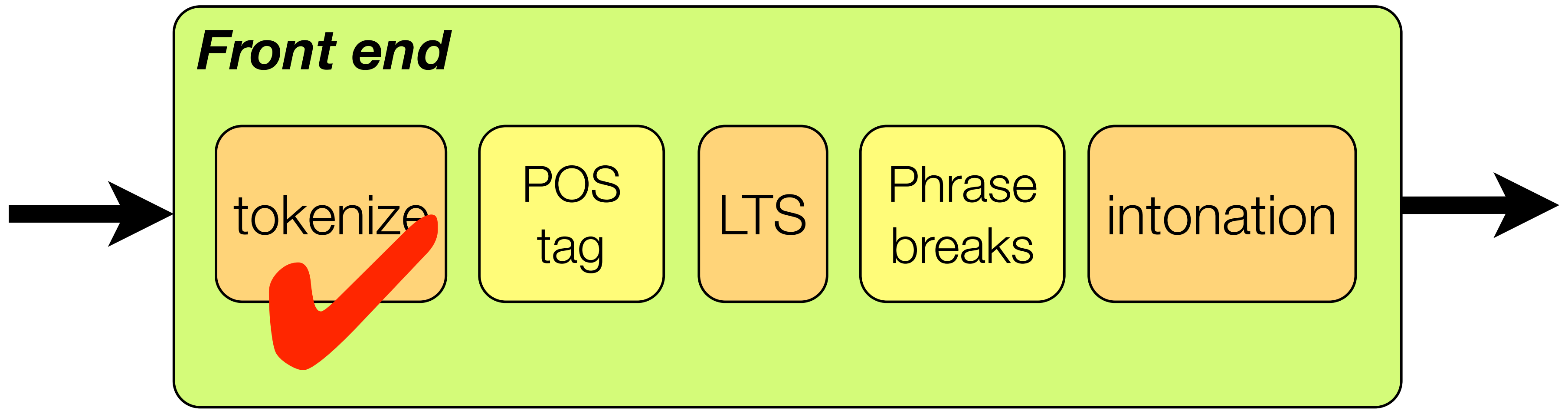

0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30
mfululizo
ubaguzi
mfuko
mpango
mgogoro
ubora
wafanyakazi
shabiki
meneja
mji
udukuzi
mtaa
kocha
mwanzo
kote
wao
zao
wanachama
Msaada
mnada
mzozo
nyasi
rahisi
mtindo
LONDON
Kabila
hilo
mchana
huo
hio
mrefu
nyingi
kweli
hiyo
Shabaab
msafara
yenu
bayana
hizo
Binadamu

Word vectors as a substitute for POS tags



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

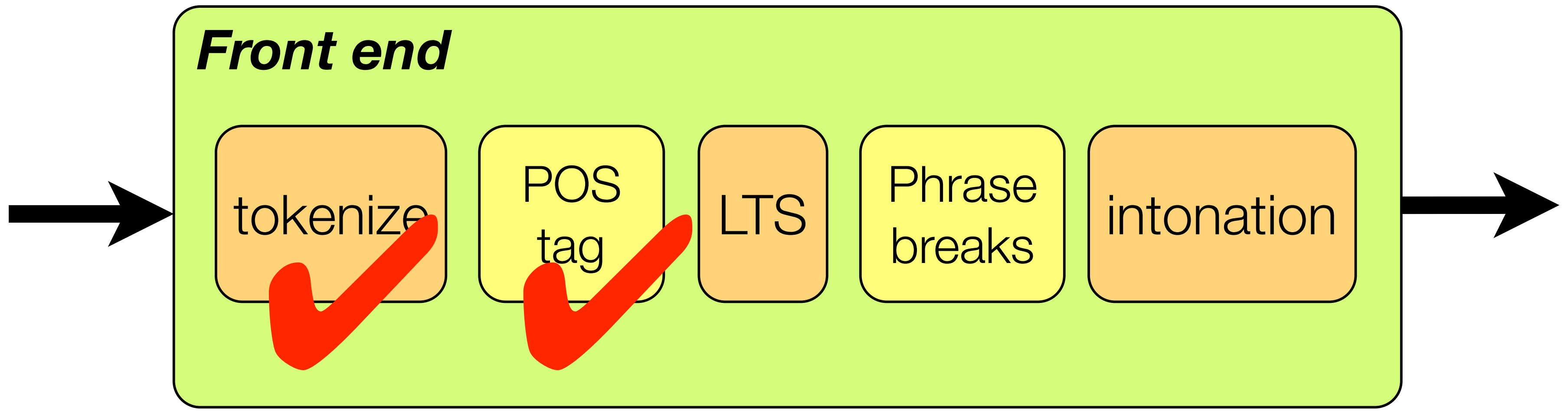
Word vectors as a substitute for POS tags



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

```
mzee       0.48536  0.09108 -0.0777
mzigo      0.24160  0.29423  0.0976
mziki      0.17319  0.18797  0.2416
mzima      0.15011  0.14782  0.1343
mzinga     0.54811  0.76613 -0.3259
mzio       0.16126  0.12330  0.2577
mzito      0.40942  0.31533 -0.1686
mzizi      0.54811  0.76613 -0.3259
mzozo      0.15992  0.10262  0.1615
mzungu     0.38154 -0.18574  0.0152
mzunguko   0.14774  0.13449  0.1957
mzuri      0.15253  0.14582  0.0649
n          0.62711 -0.43579 -0.0456
na         0.30476  0.07495 -0.1140
naam       0.73705 -0.32054  0.2075
naamini    0.29768 -0.17173  0.0118
nacho      0.46656  0.46190 -0.2337
nadhani    0.37699 -0.07810 -0.0740
nadharia   0.16468  0.22734  0.1558
```

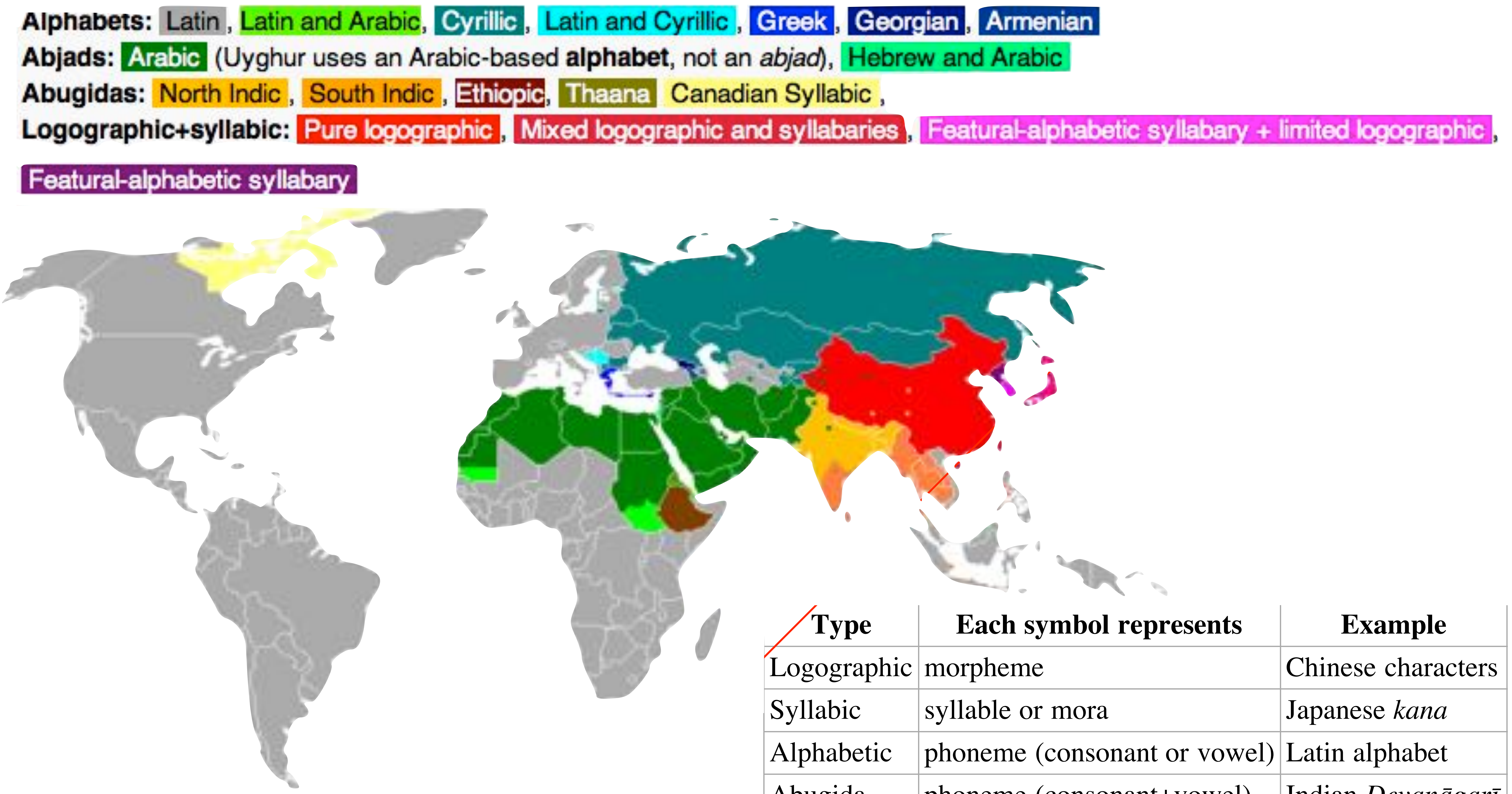
Word vectors as a substitute for POS tags



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236" processors_used=",word_splitter">
  <token text="_END_" token_class="_END_"/>
  <token text="Khartoum" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="imejitenga" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word"/>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word"/>
  <token text="." token_class="punctuation"/>
  <token text="_END_" token_class="_END_"/>
</utt>
```

```
mzee       0.48536  0.09108 -0.0777
mzigo      0.24160  0.29423  0.0976
mziki      0.17319  0.18797  0.2416
mzima      0.15011  0.14782  0.1343
mzinga     0.54811  0.76613 -0.3259
mzio       0.16126  0.12330  0.2577
mzito      0.40942  0.31533 -0.1686
mzizi      0.54811  0.76613 -0.3259
mzozo      0.15992  0.10262  0.1615
mzungu     0.38154 -0.18574  0.0152
mzunguko   0.14774  0.13449  0.1957
mzuri      0.15253  0.14582  0.0649
n          0.62711 -0.43579 -0.0456
na         0.30476  0.07495 -0.1140
naam       0.73705 -0.32054  0.2075
naamini    0.29768 -0.17173  0.0118
nacho      0.46656  0.46190 -0.2337
nadhani    0.37699 -0.07810 -0.0740
nadharia   0.16468  0.22734  0.1558
```

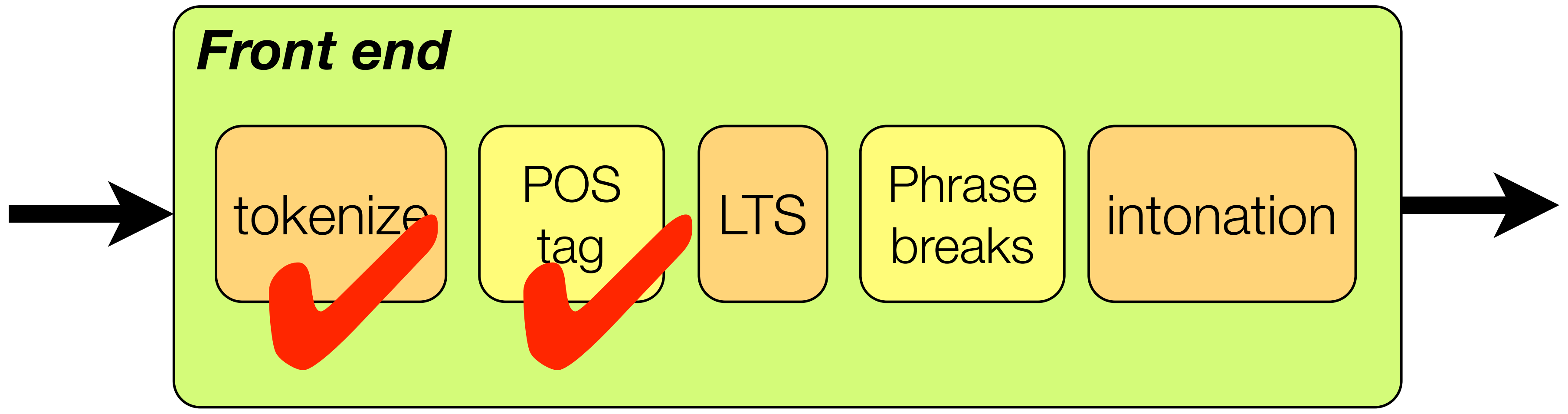
Letters as a substitute for phonemes


Alphabets: Latin, Latin and Arabic, Cyrillic, Latin and Cyrillic, Greek, Georgian, Armenian
Abjads: Arabic (Uyghur uses an Arabic-based alphabet, not an abjad), Hebrew and Arabic
Abugidas: North Indic, South Indic, Ethiopic, Thaana, Canadian Syllabic,
Logographic+syllabic: Pure logographic, Mixed logographic and syllabaries, Featural-alphabetic syllabary + limited logographic,
Featural-alphabetic syllabary

Type	Each symbol represents	Example
Logographic	morpheme	Chinese characters
Syllabic	syllable or mora	Japanese *kana*
Alphabetic	phoneme (consonant or vowel)	Latin alphabet
Abugida	phoneme (consonant+vowel)	Indian *Devanāgarī*
Abjad	phoneme (consonant)	Arabic alphabet
Featural	phonetic feature	Korean *hangul*

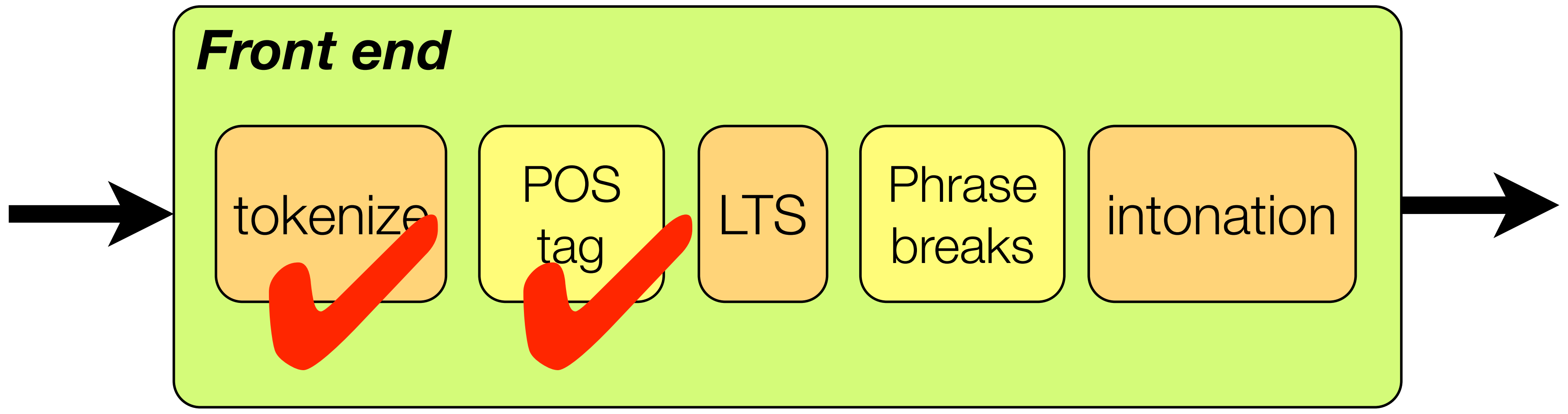
Map credit: Wikipedia by JWB
CC-BY-SA-3.0 via Wikimedia Commons

“Letter to sound”



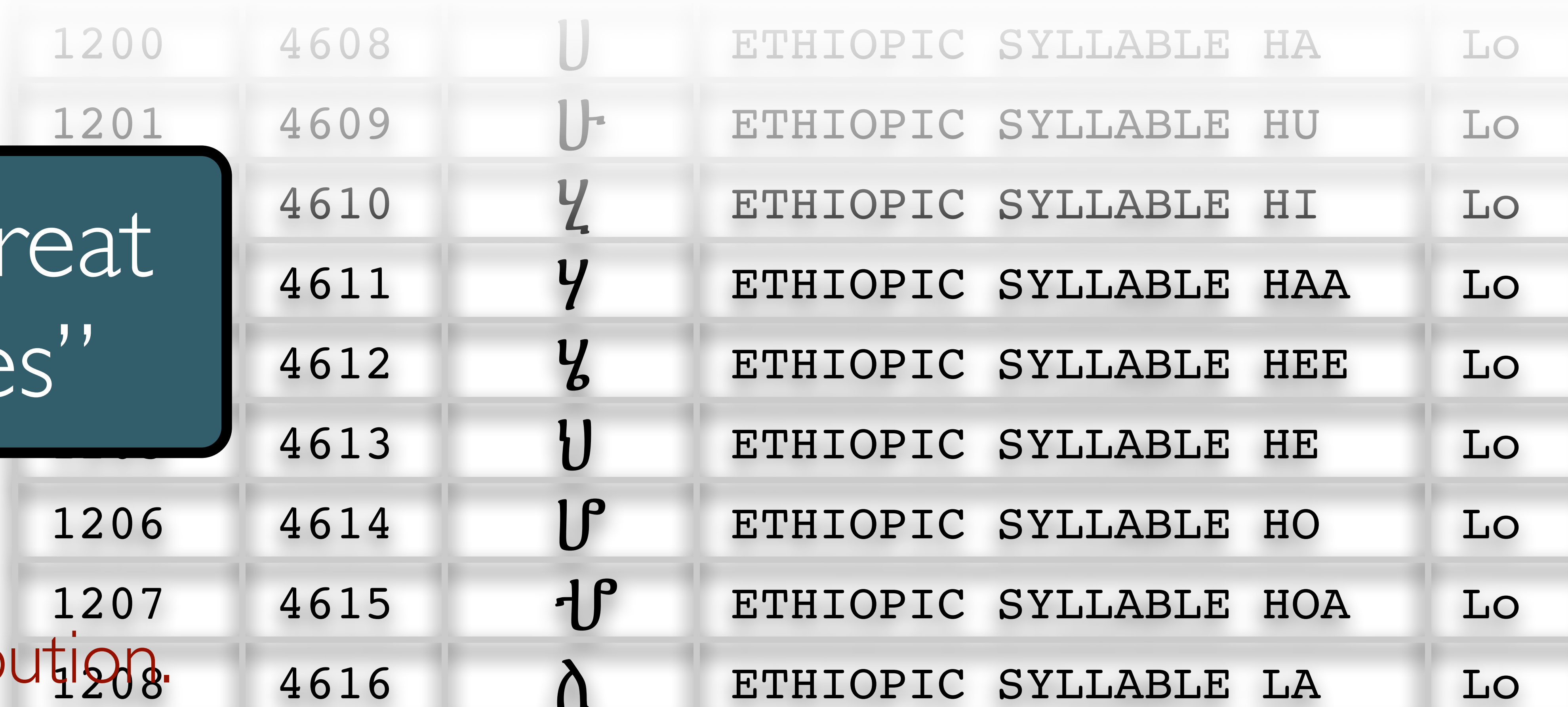
```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5">
  <token text="_END_" token_class="_END_">...</token>
  <token text="Khartoum" token_class="word">...</token>
  <token text=" " token_class="space">
    <segment pronunciation="_POSS_PAUSE_"/>
  </token>
  <token text="imejitenga" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="na" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="mzozo" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="huo" token_class="word">
    <segment pronunciation="h"/>
    <segment pronunciation="u"/>
    <segment pronunciation="o"/>
  </token>
  <token text="." token_class="punctuation">
    <segment pronunciation="_PROB_PAUSE_"/>
  </token>
```

“Letter to sound”



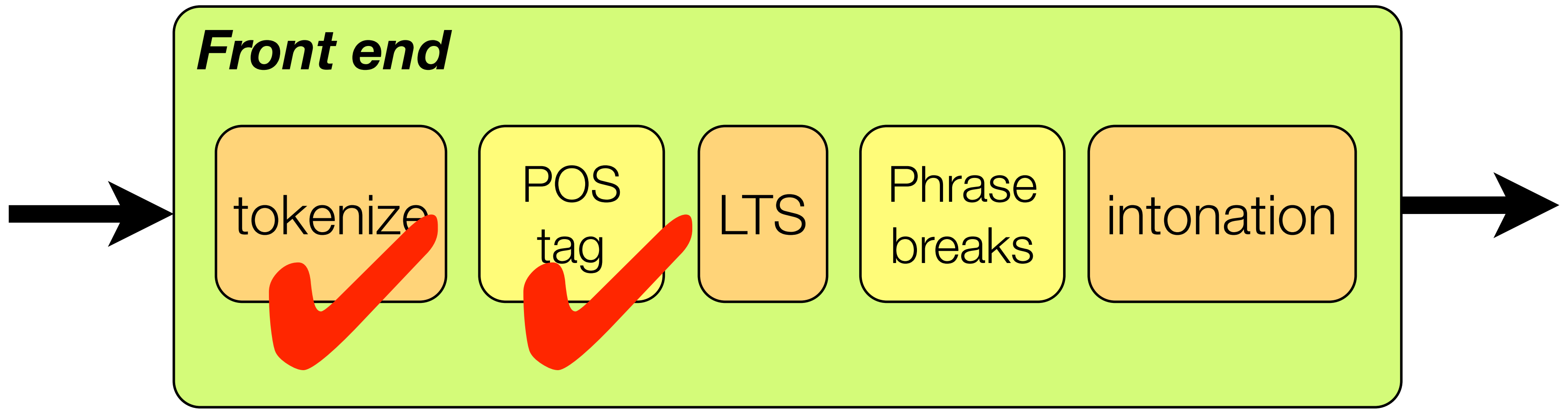
```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5">
  <token text="_END_" token_class="_END_">...</token>
  <token text="Khartoum" token_class="word">...</token>
  <token text=" " token_class="space">
    <segment pronunciation="_POSS_PAUSE_"/>
  </token>
  <token text="imejitenga" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="na" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="mzozo" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="huo" token_class="wor
    <segment pronunciation="h"/>
    <segment pronunciation="u"/>
    <segment pronunciation="o"/>
  </token>
  <token text="." token_class="punctuation">
    <segment pronunciation="_PROB_PAUSE_"/>
```

Most naive case: treat letters as “phones”



1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
[illegible]	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
[illegible]	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
[illegible]	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
[illegible]	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo
1208	4616	ለ	ETHIOPIC SYLLABLE LA	Lo

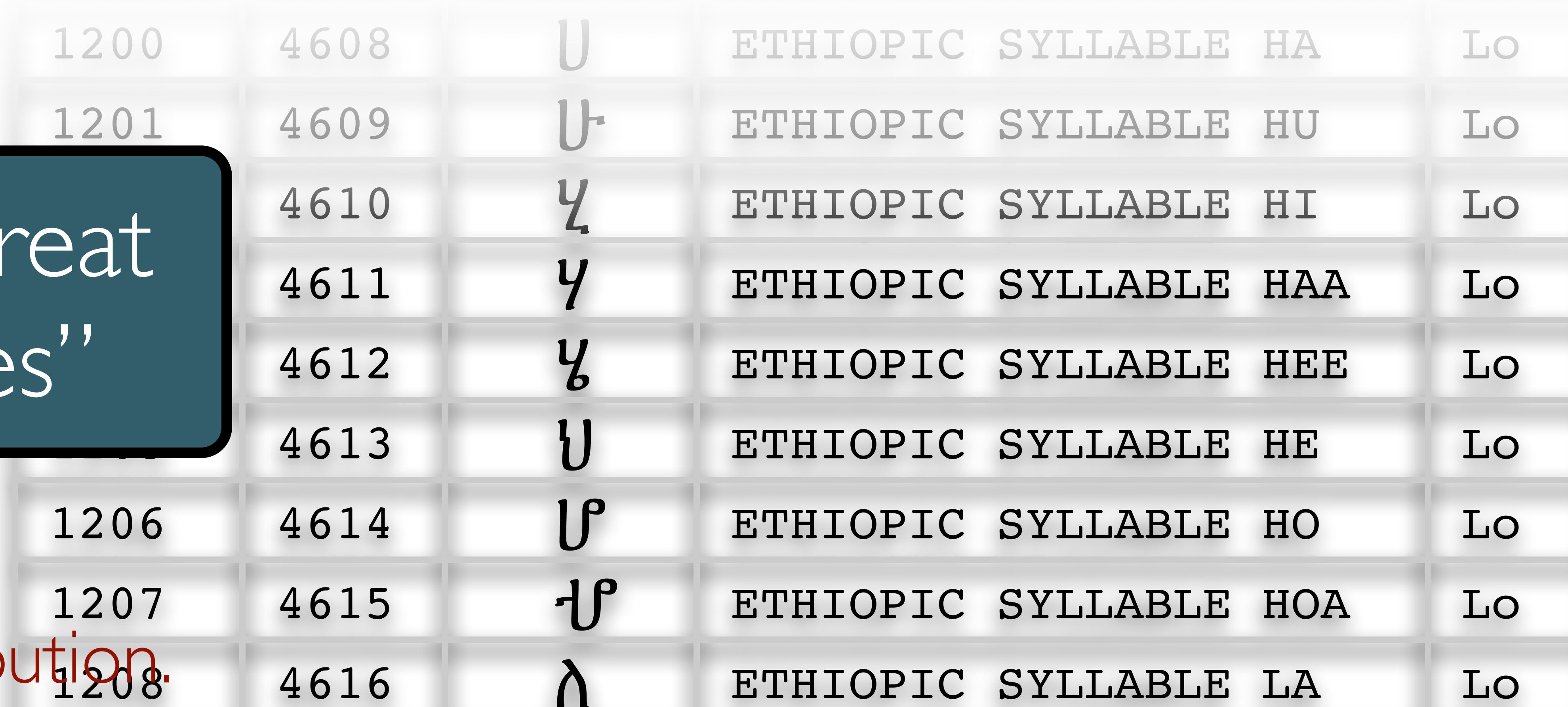
“Letter to sound”



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5">
  <token text="_END_" token_class="_END_">...</token>
  <token text="Khartoum" token_class="word">...<
  <token text=" " token_class="space">
    <segment pronunciation="_POSS_PAUSE_"/>
  </token>
  <token text="imejitenga" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="na" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="mzozo" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="huo" token_class="wor
    <segment pronunciation="h"/>
    <segment pronunciation="u"/>
    <segment pronunciation="o"/>
  </token>
  <token text="." token_class="punctuation">
    <segment pronunciation="_PROB_PAUSE_"/>
```

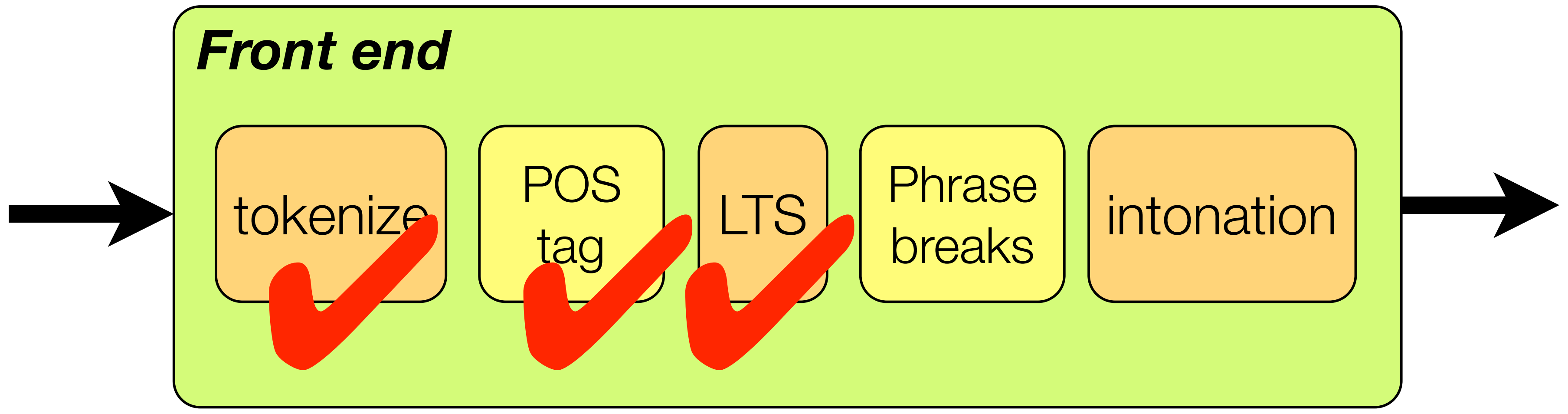
Initial guess at position of pauses

Most naive case: treat letters as “phones”



1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
[illegible]	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
[illegible]	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
[illegible]	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
[illegible]	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo
1208	4616	ለ	ETHIOPIC SYLLABLE LA	Lo

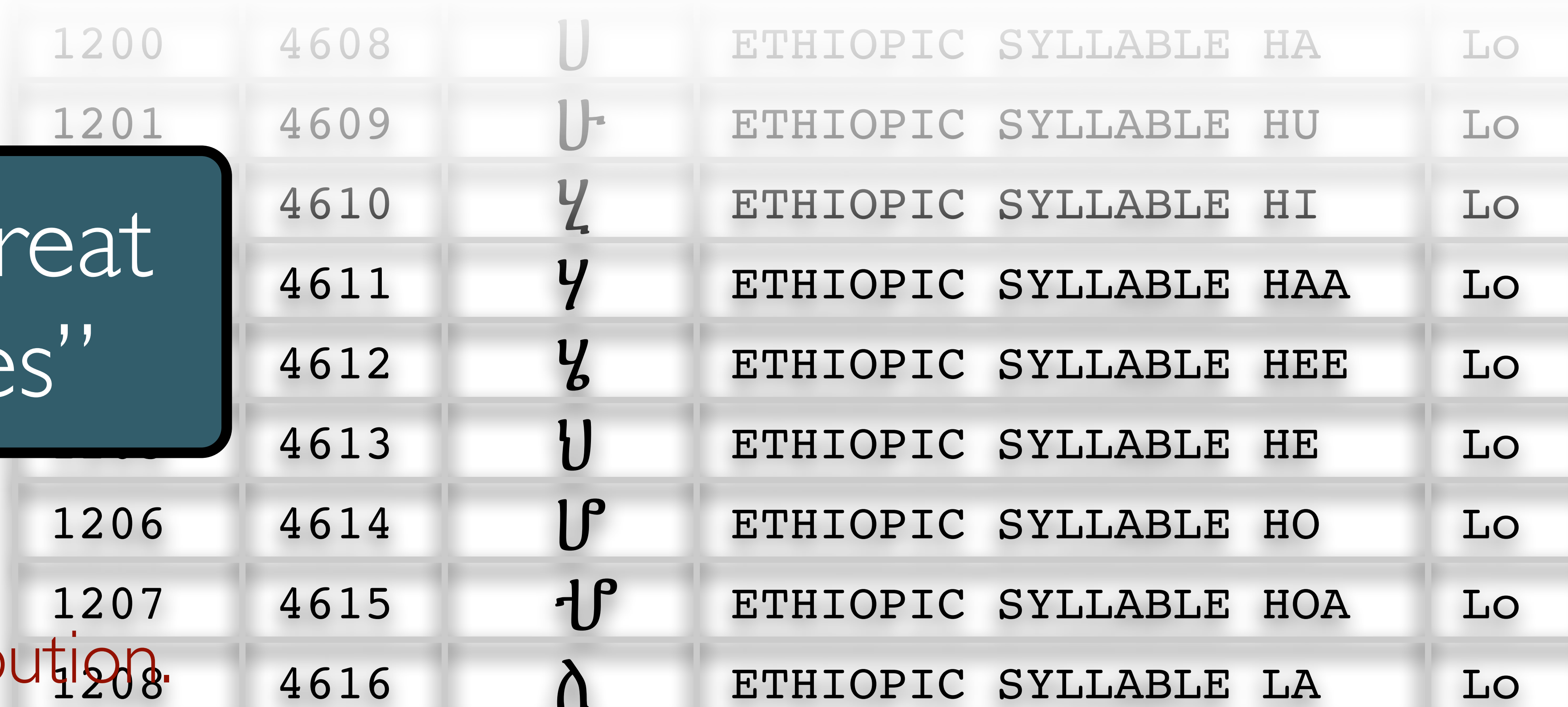
“Letter to sound”



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5">
  <token text="_END_" token_class="_END_">...</token>
  <token text="Khartoum" token_class="word">...<
  <token text=" " token_class="space">
    <segment pronunciation="_POSS_PAUSE_"/>
  </token>
  <token text="imejitenga" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="na" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="mzozo" token_class="word">...</token>
  <token text=" " token_class="space">...</token>
  <token text="huo" token_class="wor
    <segment pronunciation="h"/>
    <segment pronunciation="u"/>
    <segment pronunciation="o"/>
  </token>
  <token text="." token_class="punctuation">
    <segment pronunciation="_PROB_PAUSE_"/>
```

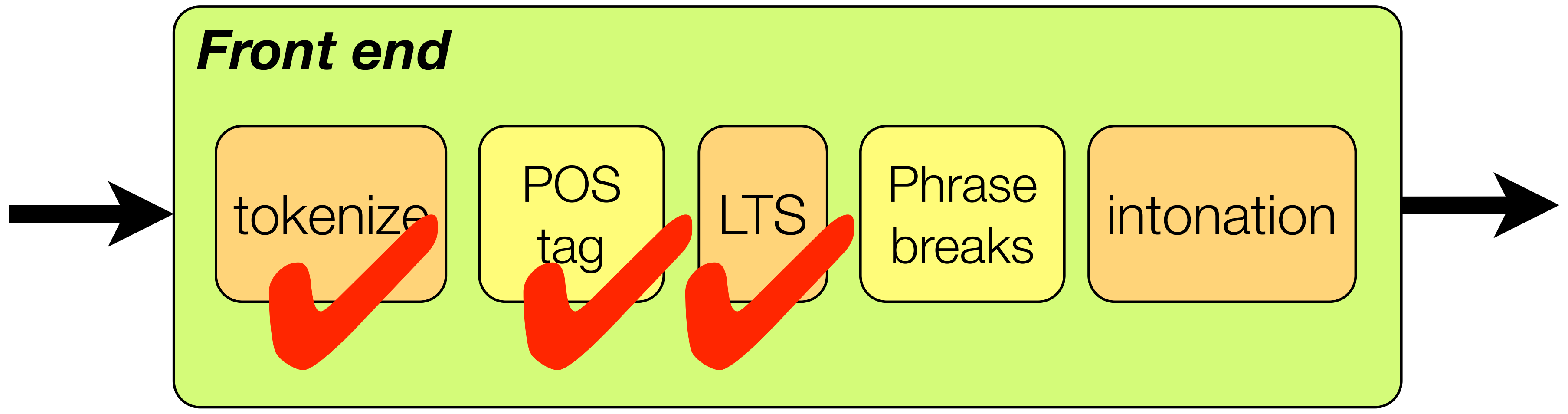
Initial guess at position of pauses

Most naive case: treat letters as “phones”

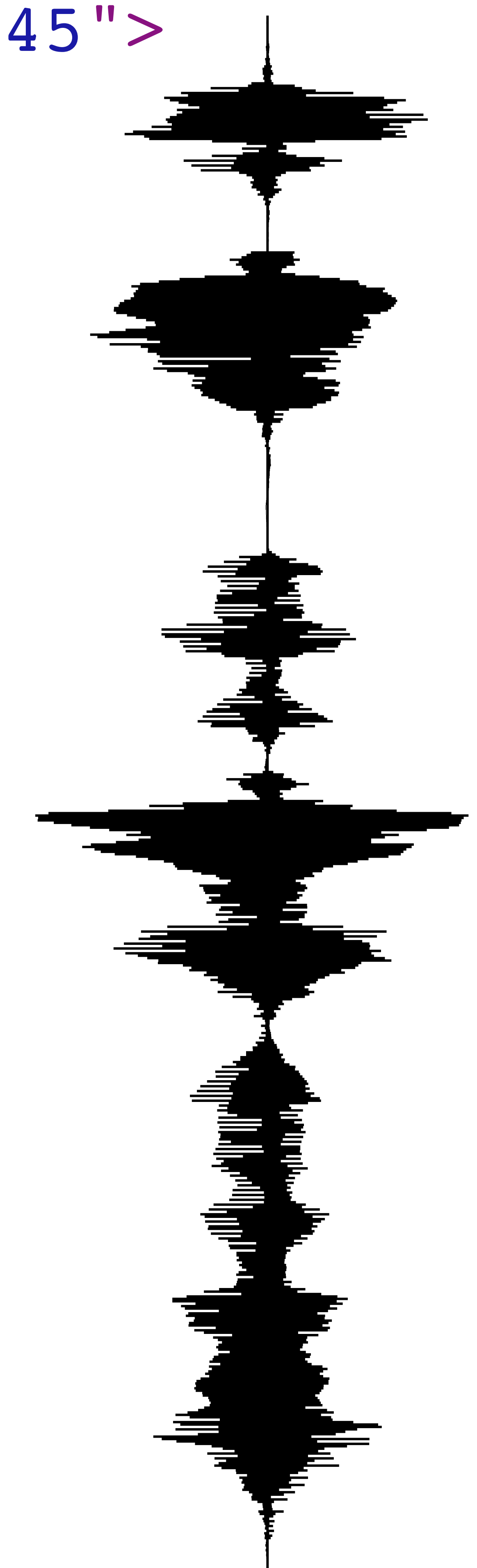


1200	4608	ሀ	ETHIOPIC SYLLABLE HA	Lo
1201	4609	ሁ	ETHIOPIC SYLLABLE HU	Lo
	4610	ሂ	ETHIOPIC SYLLABLE HI	Lo
	4611	ሃ	ETHIOPIC SYLLABLE HAA	Lo
	4612	ሄ	ETHIOPIC SYLLABLE HEE	Lo
	4613	ህ	ETHIOPIC SYLLABLE HE	Lo
1206	4614	ሆ	ETHIOPIC SYLLABLE HO	Lo
1207	4615	ሇ	ETHIOPIC SYLLABLE HOA	Lo
1208	4616	ለ	ETHIOPIC SYLLABLE LA	Lo

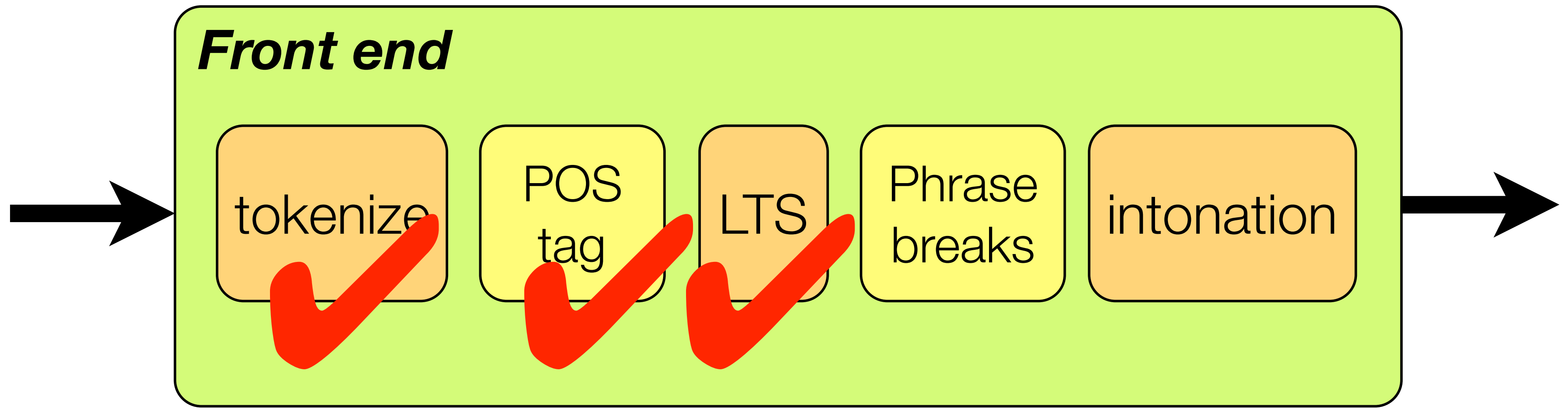
Forced alignment & silence detection



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word" start="2560" end="2660">...</token>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word" start="2975" end="3325">
    <segment pronunciation="h" start="2975" end="3000">
      <state start="2975" end="2980"/>
      <state start="2980" end="2985"/>
      <state start="2985" end="2990"/>
      <state start="2990" end="2995"/>
      <state start="2995" end="3000"/>
    </segment>
```

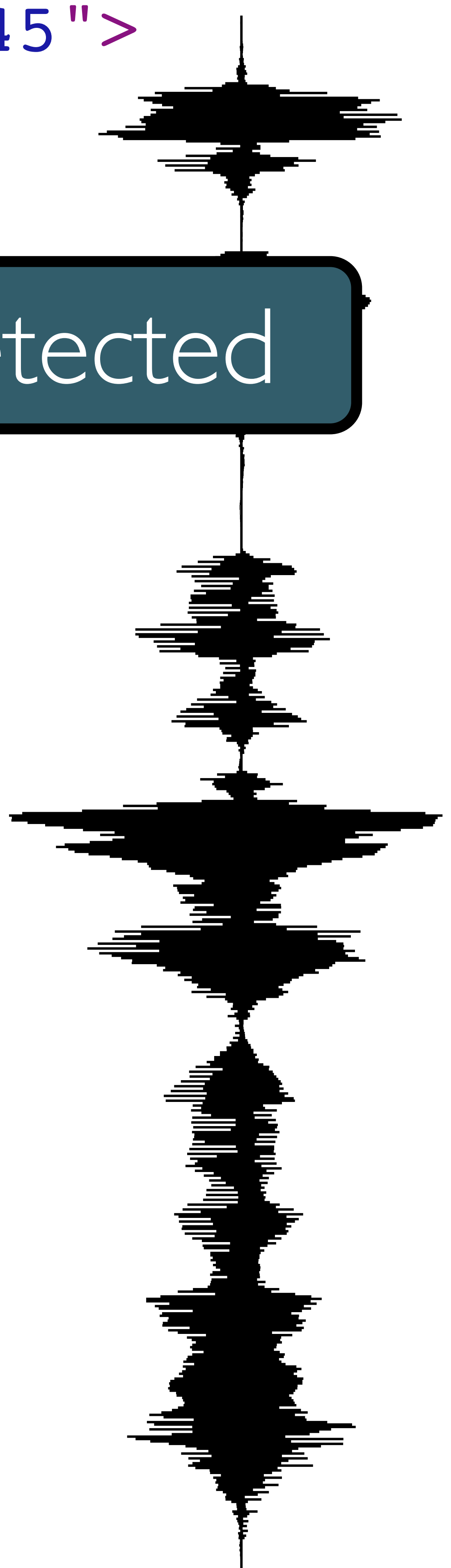


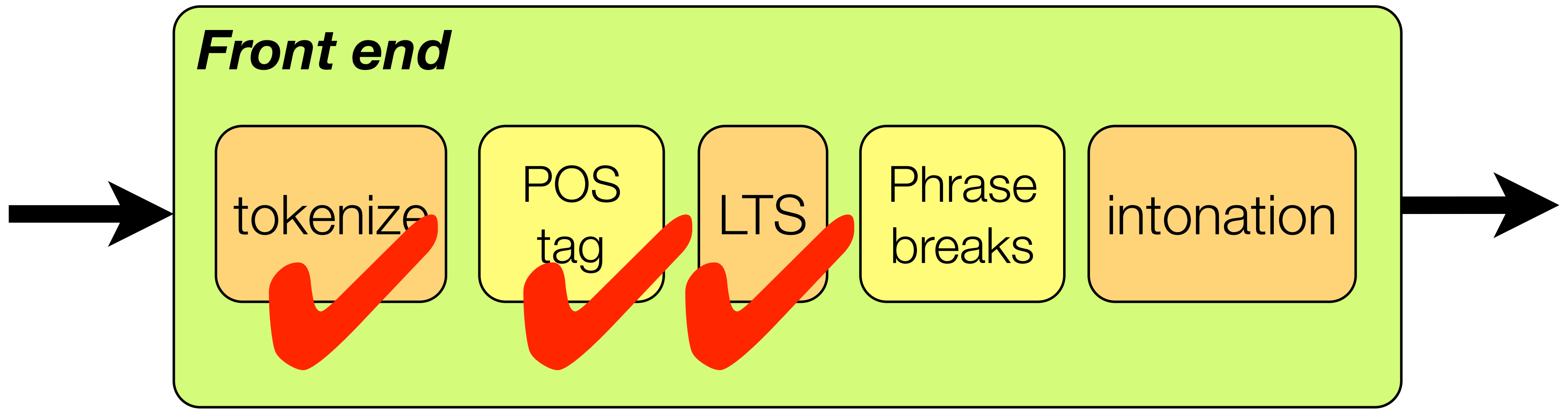
Forced alignment & silence detection



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...
  </token>
  <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word" start="2560" end="2660">...</token>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word" start="2975" end="3325">
    <segment pronunciation="h" start="2975" end="3000">
      <state start="2975" end="2980"/>
      <state start="2980" end="2985"/>
      <state start="2985" end="2990"/>
      <state start="2990" end="2995"/>
      <state start="2995" end="3000"/>
    </segment>
```

Silent segments detected

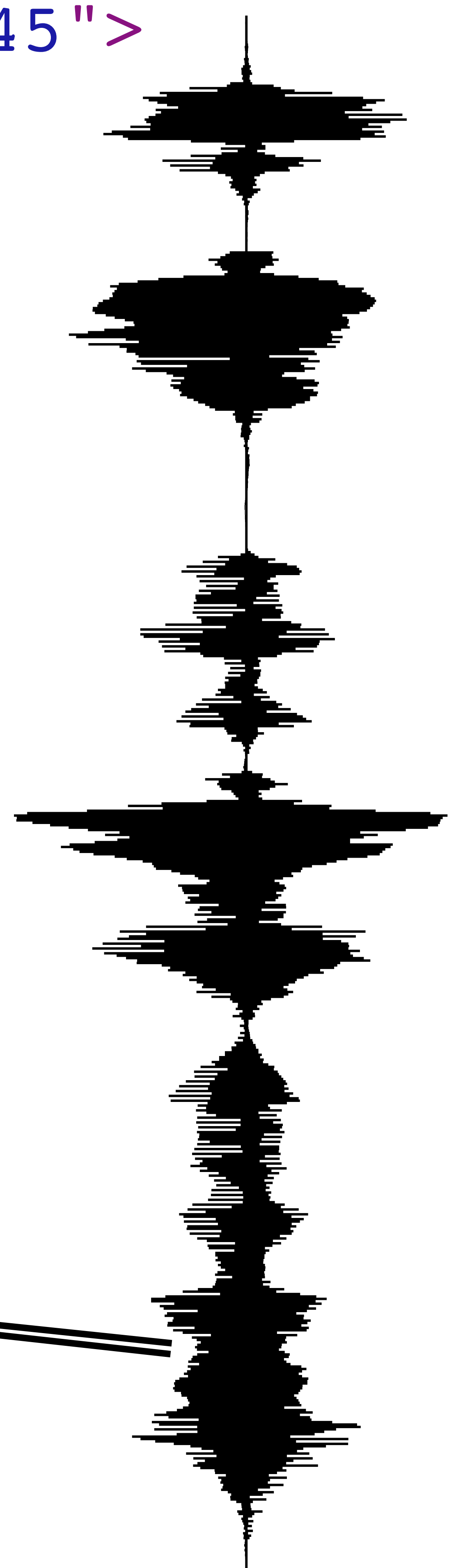




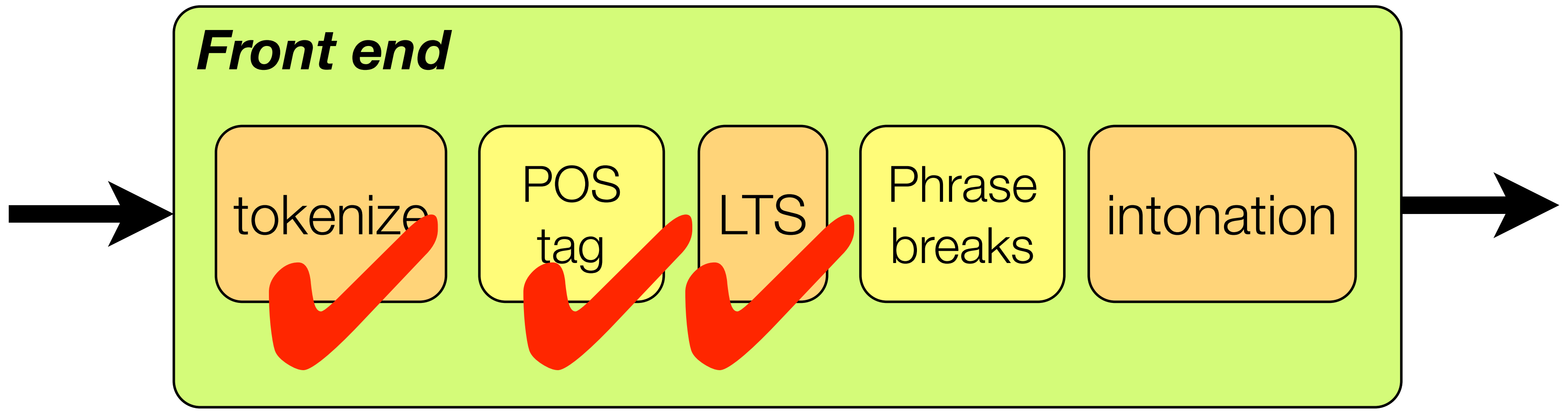
Forced alignment & silence detection

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
  <token text=" " token_class="space"/>
  <token text="na" token_class="word" start="2560" end="2660">...</token>
  <token text=" " token_class="space"/>
  <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
  <token text=" " token_class="space"/>
  <token text="huo" token_class="word" start="2975" end="3325">
    <segment pronunciation="h" start="2975" end="3000">
      <state start="2975" end="2980"/>
      <state start="2980" end="2985"/>
      <state start="2985" end="2990"/>
      <state start="2990" end="2995"/>
      <state start="2995" end="3000"/>
    </segment>
```

Subphone state timings added

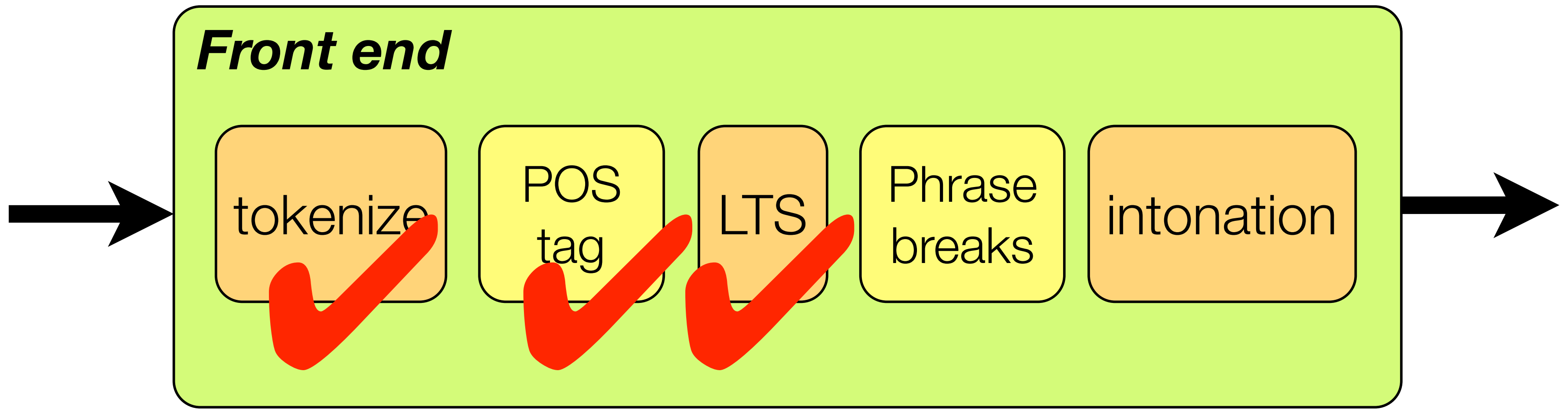


Phrasing



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner,pause_predictor,phrase_maker"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <phrase>
    <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  </phrase>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <phrase>
    <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
    <token text=" " token_class="space"/>
    <token text="na" token_class="word" start="2560" end="2660">...</token>
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

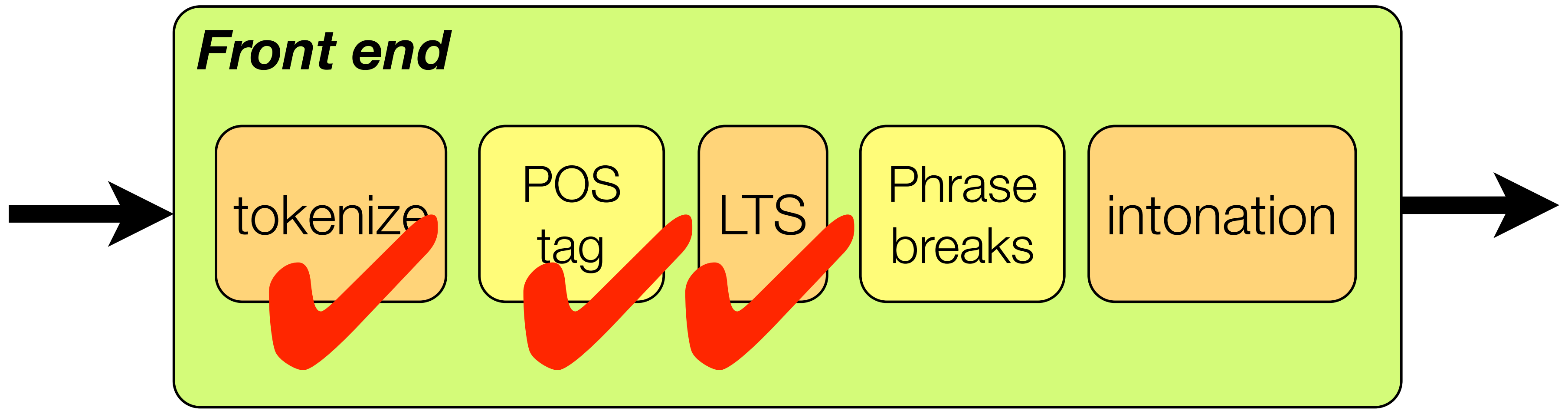
Phrasing



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner,pause_predictor,phrase_maker"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <phrase>
    <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  </phrase>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <phrase>
    <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
    <token text=" " token_class="space"/>
    <token text="na" token_class="word" start="2560" end="2660">...</token>
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

Silences treated as proxy for prosodic phrase breaks, and phrasing structure added

Phrasing

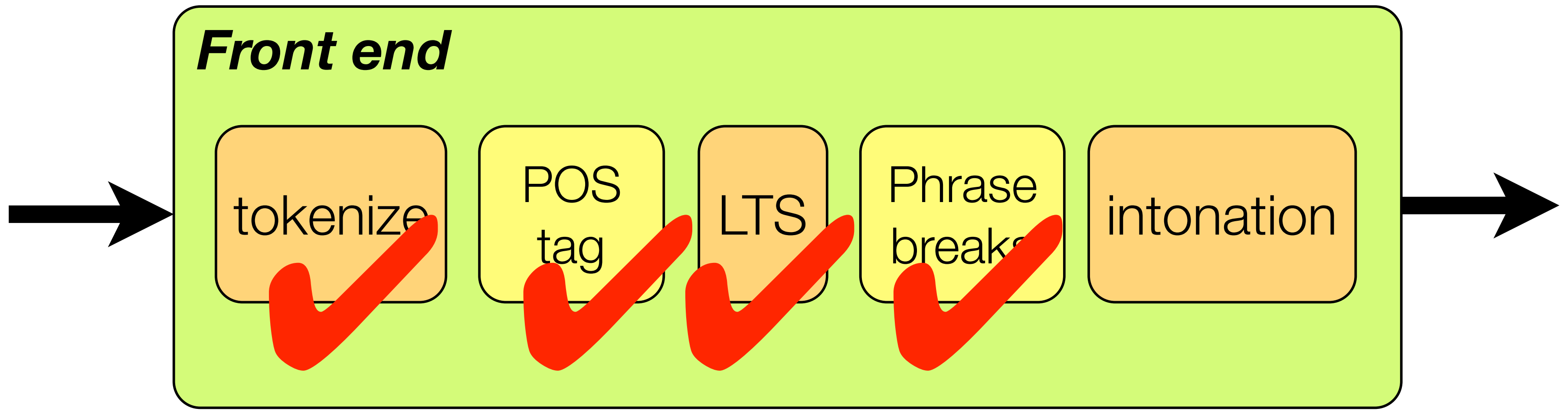


```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner,pause_predictor,phrase_maker"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <phrase>
    <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  </phrase>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <phrase>
    <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
    <token text=" " token_class="space"/>
    <token text="na" token_class="word" start="2560" end="2660">...</token>
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

Silences treated as proxy for prosodic phrase breaks, and phrasing structure added

Train a statistical model to predict breaks based on surrounding words' vectors and punctuation

Phrasing



```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner,pause_predictor,phrase_maker"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <phrase>
    <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  </phrase>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <phrase>
    <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
    <token text=" " token_class="space"/>
    <token text="na" token_class="word" start="2560" end="2660">...</token>
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

Silences treated as proxy for prosodic phrase breaks, and phrasing structure added

Train a statistical model to predict breaks based on surrounding words' vectors and punctuation

Linguistic feature engineering: flatten using XPATHS

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
processors_used=",word_splitter,segment_adder,feature_dumper,acoustic_feature_extractor,aligner,pause_predictor,phrase_maker"
acoustic_stream_names="mgc,lf0,bap" acoustic_stream_dims="60,1,5" start="0" end="4245">
  <token text="_END_" token_class="_END_" start="0" end="1115">...</token>
  <phrase>
    <token text="Khartoum" token_class="word" start="1115" end="1755">...</token>
  </phrase>
  <token text=" " token_class="space" start="1755" end="1860">
    <segment pronunciation="sil" start="1755" end="1860">...</segment>
  </token>
  <phrase>
    <token text="imejitenga" token_class="word" start="1860" end="2560">...</token>
    <token text=" " token_class="space"/>
    <token text="na" token_class="word" start="2560" end="2660">...</token>
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

Linguistic feature engineering: flatten using XPATHS

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
pro                                                                          se_maker"
a
```

```
l_segment = ./ancestor::segment/preceding::segment[1]/attribute::pronunciation      = o
c_segment = ./ancestor::segment/attribute::pronunciation                            = h
length_current_word = count(ancestor::token/descendant::segment)                    = 3
till_phrase_end_in_words = count_Xs_till_end_Y('token[@token_class=\"word\"]', 'phrase')  = 0
```

etc...

```
    <token text=" " token_class="space"/>
    <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
    <token text=" " token_class="space"/>
    <token text="huo" token_class="word" start="2975" end="3325">
      <segment pronunciation="h" start="2975" end="3000">
        <state start="2975" end="2980"/>
        <state start="2980" end="2985"/>
        <state start="2985" end="2990"/>
        <state start="2990" end="2995"/>
        <state start="2995" end="3000"/>
      </segment>
      <segment pronunciation="u" start="3000" end="3155">...</segment>
      <segment pronunciation="o" start="3155" end="3325">...</segment>
    </token>
  </phrase>
```

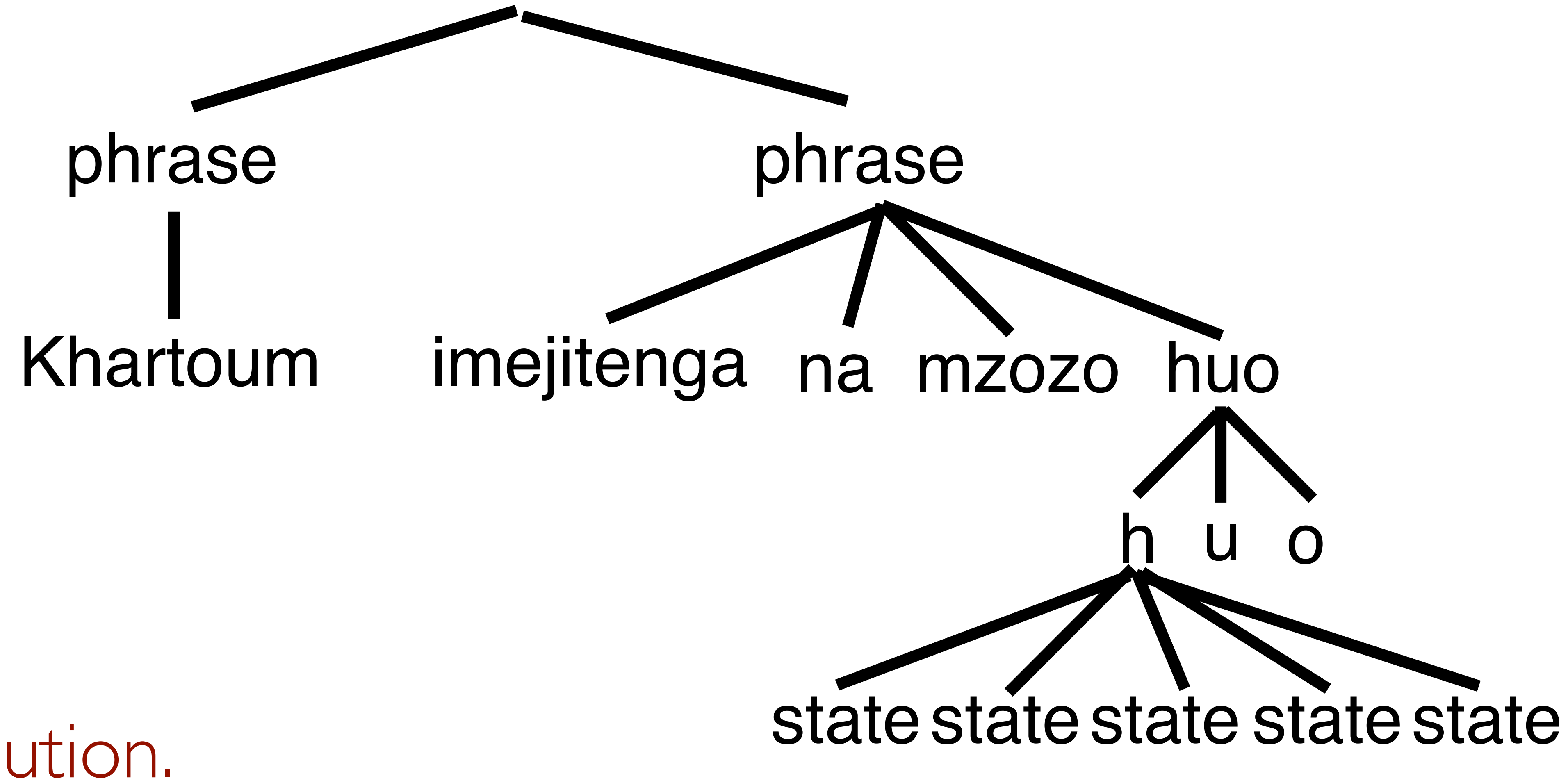
Linguistic feature engineering: flatten using XPATHS

```
<utt text="Khartoum imejitenga na mzozo huo." waveform="./wav/pm_n2236.wav"
utterance_name="pm_n2236"
pro                                                                       se_maker"
a
```

```
l_segment = ./ancestor::segment/preceding::segment[1]/attribute::pronunciation   = o
c_segment = ./ancestor::segment/attribute::pronunciation                          = h
length_current_word = count(ancestor::token/descendant::segment)                  = 3
till_phrase_end_in_words = count_Xs_till_end_Y('token[@token_class=\"word\"]', 'phrase')  = 0
```

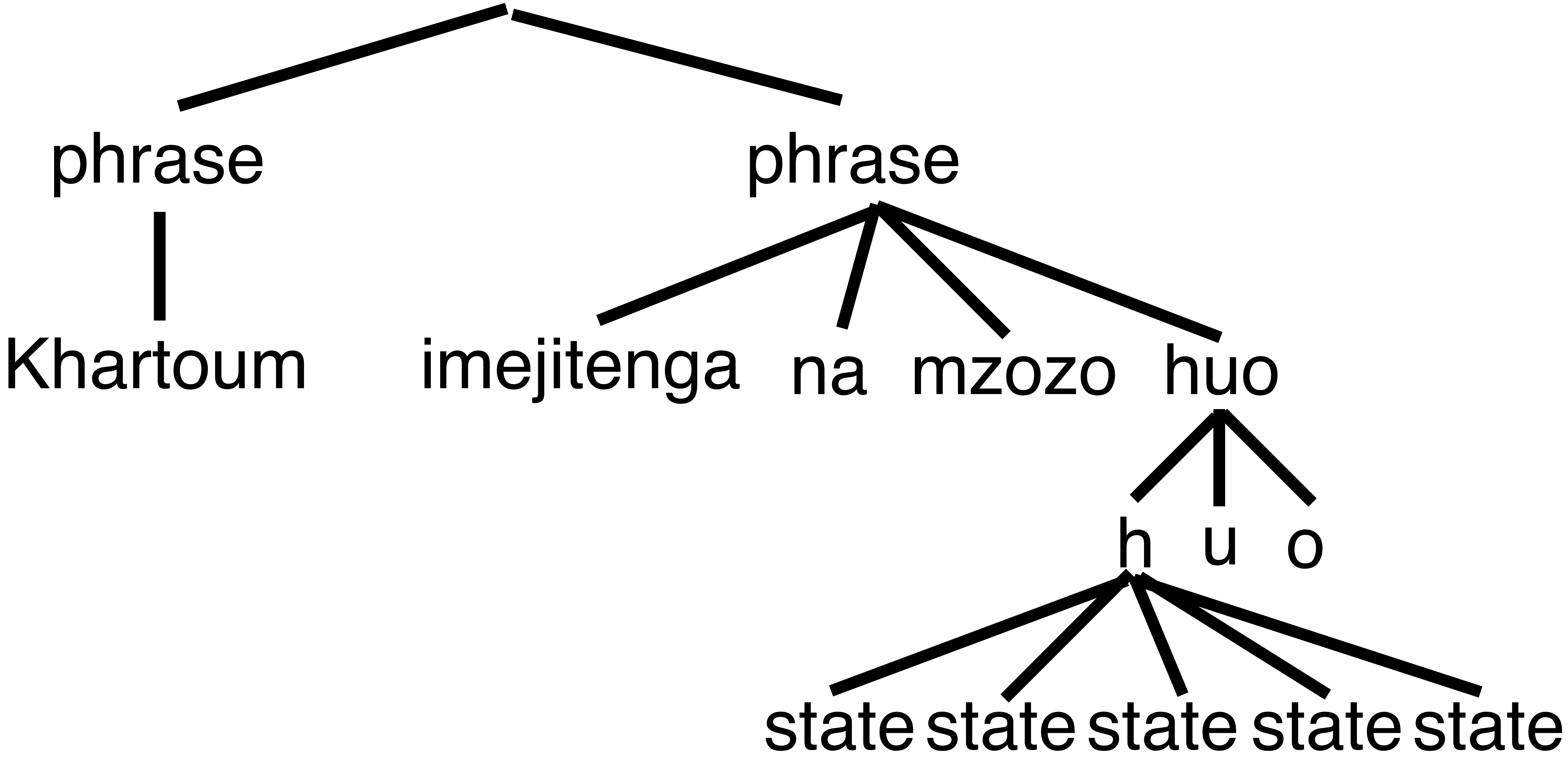
etc...

```
   <token text=" " token_class="space"/>
   <token text="mzozo" token_class="word" start="2660" end="2975">...</token>
   <token text=" " token_class="space"/>
   <token text="huo" token_class="word" start="2975" end="3325">
     <segment pronunciation="h" start="2975" end="3000">
       <state start="2975" end="2980"/>
       <state start="2980" end="2985"/>
       <state start="2985" end="2990"/>
       <state start="2990" end="2995"/>
       <state start="2995" end="3000"/>
     </segment>
     <segment pronunciation="u" start="3000" end="3155">...</segment>
     <segment pronunciation="o" start="3155" end="3325">...</segment>
   </token>
 </phrase>
```



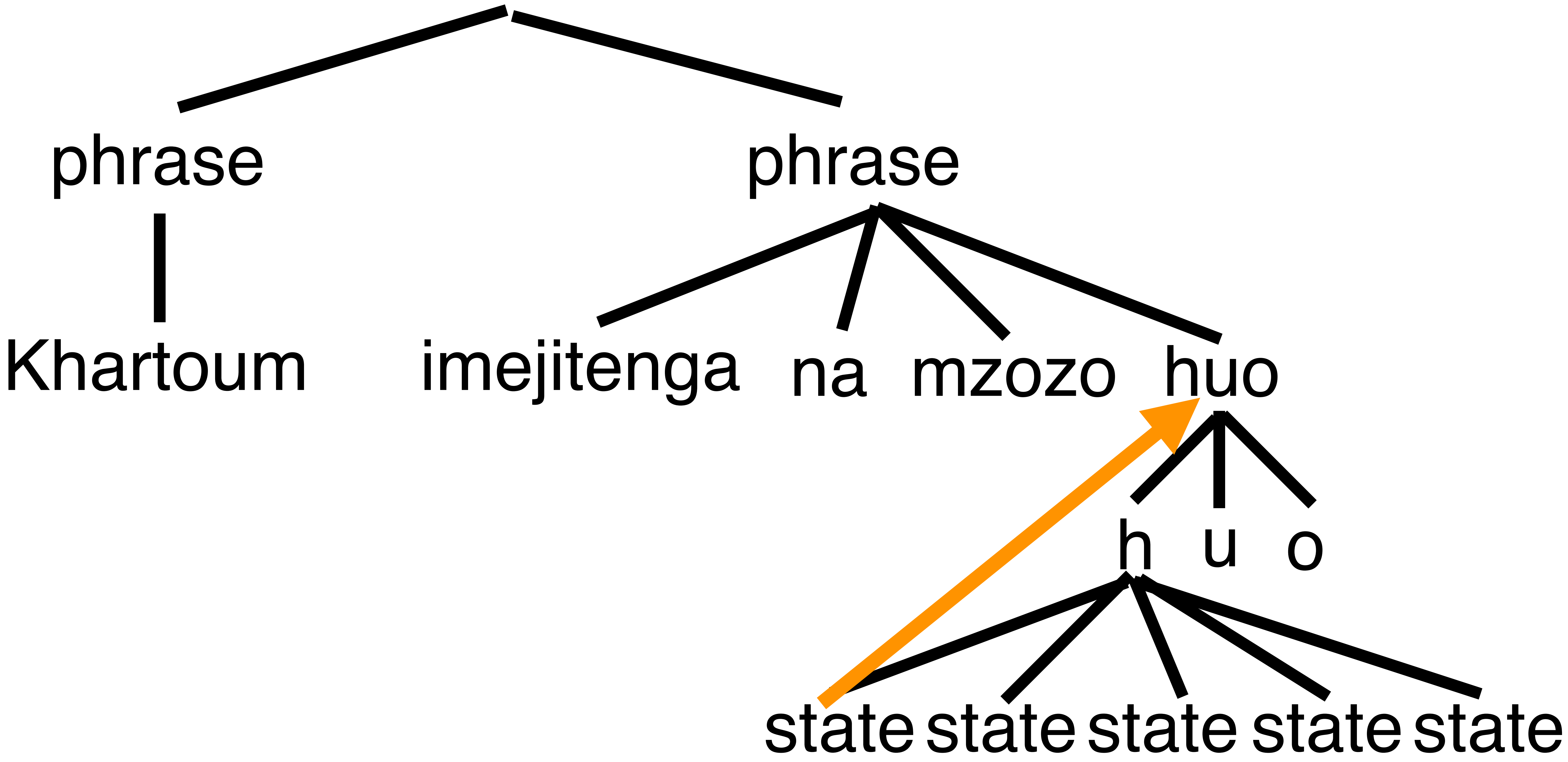
Linguistic feature engineering: flatten using XPATHS

```
l_segment = ./ancestor::segment/preceding::segment[1]/attribute::pronunciation          = o
c_segment = ./ancestor::segment/attribute::pronunciation                                = h
length_current_word = count(ancestor::token/descendant::segment)                        = 3
till_phrase_end_in_words = count_Xs_till_end_Y('token[@token_class=\"word\"]', 'phrase')  = 0
                                        etc...
```



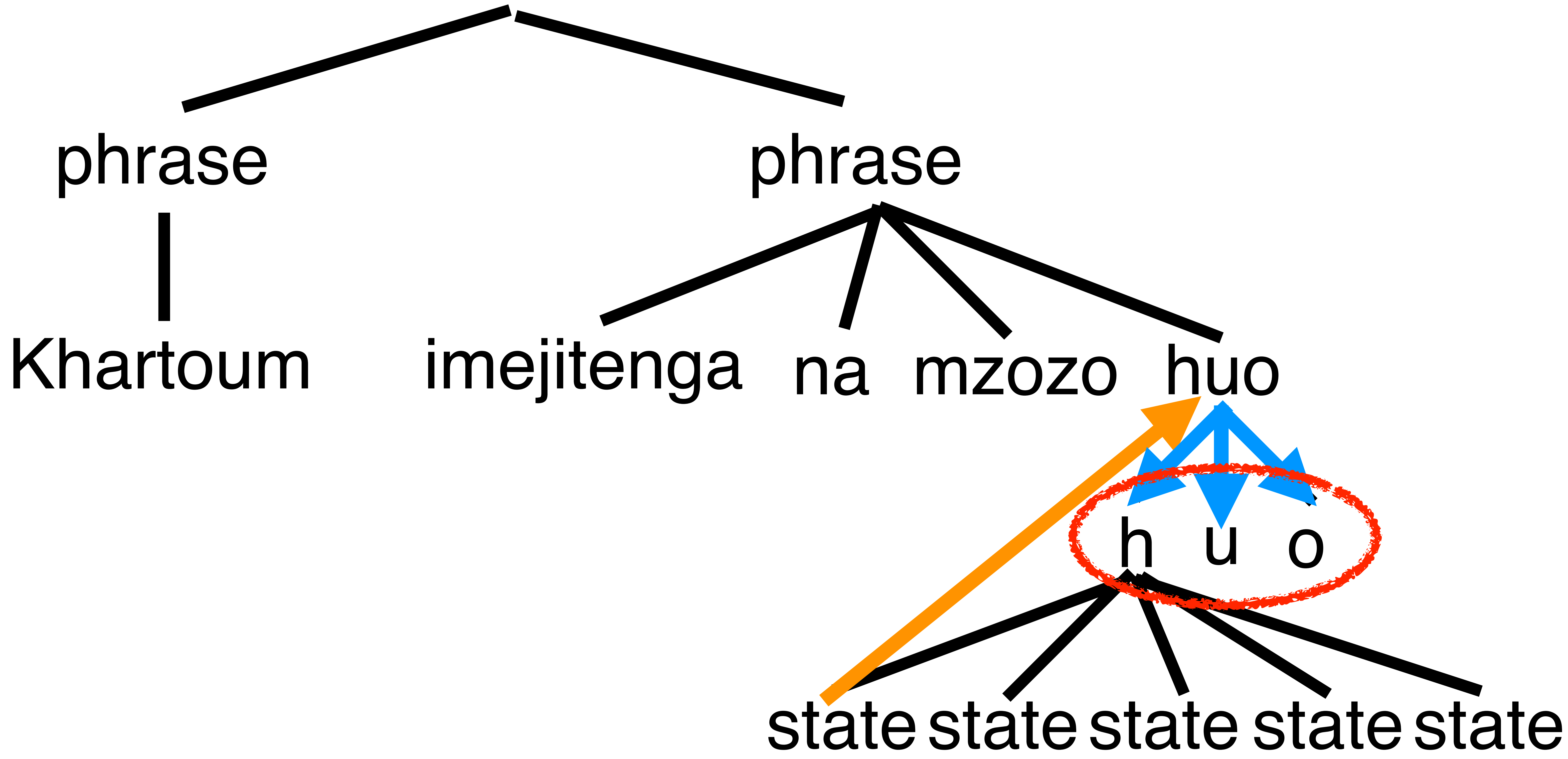
Linguistic feature engineering: flatten using XPATHS

```
l_segment = ./ancestor::segment/preceding::segment[1]/attribute::pronunciation         = o
c_segment = ./ancestor::segment/attribute::pronunciation                               = h
length_current_word = count(ancestor::token/descendant::segment)                       = 3
till_phrase_end_in_words = count_Xs_till_end_Y('token[@token_class=\"word\"]', 'phrase')  = 0
                                          etc...
```



Linguistic feature engineering: flatten using XPATHS

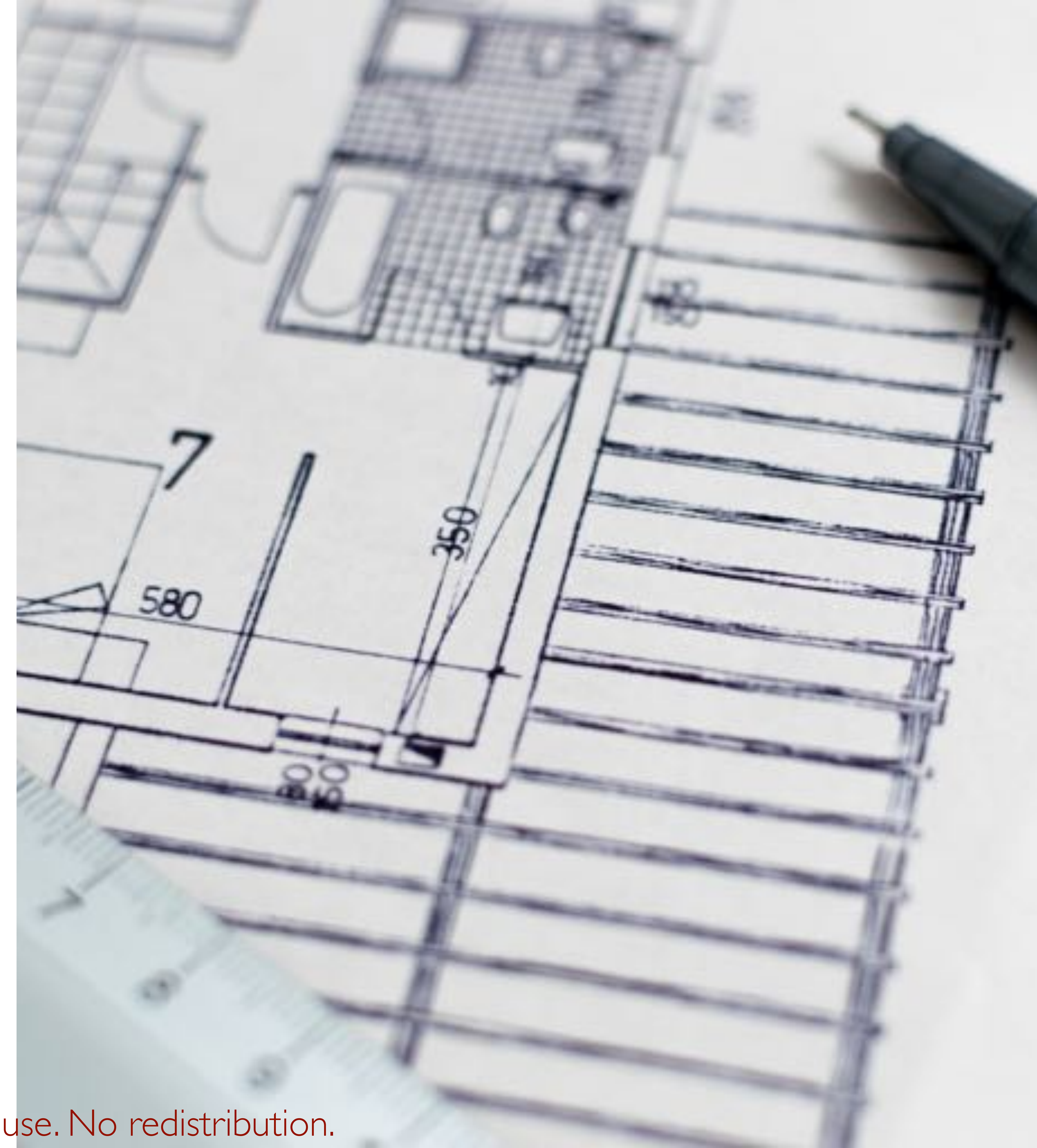
```
l_segment = ./ancestor::segment/preceding::segment[1]/attribute::pronunciation          = o
c_segment = ./ancestor::segment/attribute::pronunciation                                = h
length_current_word = count(ancestor::token/descendant::segment)                        = 3
till_phrase_end_in_words = count_Xs_till_end_Y('token[@token_class=\"word\"]', 'phrase') = 0
                                    etc...
```



Design choices: front end

- letters *or* phonemes *or* letter embeddings
- syllabification
- various choices for word vectors

- To improve this naive front end, add
  - text normalisation
  - letter-to-sound rules



Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**
- Input
  - linguistic features
- Output
  - acoustic features

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**
- Input
  - linguistic features
- Output
  - acoustic features

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**
- Input
  - linguistic features
- Output
  - acoustic features

can choose **any regression model** that we like, but first we need to prepare input & output features

to start with, let's assume the regression is performed

- **frame-by-frame**
- at **acoustic framerate**

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**

- Input
  - linguistic features

- Output
  - acoustic features

Requirements

- vector sequence
- at acoustic framerate
- aligned with acoustic features

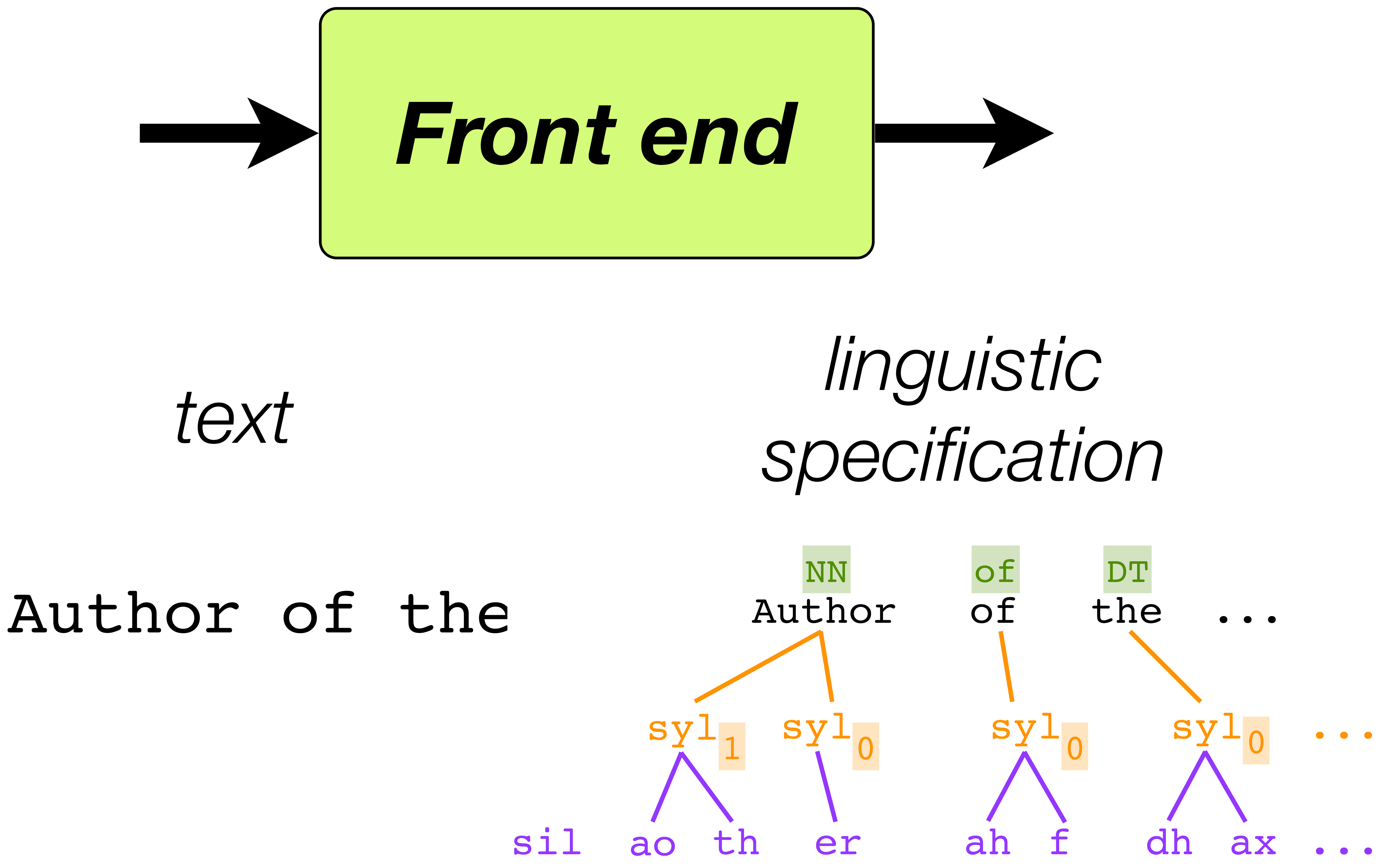
Agenda

	Topic	Presenter
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	**Srikanth Ronanki**
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

Linguistic feature extraction & engineering

Srikanth Ronanki

Feature extraction + feature engineering



Front end

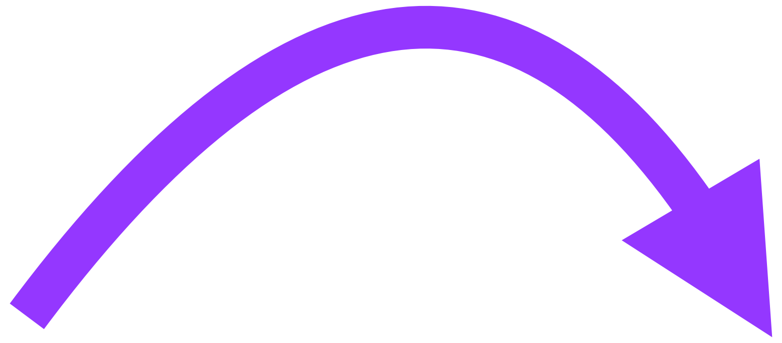
text

linguistic specification

`Author of the`

Feature extraction + feature engineering

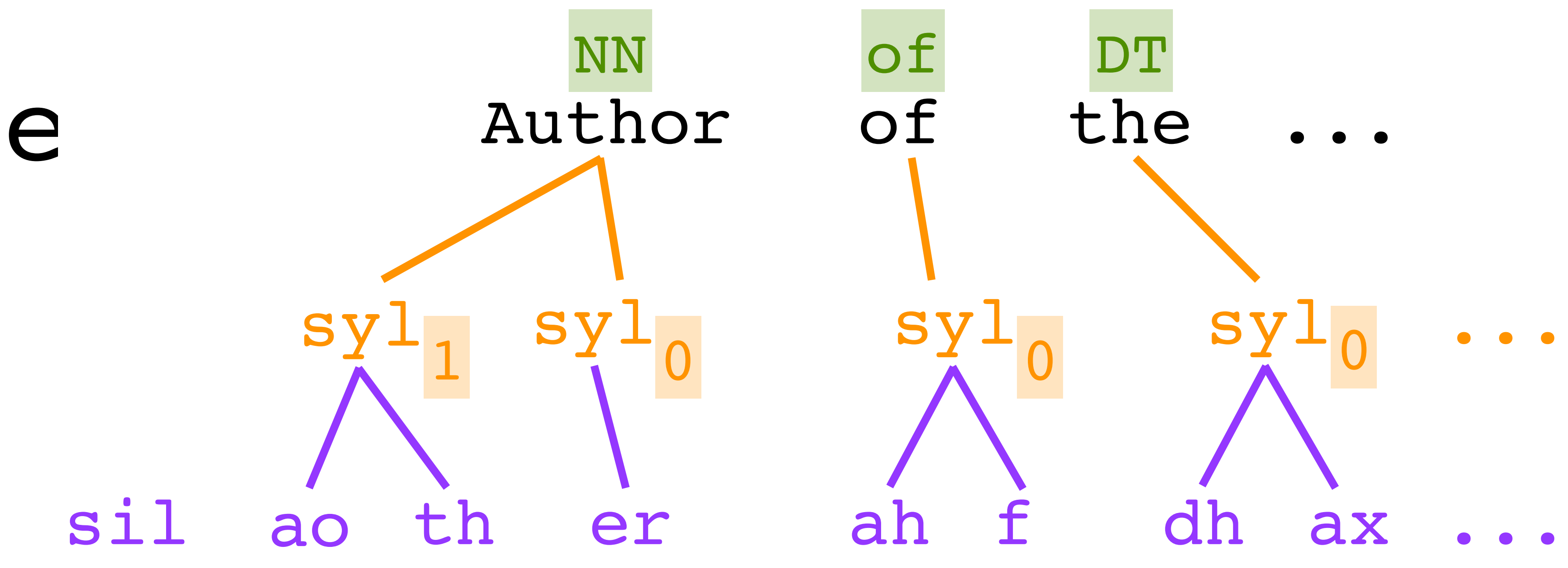
feature extraction



text

linguistic specification

Author of the



NN Author

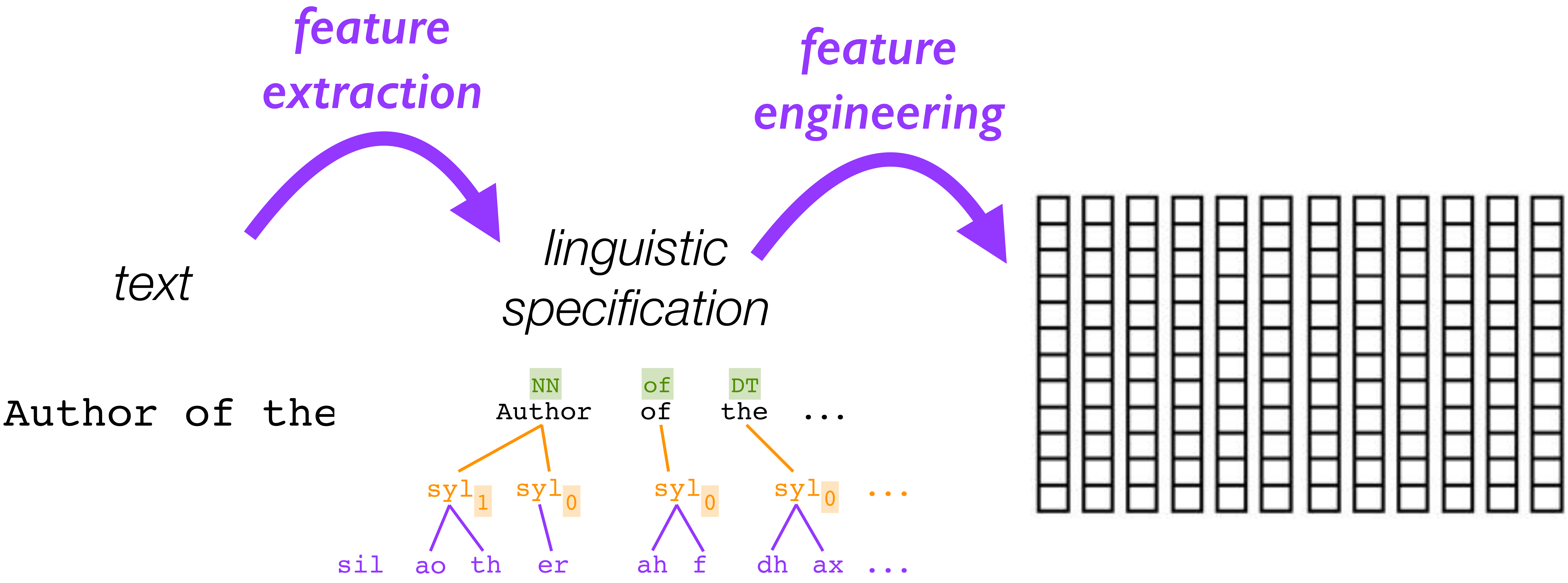
of of

DT the ...

syl 1 syl 0 syl 0 syl 0 ...

sil ao th er ah f dh ax ...

Feature extraction + feature engineering



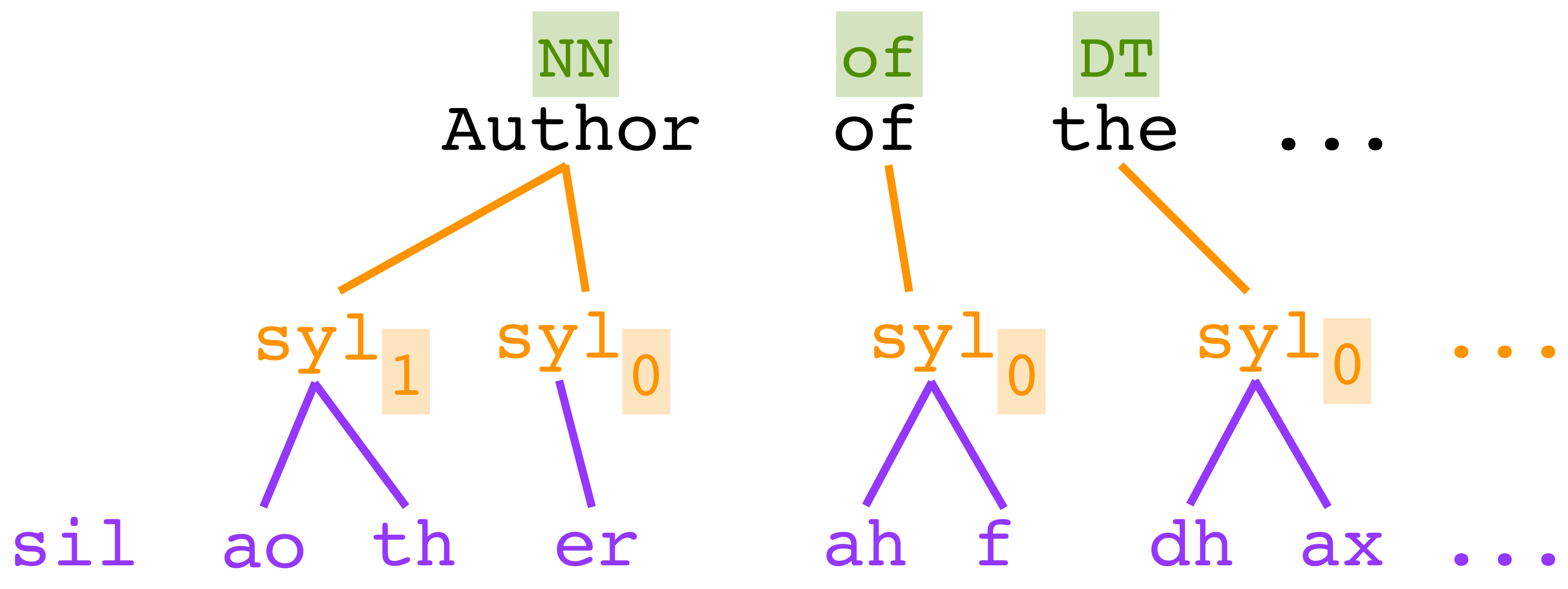
text

feature extraction

linguistic specification

feature engineering

Linguistic feature engineering

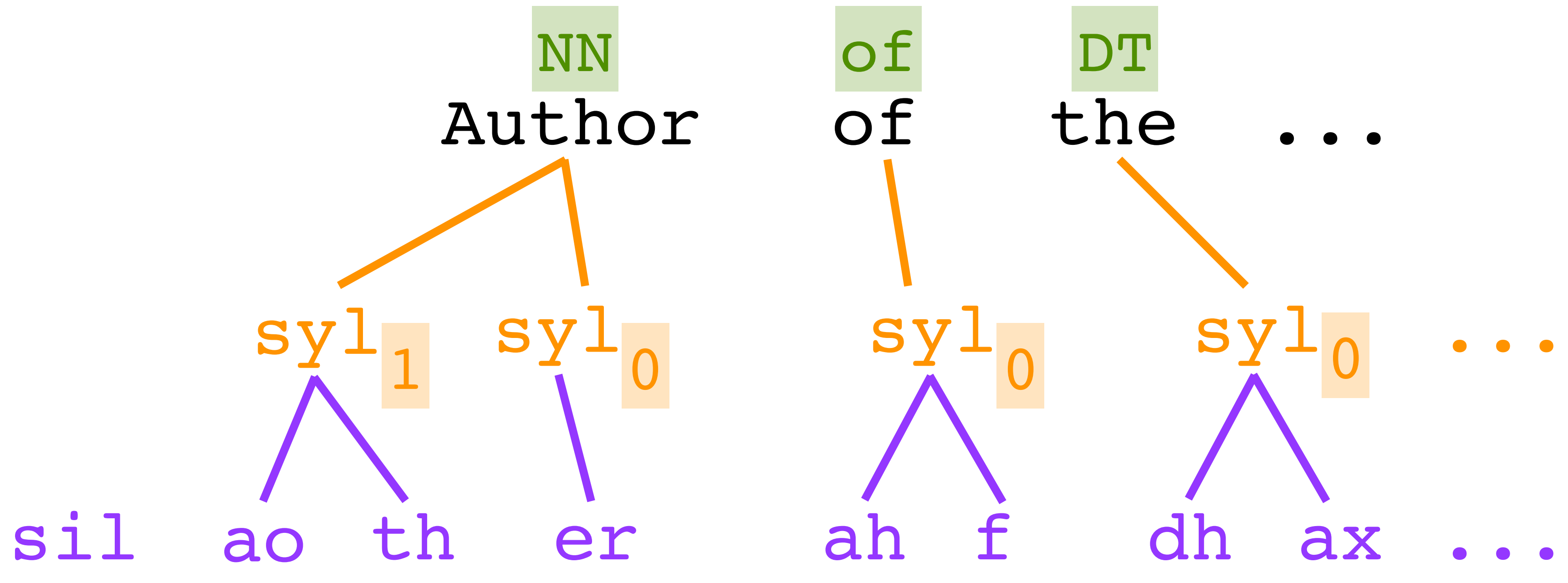


```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...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

- Run the front end
  - obtain linguistic specification

Linguistic feature engineering



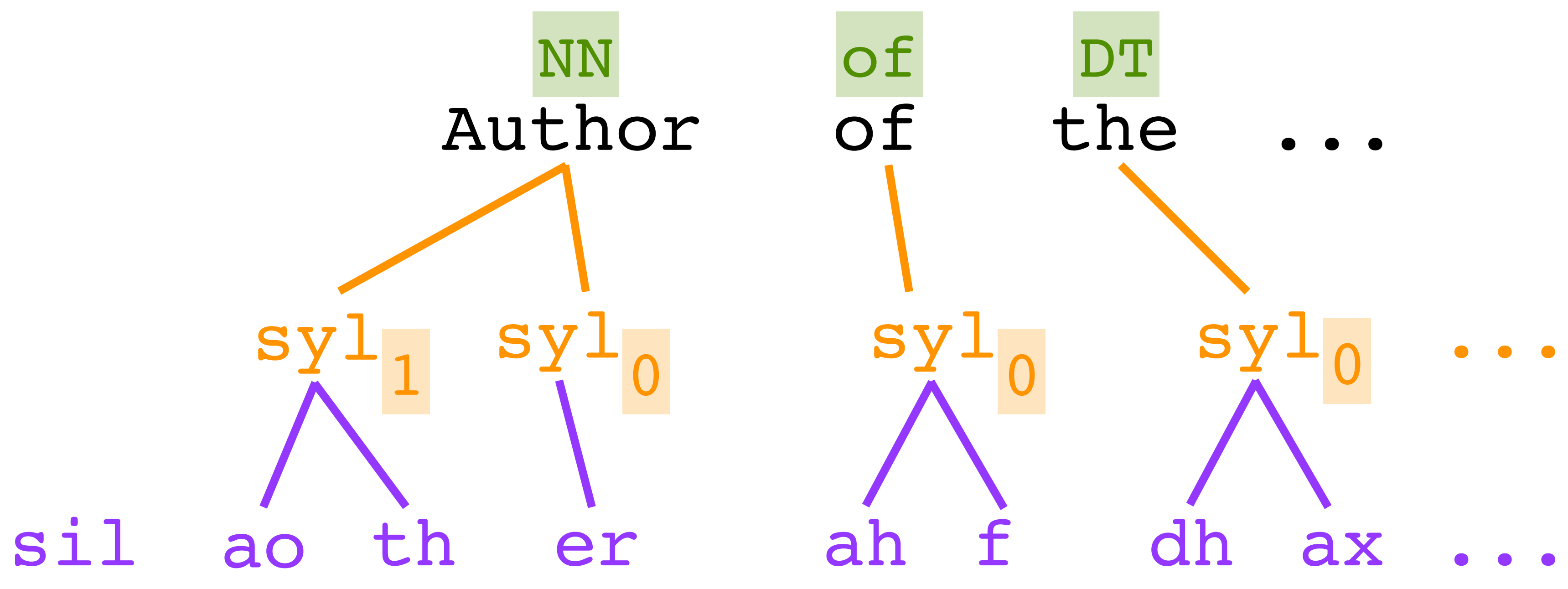
```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...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

- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

Sequence of context-dependent phones

Linguistic feature engineering



```
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$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

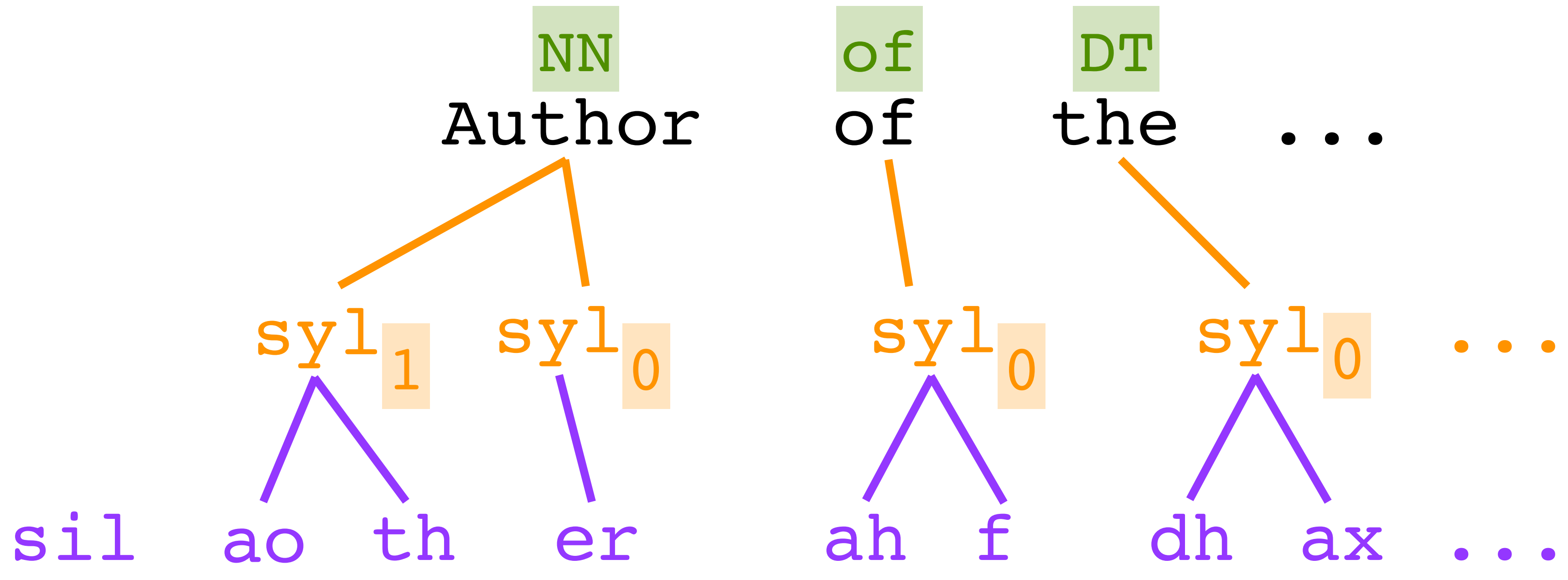
```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]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

- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

Sequence of context-dependent phones

- Encode as mostly-binary features

Linguistic feature engineering



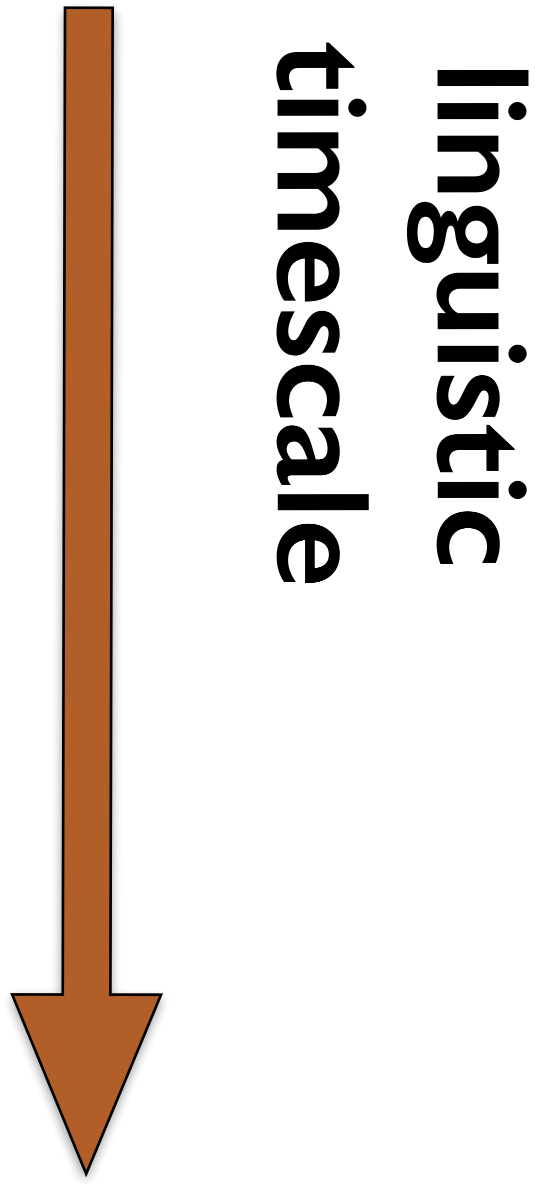
```
sil~sil-sil+ao=th@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$...
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
ao~th-er+ah=v@1_1/A:1_1_2/B:0-0-1@2-1&2-6#1-4$...
th~er-ah+v=dh@1_2/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...
er~ah-v+dh=ax@2_1/A:0_0_1/B:1-0-2@1-1&3-5#1-3$...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

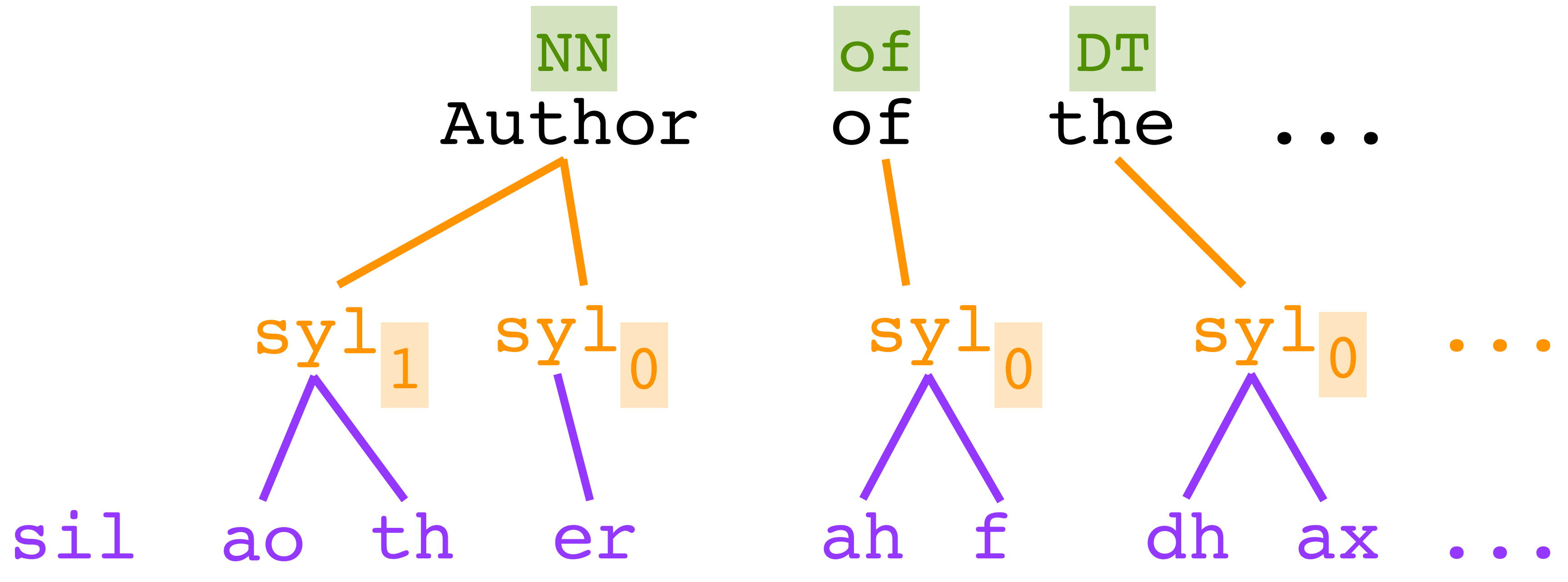
- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

Sequence of context-dependent phones

- Encode as mostly-binary features



Linguistic feature engineering



```
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$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

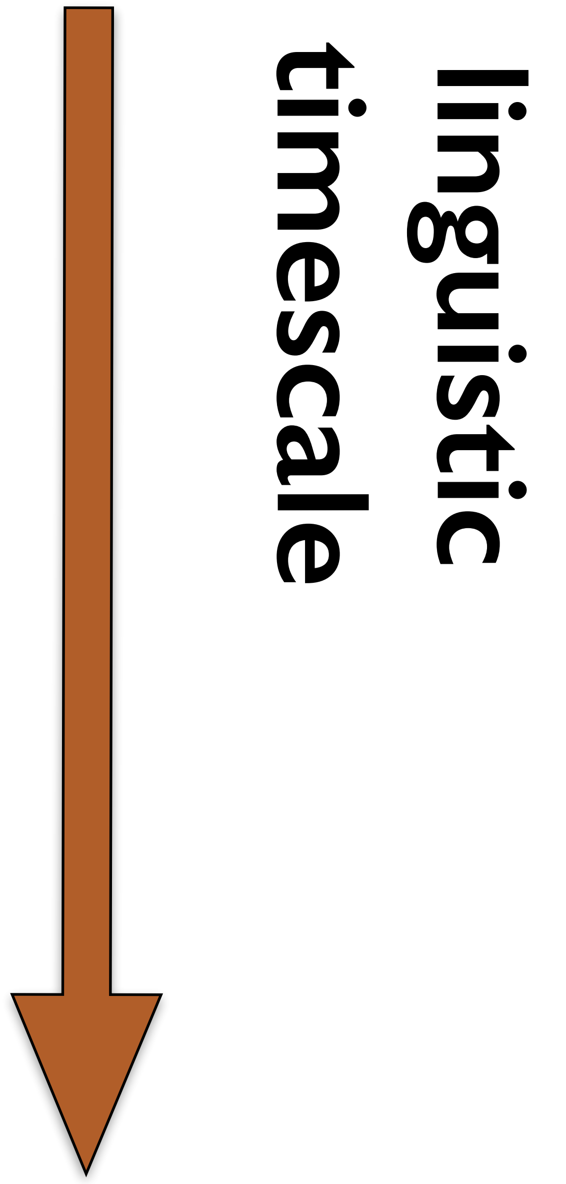
```
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2  0.1]
…
[0 0 1 0 0 1 0 1 1 0 … 0.2  1.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.4  0.5]
[0 0 1 0 0 1 0 1 1 0 … 0.4  1.0]
…
[0 0 1 0 0 1 0 1 1 0 … 1.0  1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.2  0.2]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

- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

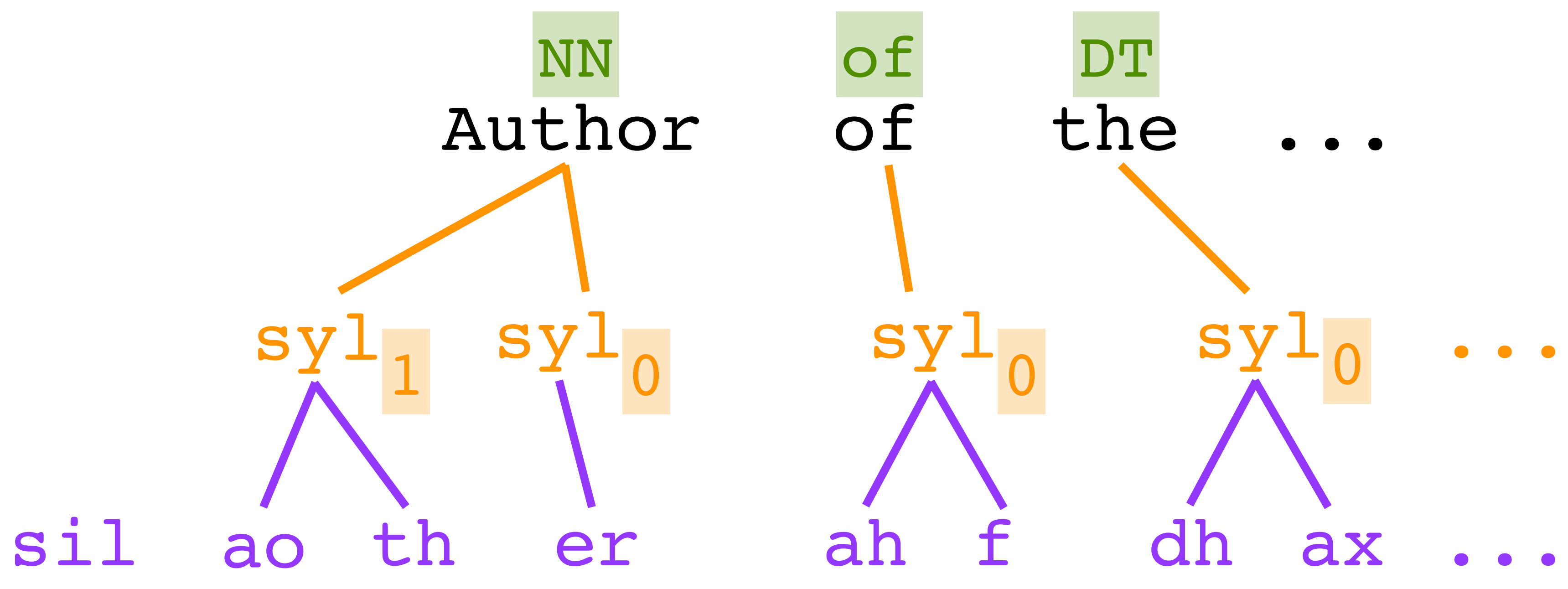
Sequence of context-dependent phones

- Encode as mostly-binary features
- Upsample using duration information

Frame sequence



Linguistic feature engineering



```
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]
…
```

- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

Sequence of context-dependent phones

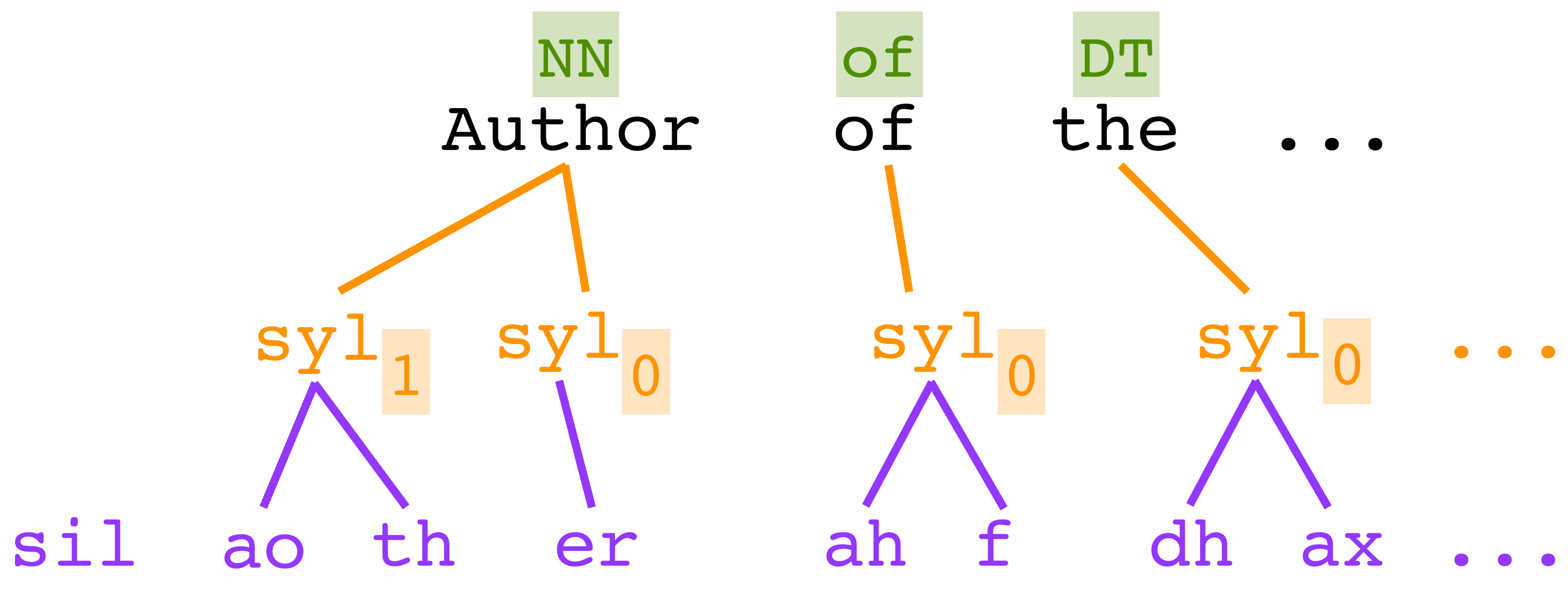
- Encode as mostly-binary features
- Upsample using duration information

Frame sequence

linguistic timescale

time is now at a fixed framerate

Linguistic feature engineering



```
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]
…
```

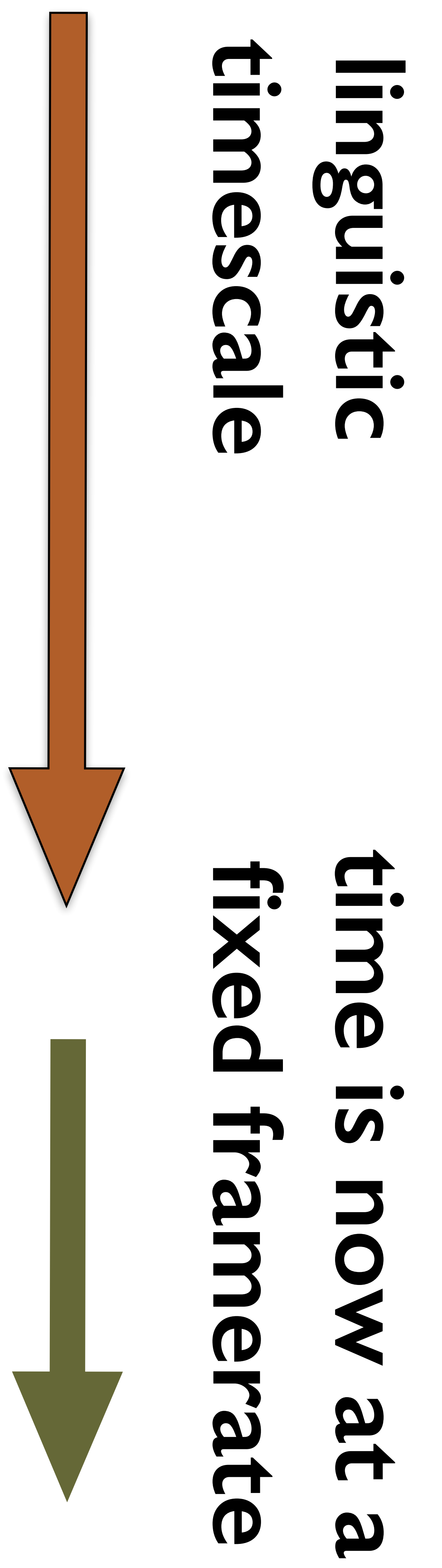
- Run the front end
  - obtain linguistic specification
- Flatten linguistic specification
  - attach contextual information to phones

Sequence of context-dependent phones

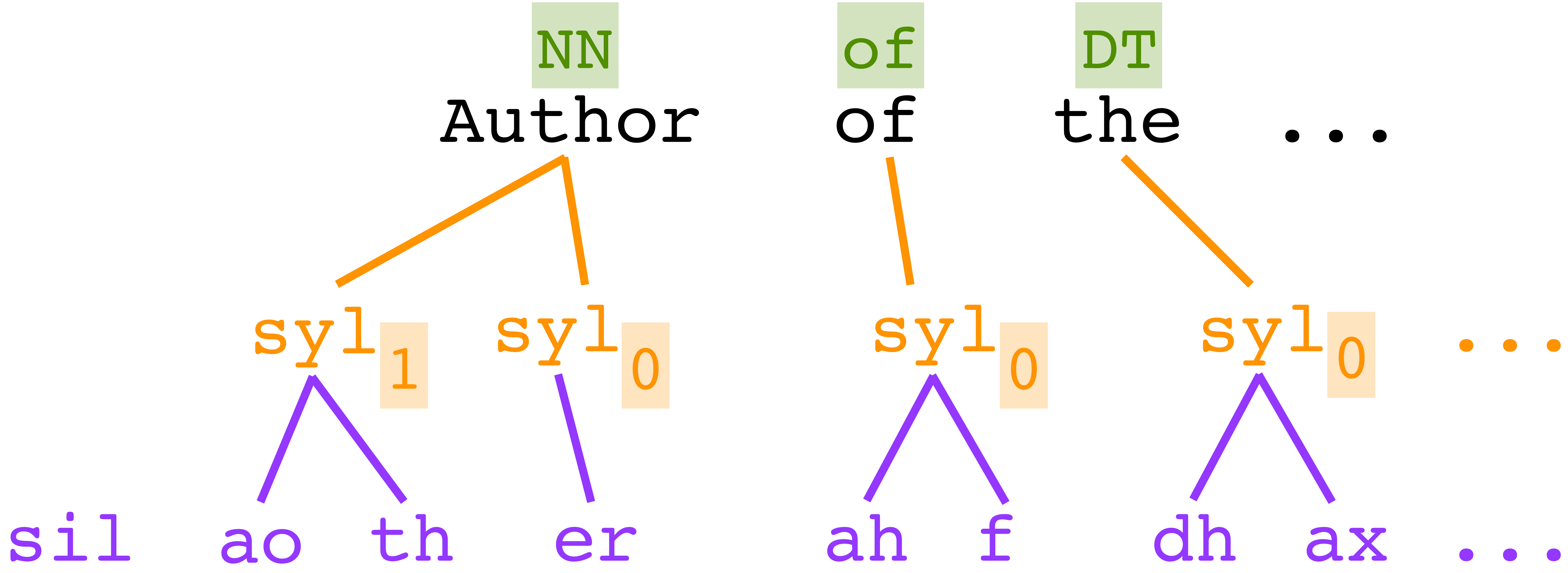
- Encode as mostly-binary features
- Upsample using duration information

Frame sequence

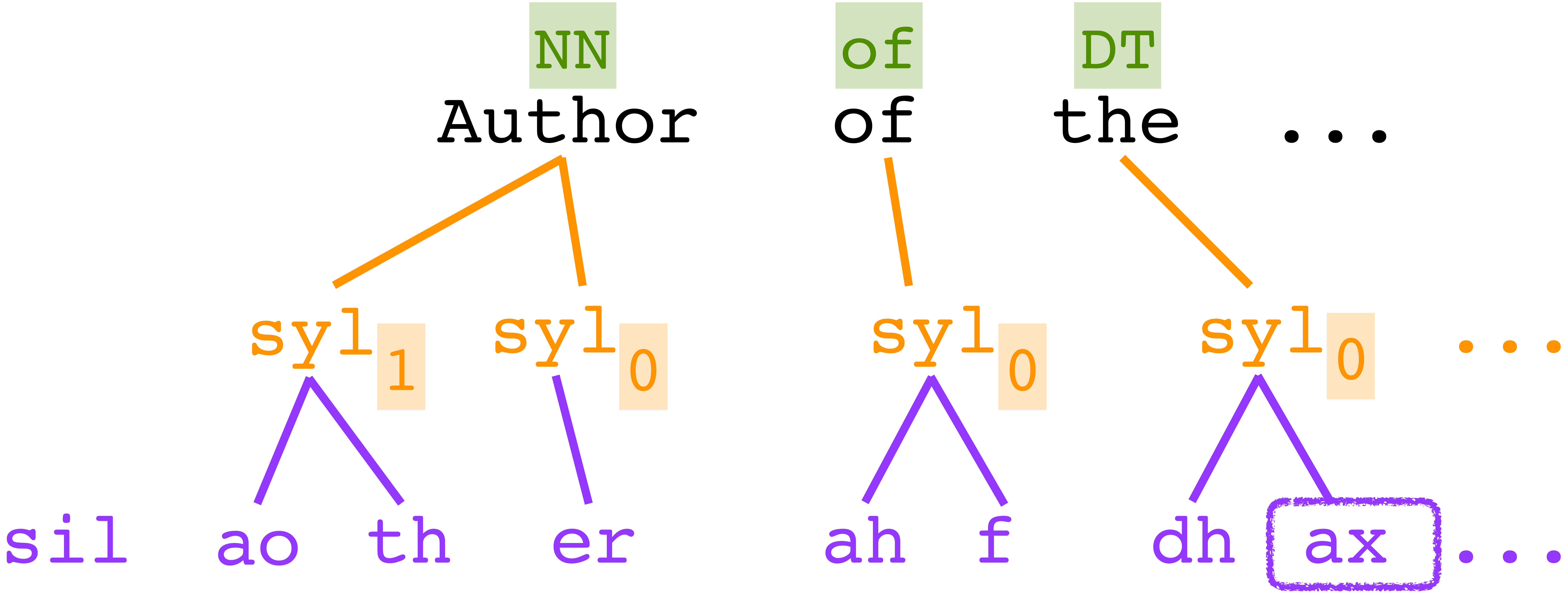
- Add fine-grained positional information



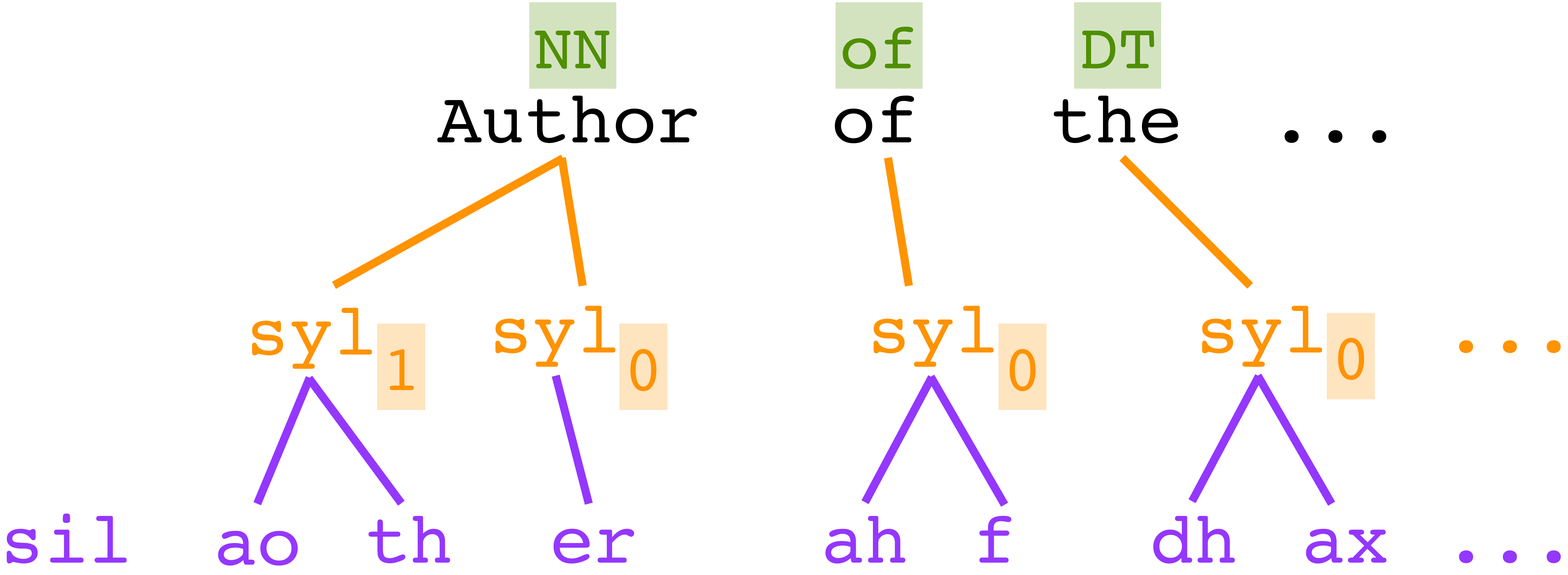
Linguistic feature engineering: flatten to context-dependent phones



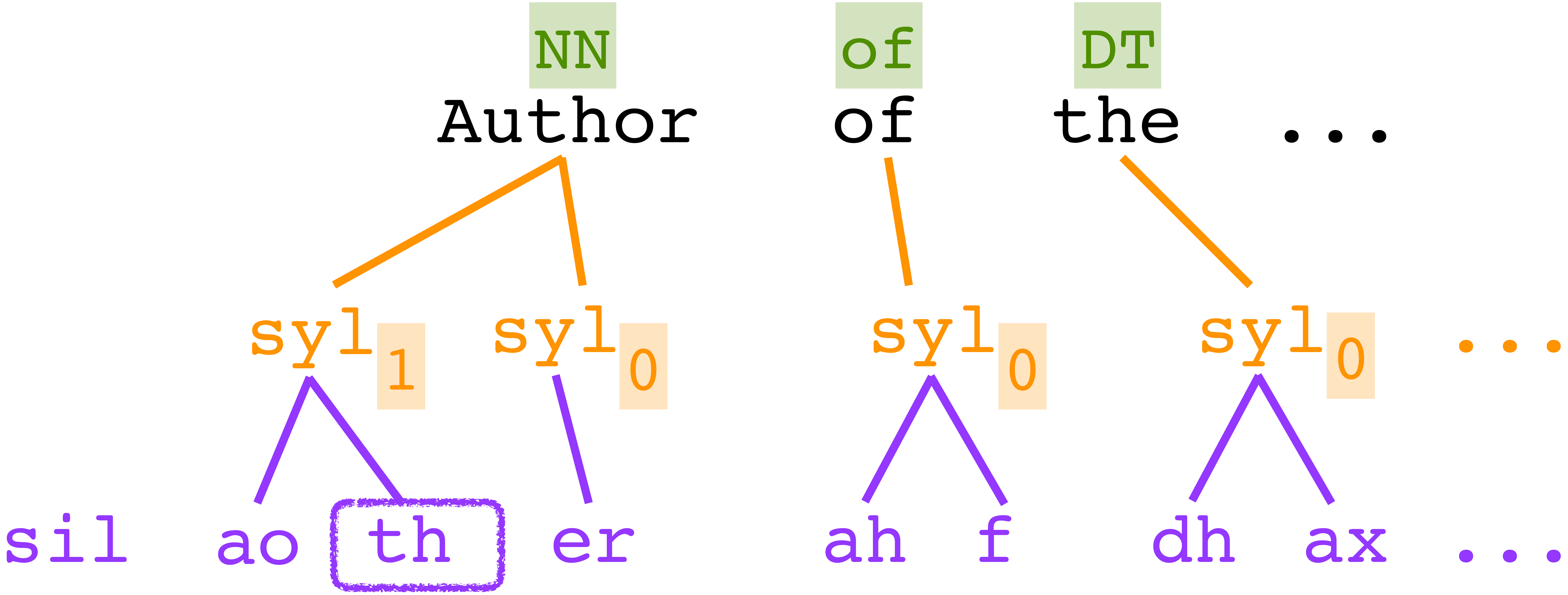
Linguistic feature engineering: flatten to context-dependent phones



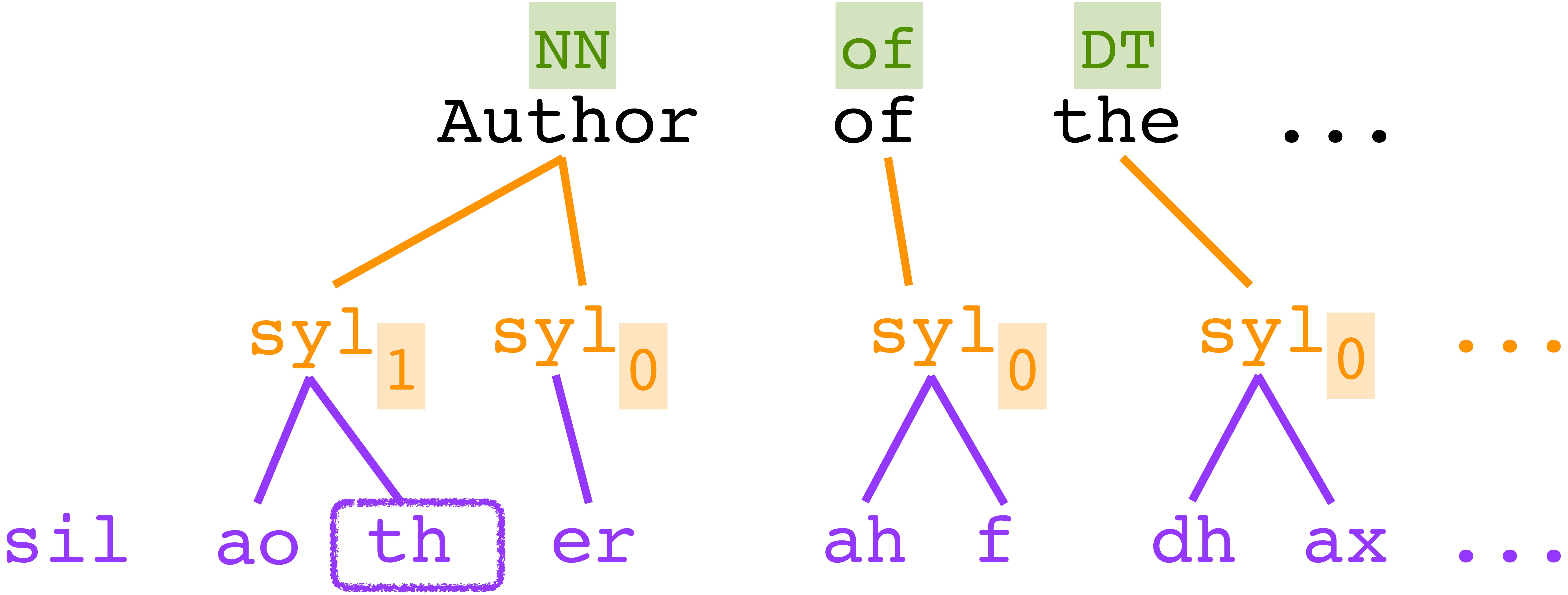
Linguistic feature engineering: flatten to context-dependent phones



Linguistic feature engineering: flatten to context-dependent phones

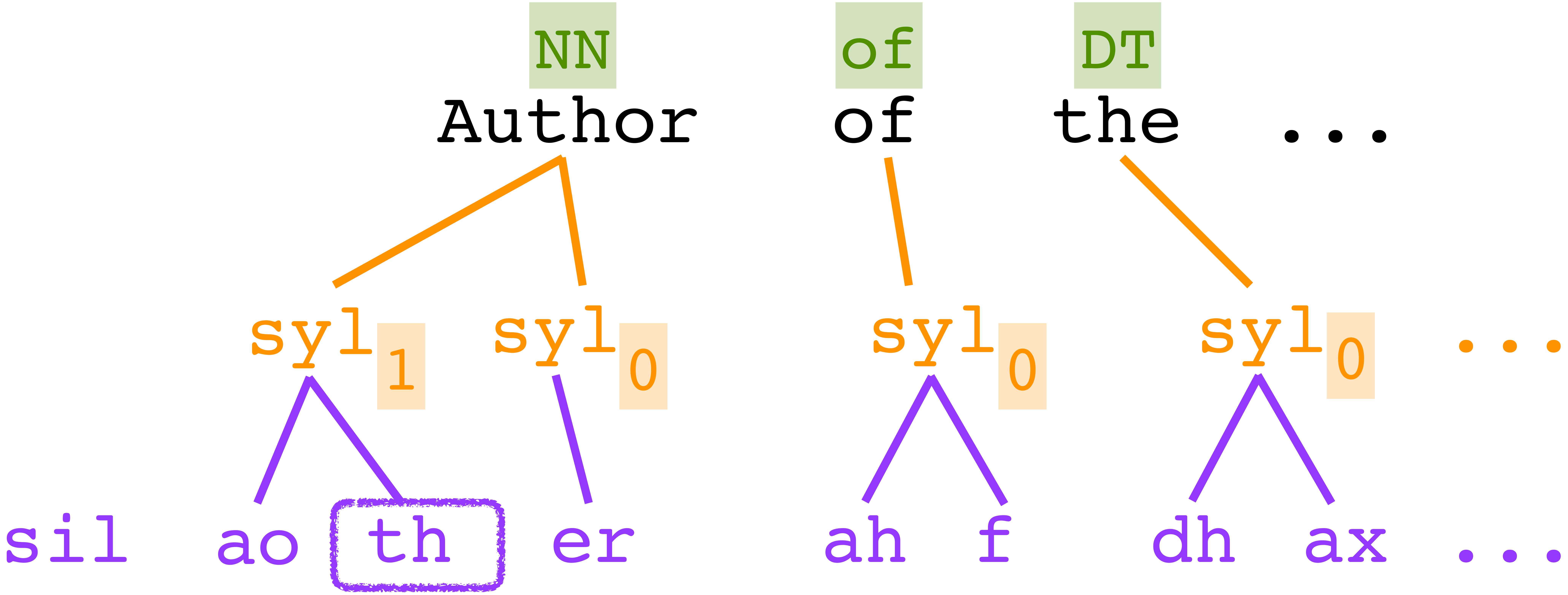


Linguistic feature engineering: flatten to context-dependent phones



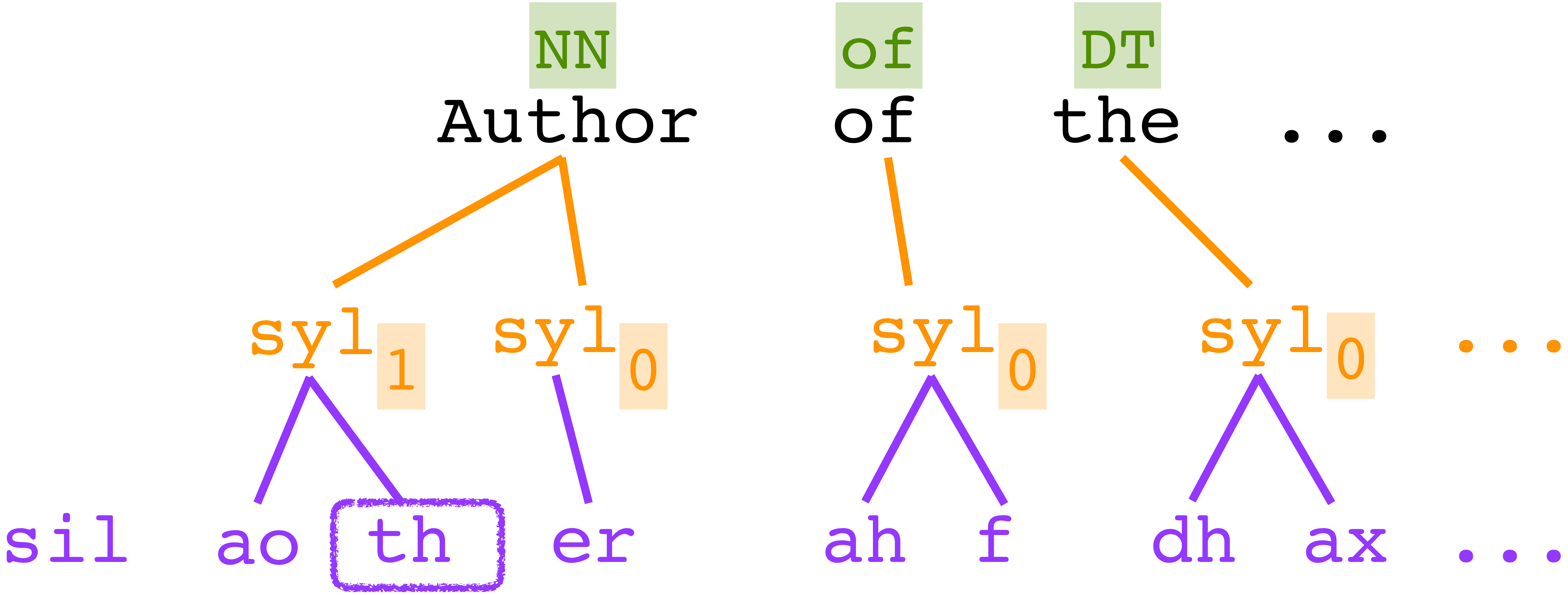
th

Linguistic feature engineering: flatten to context-dependent phones



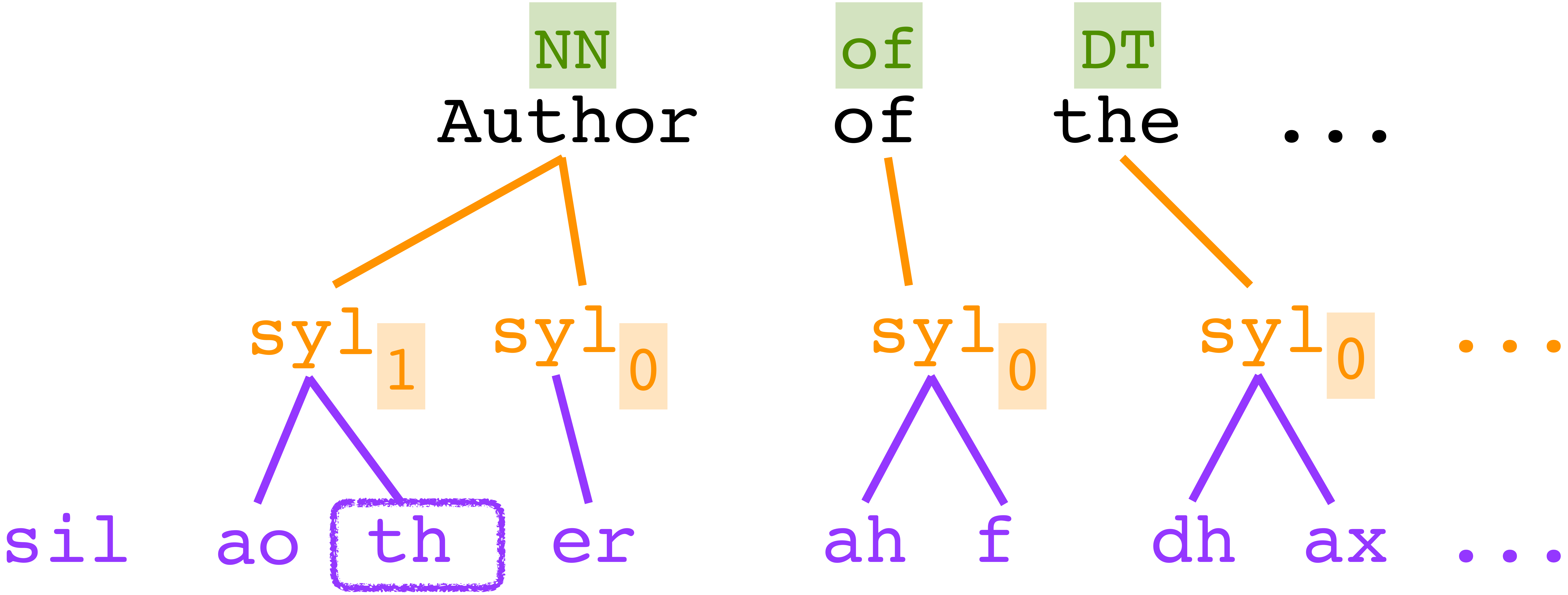
ao-th+er

Linguistic feature engineering: flatten to context-dependent phones



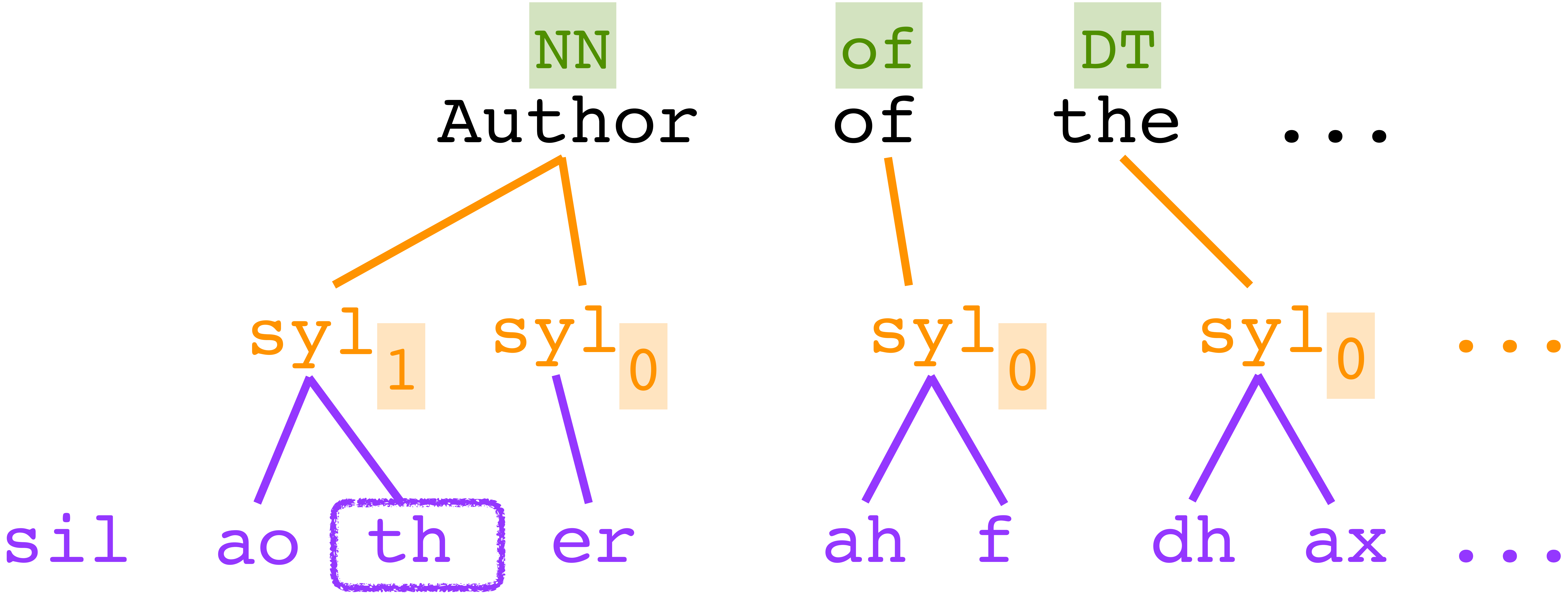
sil~ao-th+er=ah

Linguistic feature engineering: flatten to context-dependent phones



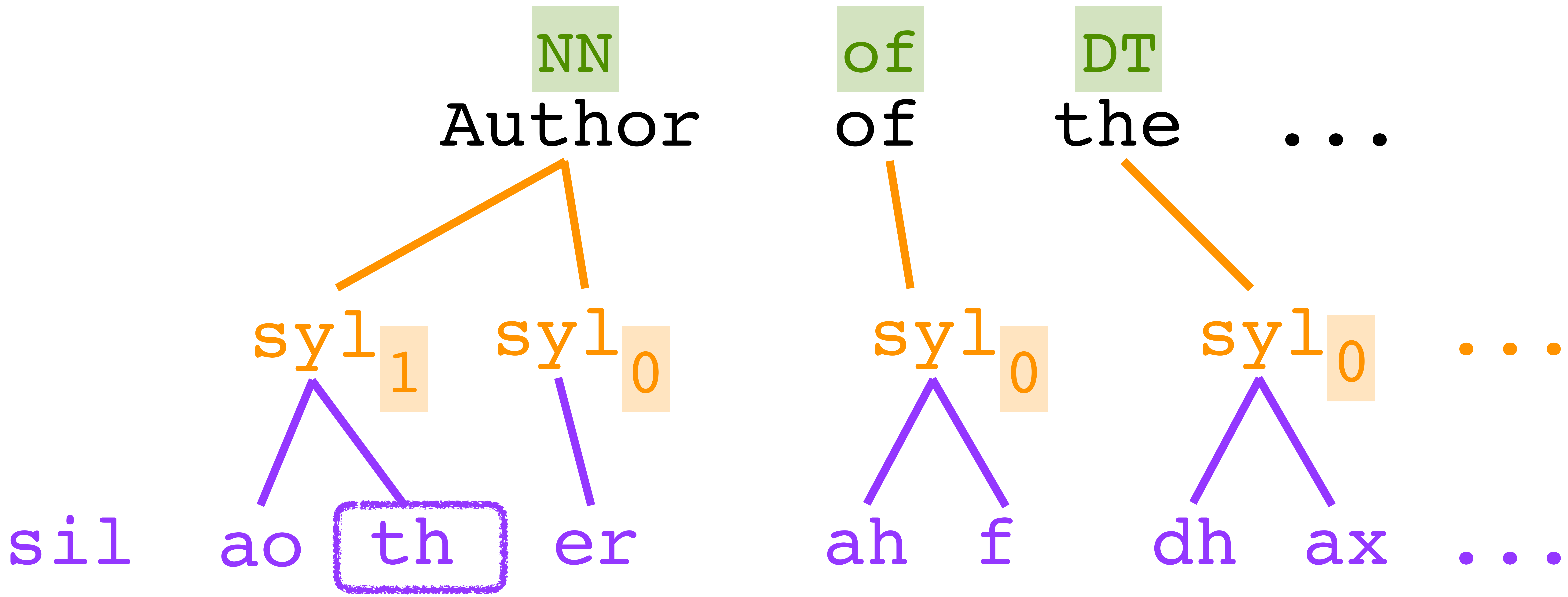
sil~ao-th+er=ah@

Linguistic feature engineering: flatten to context-dependent phones



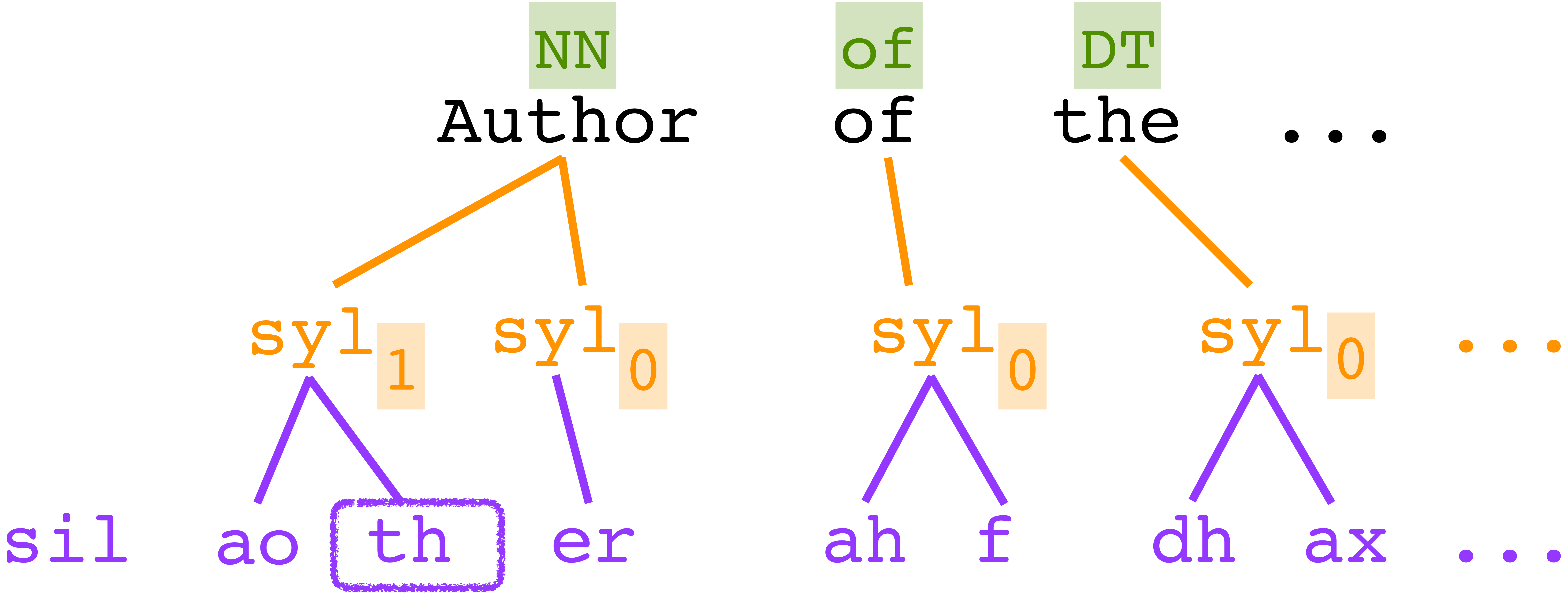
sil~ao-th+er=ah@2_2

Linguistic feature engineering: flatten to context-dependent phones



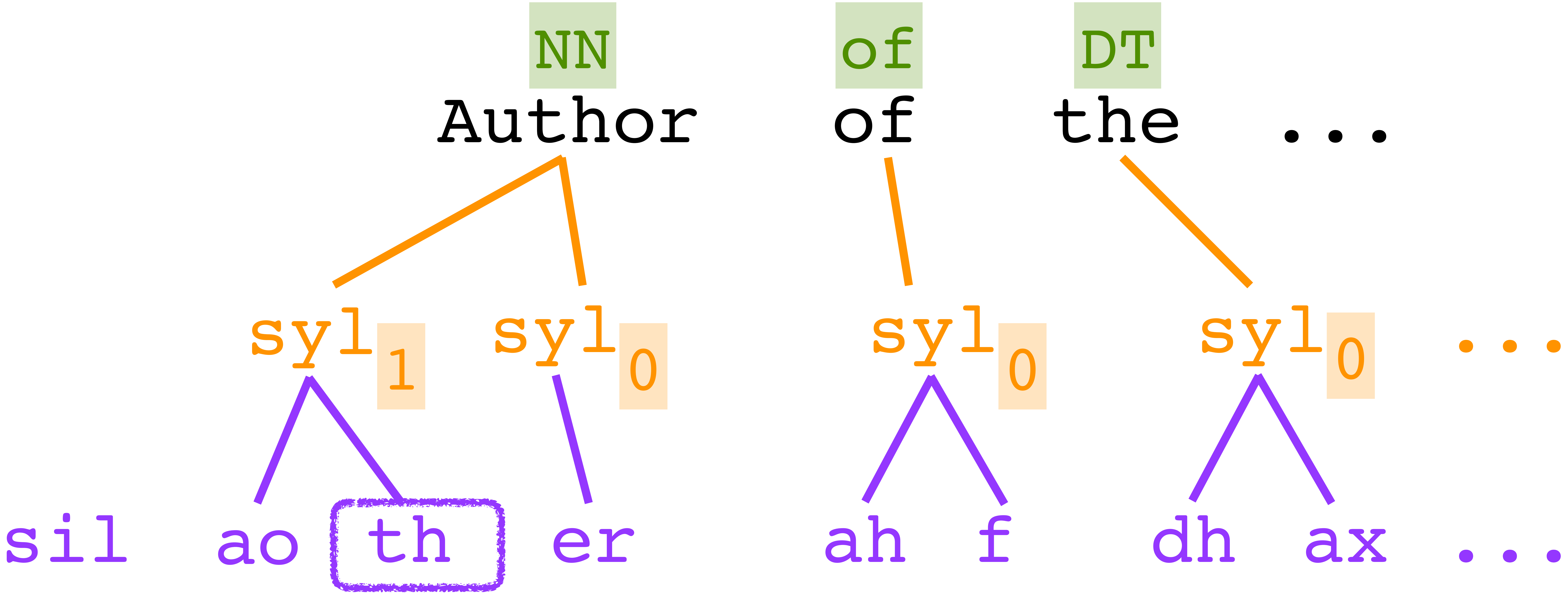
sil~ao-th+er=ah@2_2/A:0_0_0

Linguistic feature engineering: flatten to context-dependent phones



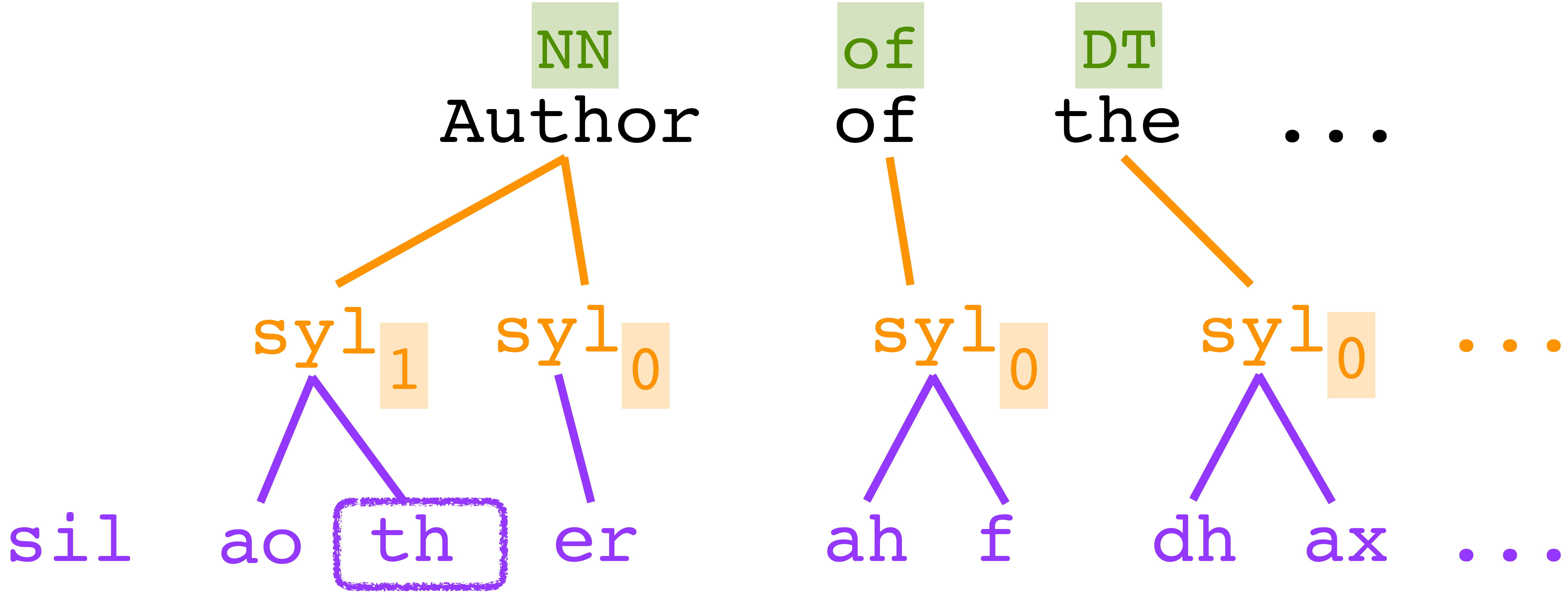
sil~ao-th+er=ah@2_2/A:0_0_0/B:1-1-2

Linguistic feature engineering: flatten to context-dependent phones



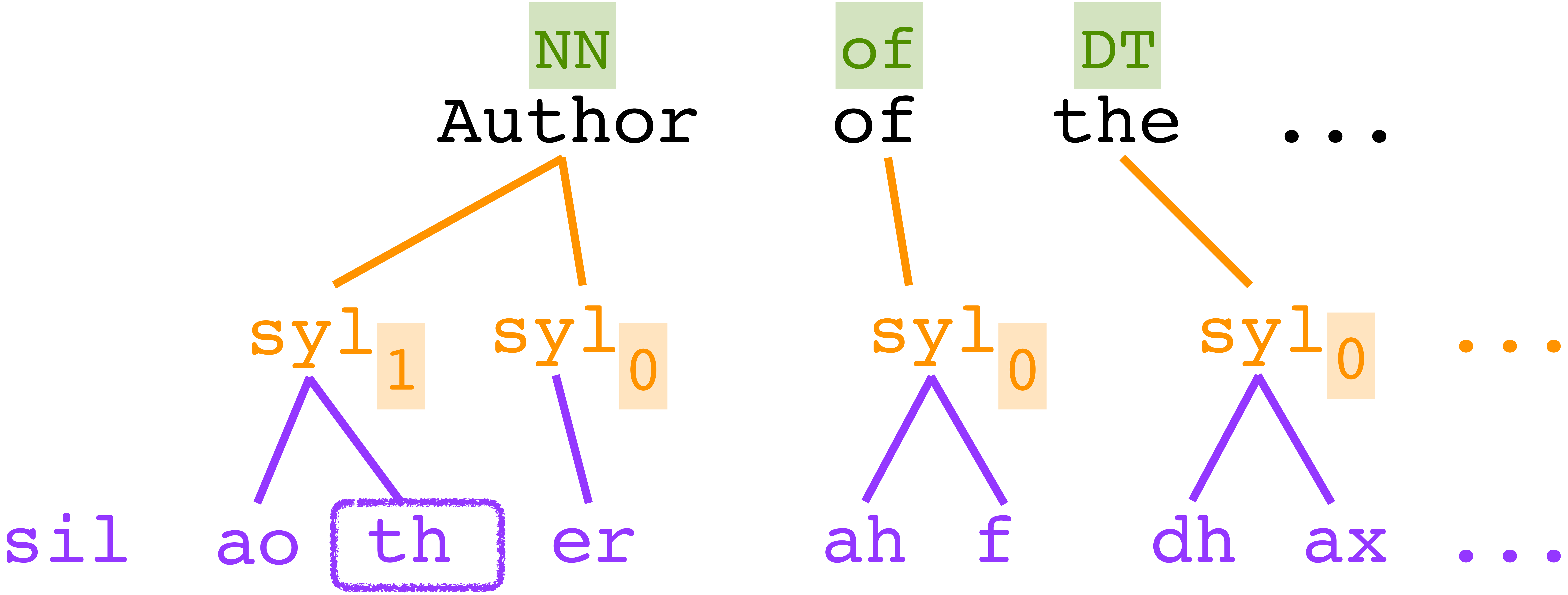
sil~ao-th+er=ah@2_2/A:0_0_0/B:1-1-2@1-2

Linguistic feature engineering: flatten to context-dependent phones



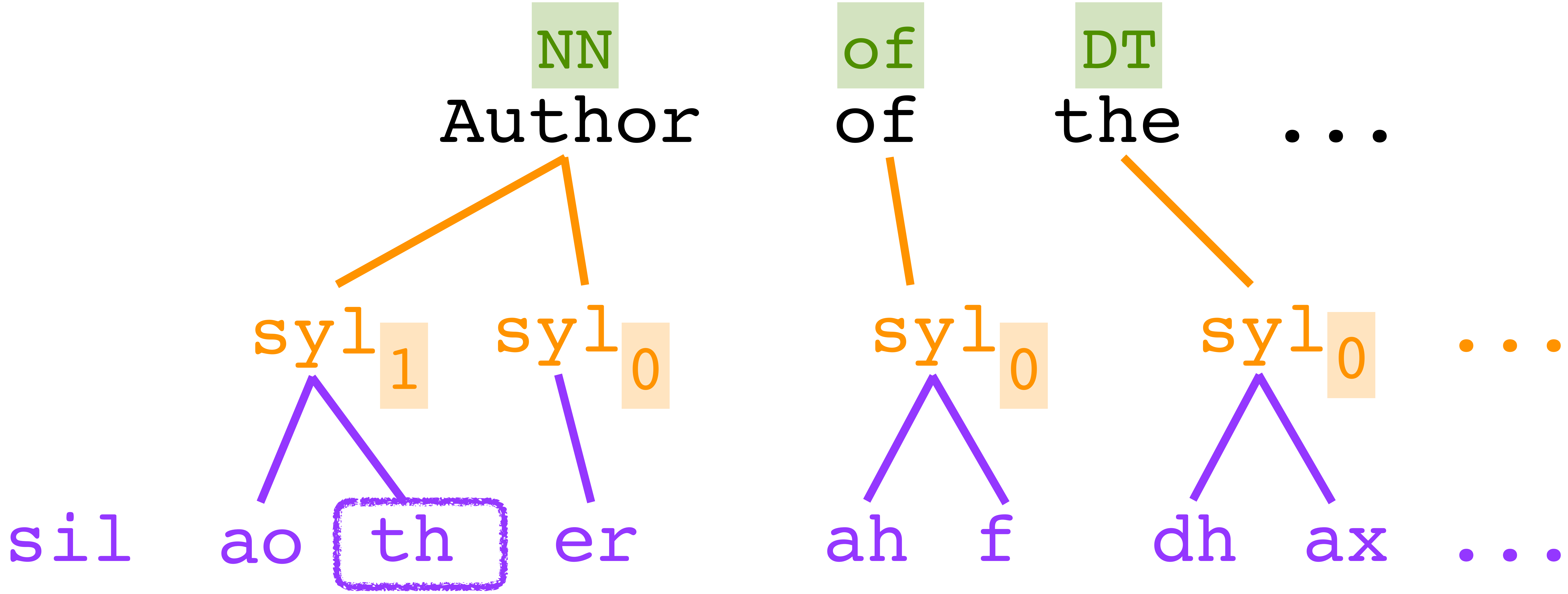
sil~ao-th+er=ah@2_2/A:0_0_0/B:1-1-2@1-2&1-7

Linguistic feature engineering: flatten to context-dependent phones



sil~ao-th+er=ah@2_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4

Linguistic feature engineering: flatten to context-dependent phones



sil~ao-th+er=ah@2_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...

Anatomy of a context-dependent phone

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

Anatomy of a context-dependent phone

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

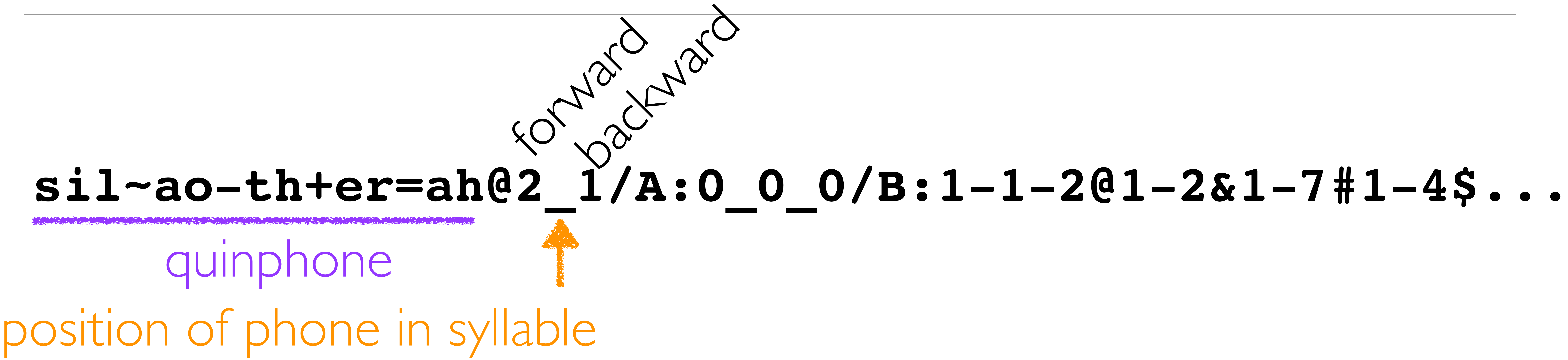
Anatomy of a context-dependent phone

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

Anatomy of a context-dependent phone



forward

backward

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

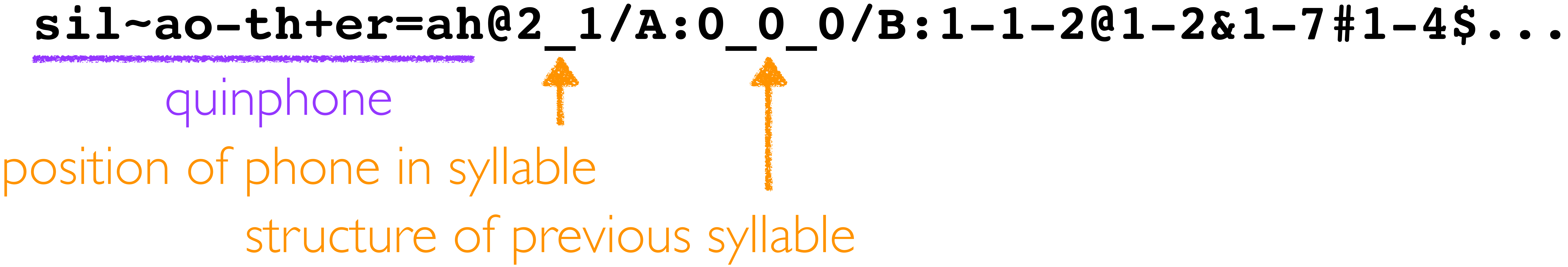
Anatomy of a context-dependent phone

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

Anatomy of a context-dependent phone



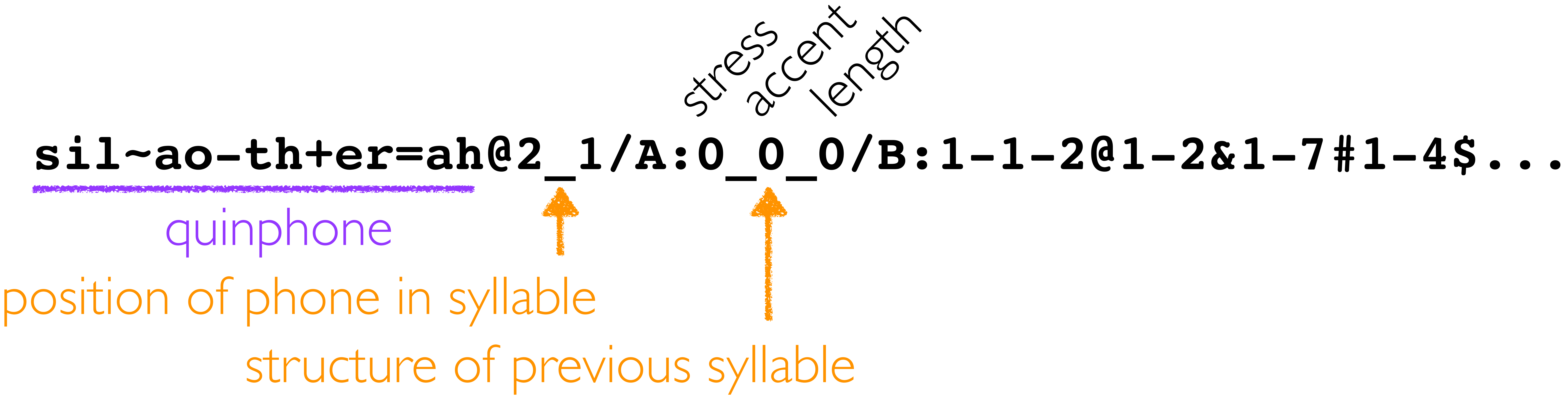
```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

structure of previous syllable

Anatomy of a context-dependent phone



stress accent length

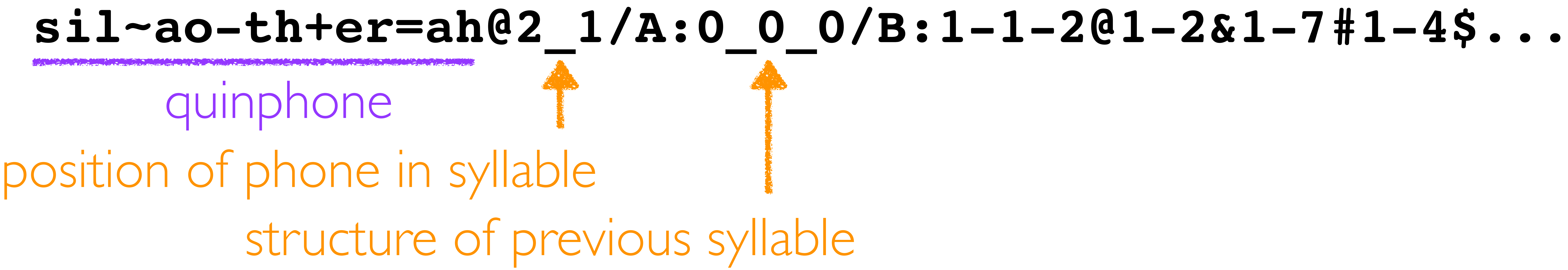
```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

structure of previous syllable

Anatomy of a context-dependent phone



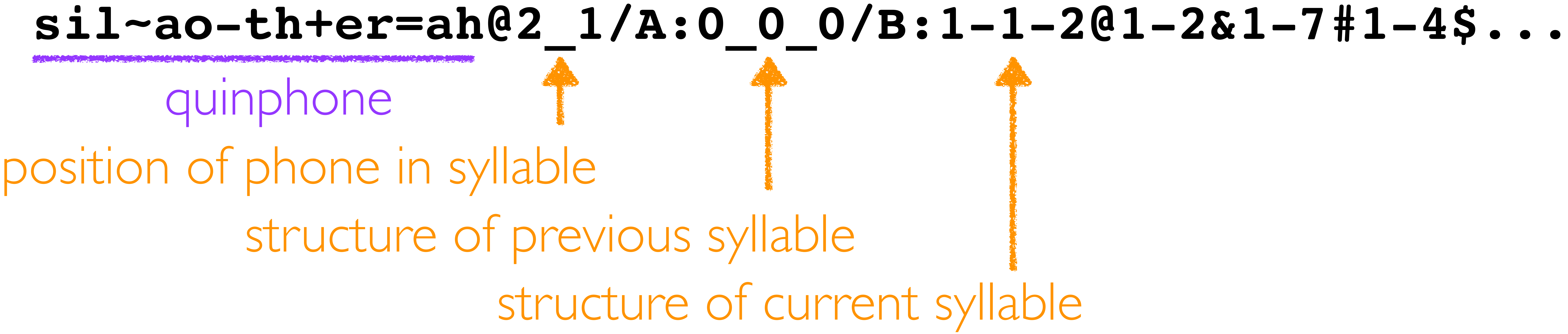
```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

structure of previous syllable

Anatomy of a context-dependent phone



```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

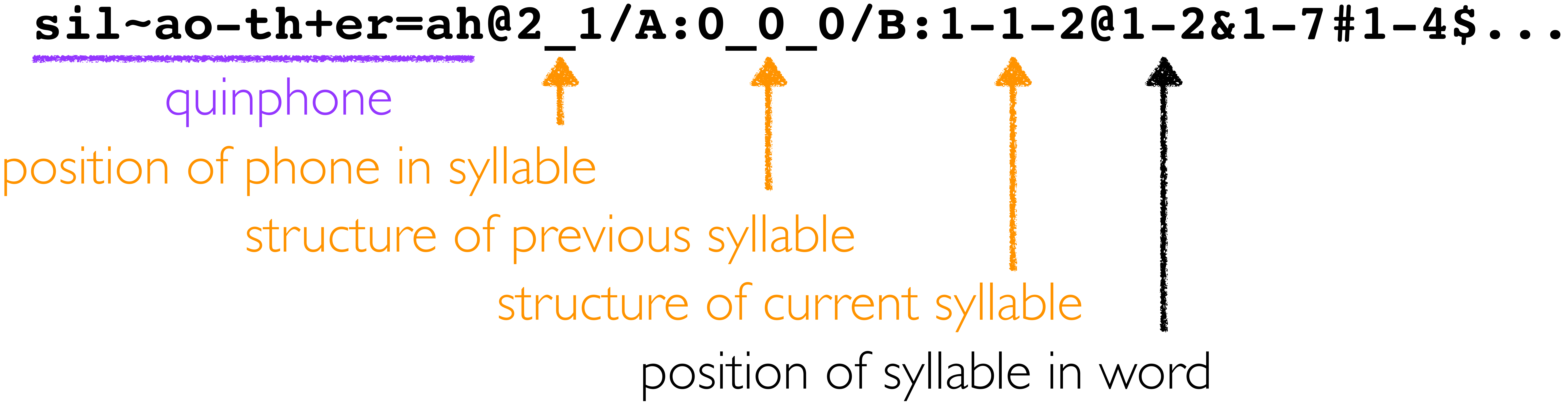
quinphone

position of phone in syllable

structure of previous syllable

structure of current syllable

Anatomy of a context-dependent phone



```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

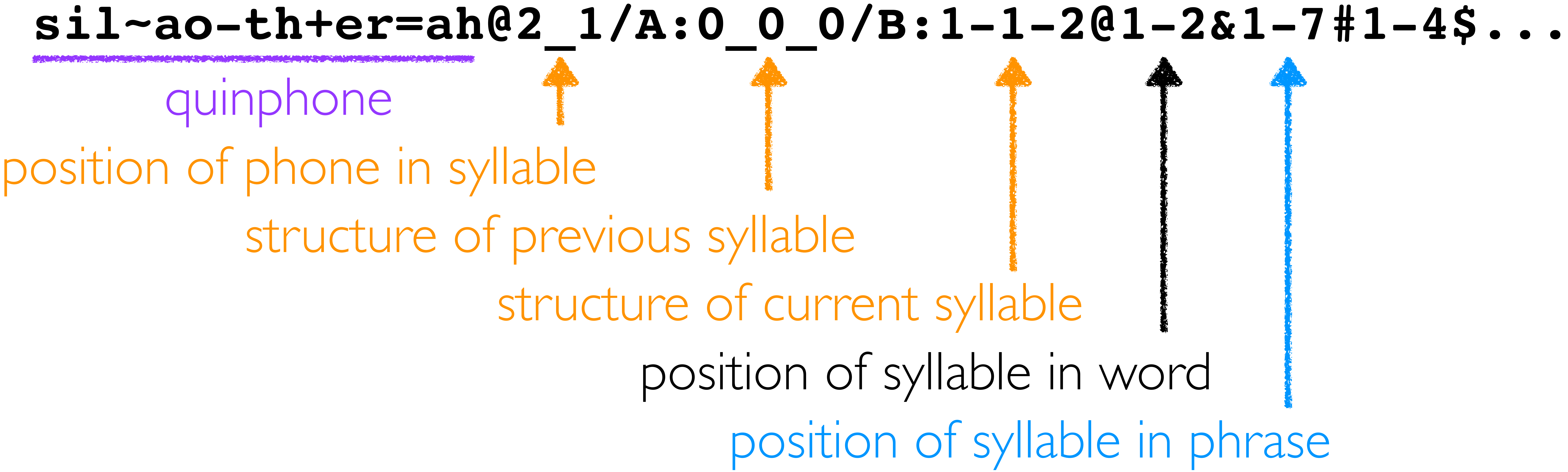
position of phone in syllable

structure of previous syllable

structure of current syllable

position of syllable in word

Anatomy of a context-dependent phone



```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

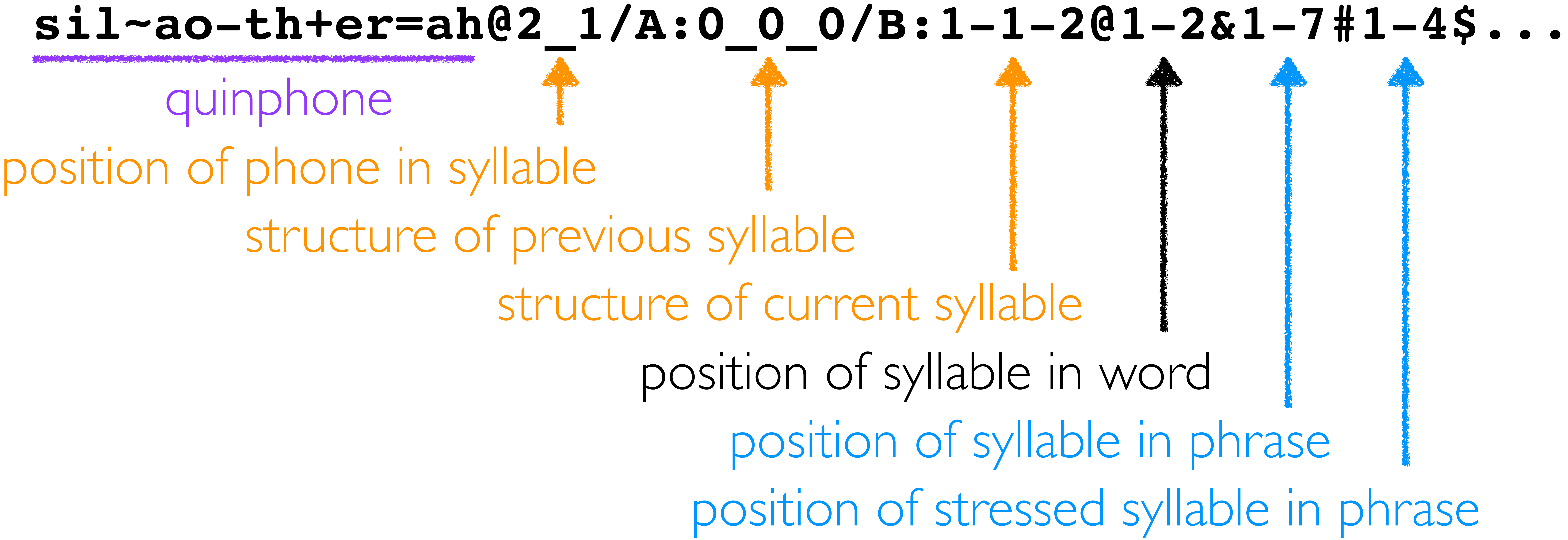
structure of previous syllable

structure of current syllable

position of syllable in word

position of syllable in phrase

Anatomy of a context-dependent phone



```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
```

quinphone

position of phone in syllable

structure of previous syllable

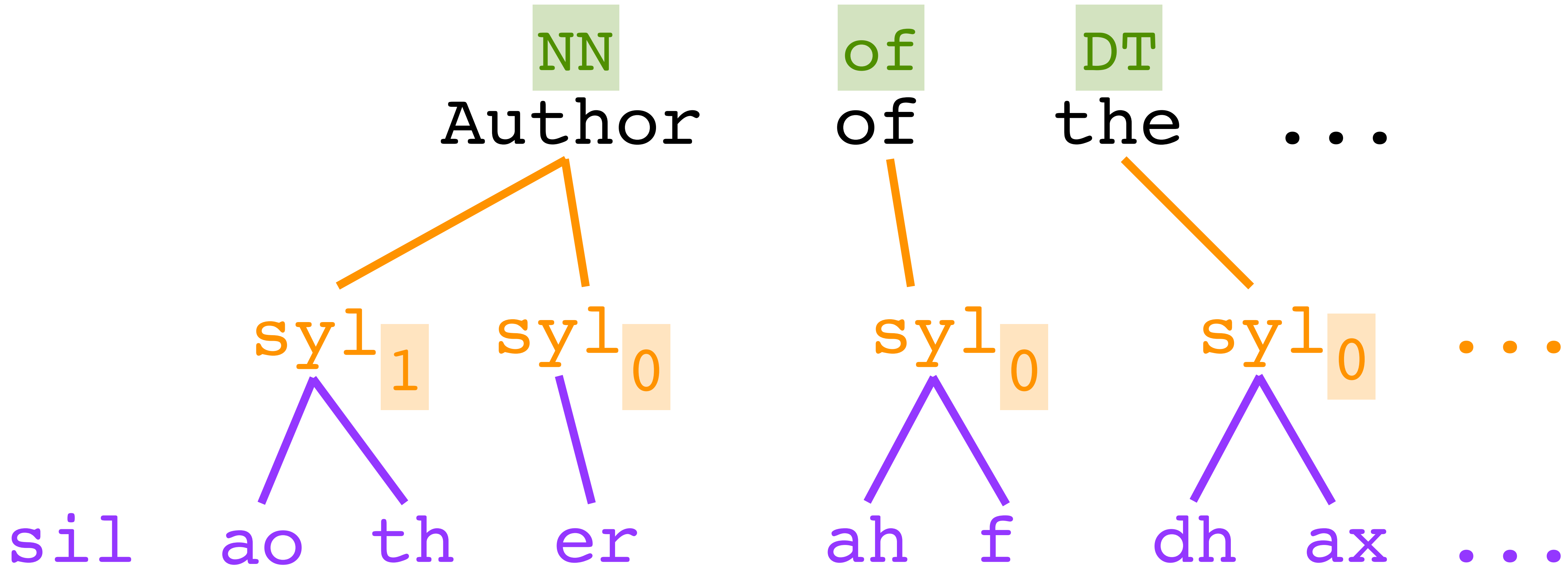
structure of current syllable

position of syllable in word

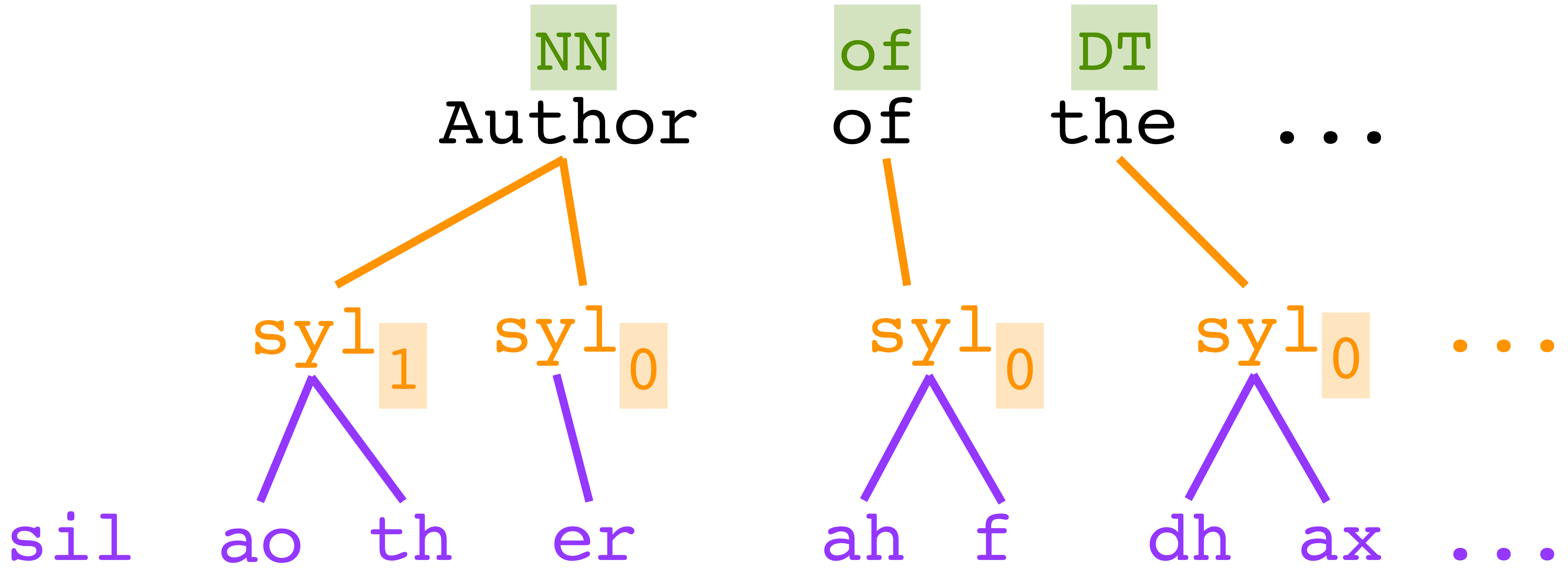
position of syllable in phrase

position of stressed syllable in phrase

Linguistic feature engineering: flatten to context-dependent phones



Linguistic feature engineering: flatten to context-dependent phones



```
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$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

Linguistic feature engineering: flatten to context-dependent phones

```
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$...
v~dh-ax+d=ey@2_1/A:1_0_2/B:0-0-2@1-1&4-4#2-3$...
```

next, encode each context-dependent phone as a vector

Example: encode one quinphone using 1-of-40 binary codes

`sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...`

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...
```

```
1000000000000000000000000000000000000000
```

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...

```
1000000000000000000000000000000000000000
0000100000000000000000000000000000000000
```

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...

```
1000000000000000000000000000000000000000
0000100000000000000000000000000000000000
0000000000000000000000000000000010000000
```

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...
```

```
1000000000000000000000000000000000000000
0000100000000000000000000000000000000000
0000000000000000000000000000000010000000
0000000000001000000000000000000000000000
```

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

```
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@...
```

```
1000000000000000000000000000000000000000
0000100000000000000000000000000000000000
0000000000000000000000000000000010000000
0000000000001000000000000000000000000000
0001000000000000000000000000000000000000
```

1	sil
2	aa
3	ae
4	ah
5	ao
6	aw
7	ay
8	b
9	ch
10	d
11	dh
12	eh
13	er
14	ey
15	f
16	g
17	hh
18	ih
19	iy
20	jh
21	k
22	l
23	m
24	n
25	ng
26	ow
27	oy
28	p
29	r
30	s
31	sh
32	t
33	th
34	uh
35	uw
36	v
37	w
38	y
39	z
40	zh

Example: encode one quinphone using 1-of-40 binary codes

```
1000000000000000000000000000000000000000
0000100000000000000000000000000000000000
0000000000000000000000000000000010000000
0000000000001000000000000000000000000000
0001000000000000000000000000000000000000
```

Example: encode one quinphone using 1-of-40 binary codes

```
000000000000000000000 0000000000000000000000000000000010000000   etc…

                      0000000000001000000000000000000000000000
                      0001000000000000000000000000000000000000
```

Linguistic feature engineering

```
                         .
                         .
sil~sil-ao+th=er@1_2/A:0_0_0/B:1-1-2@1-2&1-7#1-4$...
sil~ao-th+er=ah@2_1/A:0_0_0/B:1-1-2@1-2&1-7#1-4$…
                         .
                         .
```

Linguistic feature engineering

:

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

:

Linguistic feature engineering

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

Linguistic feature engineering: upsample to acoustic framerate

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

Linguistic feature engineering

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 1 0 0 1 0 1 1 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

[0 0 0 1 1 1 0 1 0 0 ...]

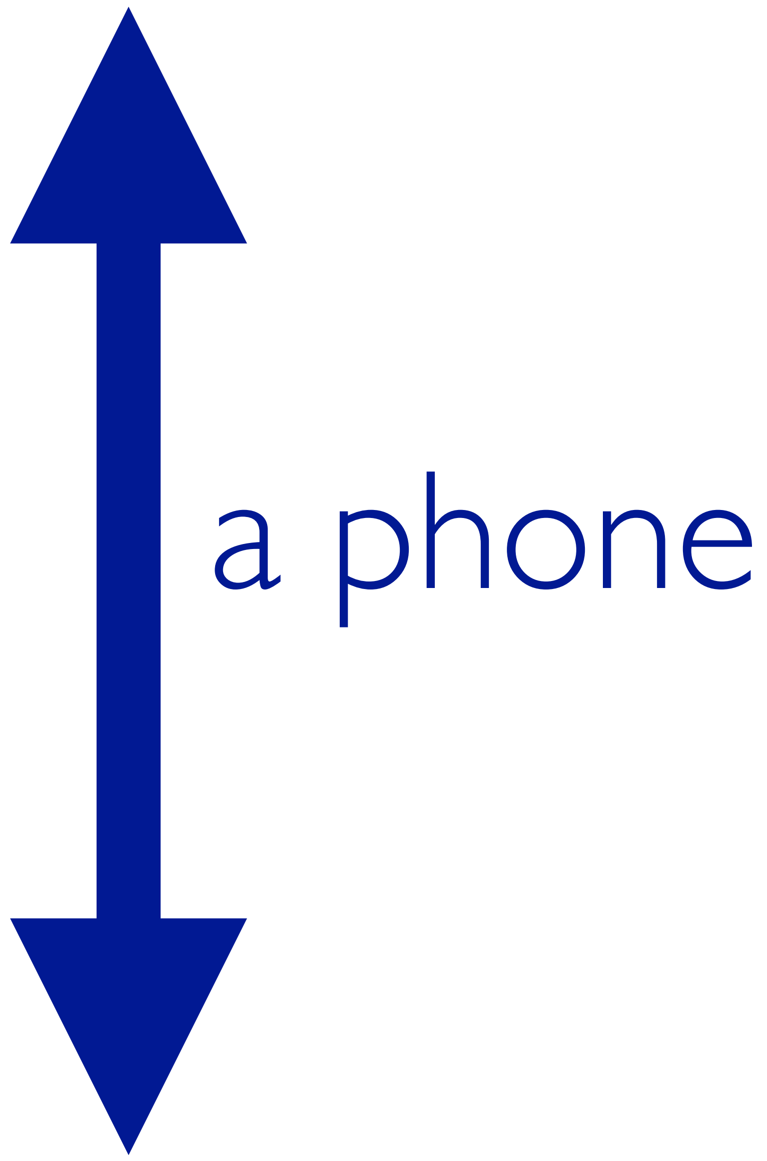
[0 0 0 1 1 1 0 1 0 0 ...]

Linguistic feature engineering: add within-phone positional features

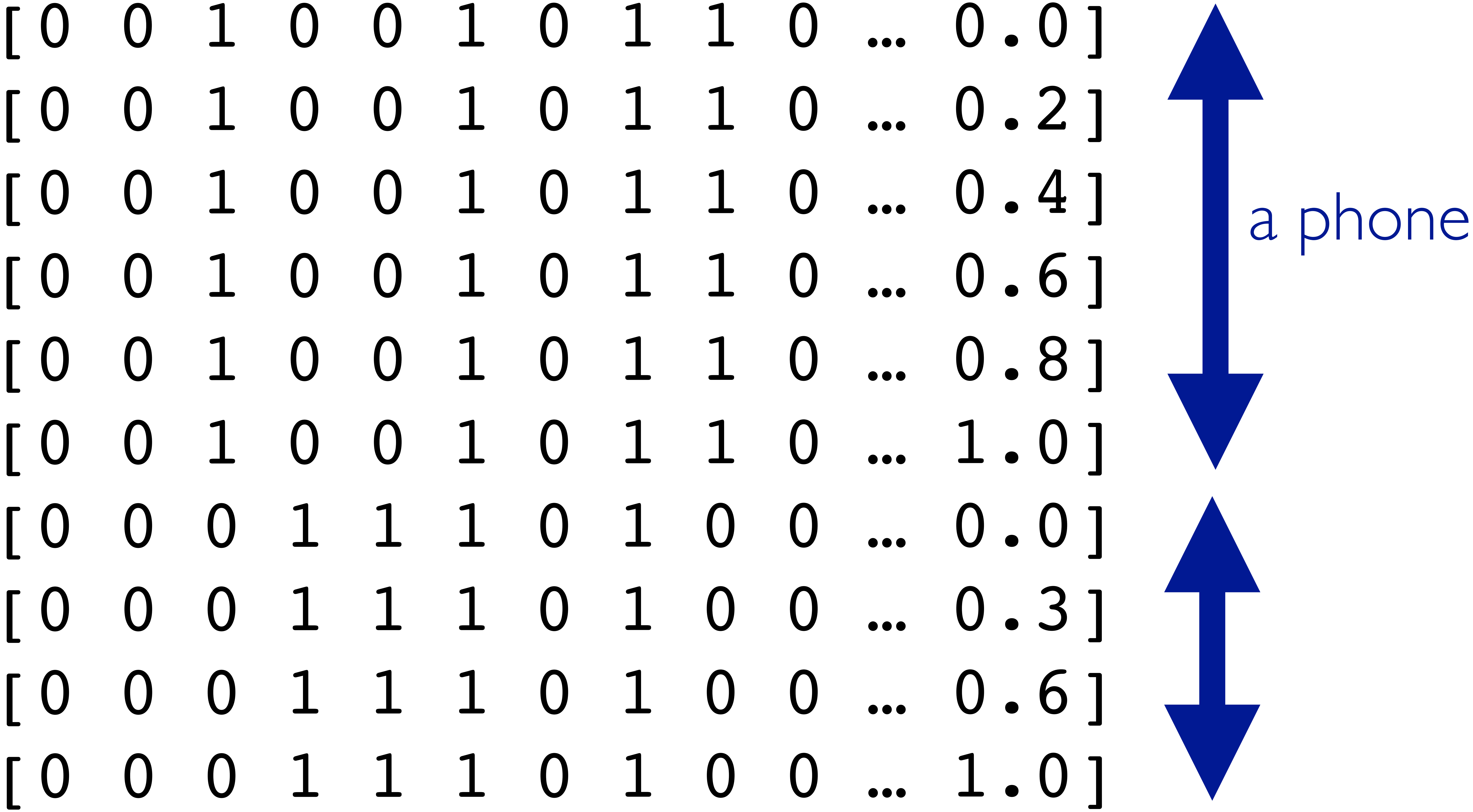
```
[0 0 1 0 0 1 0 1 1 0 … 0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2]
[0 0 1 0 0 1 0 1 1 0 … 0.4]
[0 0 1 0 0 1 0 1 1 0 … 0.6]
[0 0 1 0 0 1 0 1 1 0 … 0.8]
[0 0 1 0 0 1 0 1 1 0 … 1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.3]
[0 0 0 1 1 1 0 1 0 0 … 0.6]
[0 0 0 1 1 1 0 1 0 0 … 1.0]
```

Linguistic feature engineering: add within-phone positional features

```
[0 0 1 0 0 1 0 1 1 0 … 0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2]
[0 0 1 0 0 1 0 1 1 0 … 0.4]
[0 0 1 0 0 1 0 1 1 0 … 0.6]
[0 0 1 0 0 1 0 1 1 0 … 0.8]
[0 0 1 0 0 1 0 1 1 0 … 1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.3]
[0 0 0 1 1 1 0 1 0 0 … 0.6]
[0 0 0 1 1 1 0 1 0 0 … 1.0]
```



Linguistic feature engineering: add within-phone positional features



```
[0 0 1 0 0 1 0 1 1 0 … 0.0]
[0 0 1 0 0 1 0 1 1 0 … 0.2]
[0 0 1 0 0 1 0 1 1 0 … 0.4]
[0 0 1 0 0 1 0 1 1 0 … 0.6]
[0 0 1 0 0 1 0 1 1 0 … 0.8]
[0 0 1 0 0 1 0 1 1 0 … 1.0]
[0 0 0 1 1 1 0 1 0 0 … 0.0]
[0 0 0 1 1 1 0 1 0 0 … 0.3]
[0 0 0 1 1 1 0 1 0 0 … 0.6]
[0 0 0 1 1 1 0 1 0 0 … 1.0]
```

Where ***exactly*** do the durations come from?

Duration

During system building (training)

- the training data must be **aligned**
- this is almost always done using **forced alignment** techniques borrowed from automatic speech recognition
- *Exception: true sequence-to-sequence models may not require such alignments*

For text-to-speech synthesis

- from a **duration model**
- learned from force-aligned speech (the same data as the acoustic model)
- *Exception: sometimes we might* ***copy*** *durations from a held-out natural example of the same utterance*

Where ***exactly*** do the durations come from?

02_prepare_labels.sh

```
# alignments can be state-level (like HTS) or phone-level
if [ "$Labels" == "state_align" ]
    ./scripts/run_state_aligner.sh $wav_dir $inp_txt $lab_dir $global_config_file

elif [ "$Labels" == "phone_align" ]
    ./scripts/run_phone_aligner.sh $wav_dir $inp_txt $lab_dir $global_config_file


# the alignments will be used to train the duration model later
cp -r $lab_dir/label_$Labels $duration_data_dir


# and to upsample the linguistic features to acoustic frame rate
# when training the acoustic model
cp -r $lab_dir/label_$Labels $acoustic_data_dir
```

run_state_aligner.sh

```
# do prepare full-contextual labels without timestamps
echo "preparing full-contextual labels using Festival frontend..."
bash ${WorkDir}/scripts/prepare_labels_from_txt.sh $inp_txt $lab_dir $global_config_file $train

# do forced alignment using HVite from HTK
python $aligner/forced_alignment.py
```

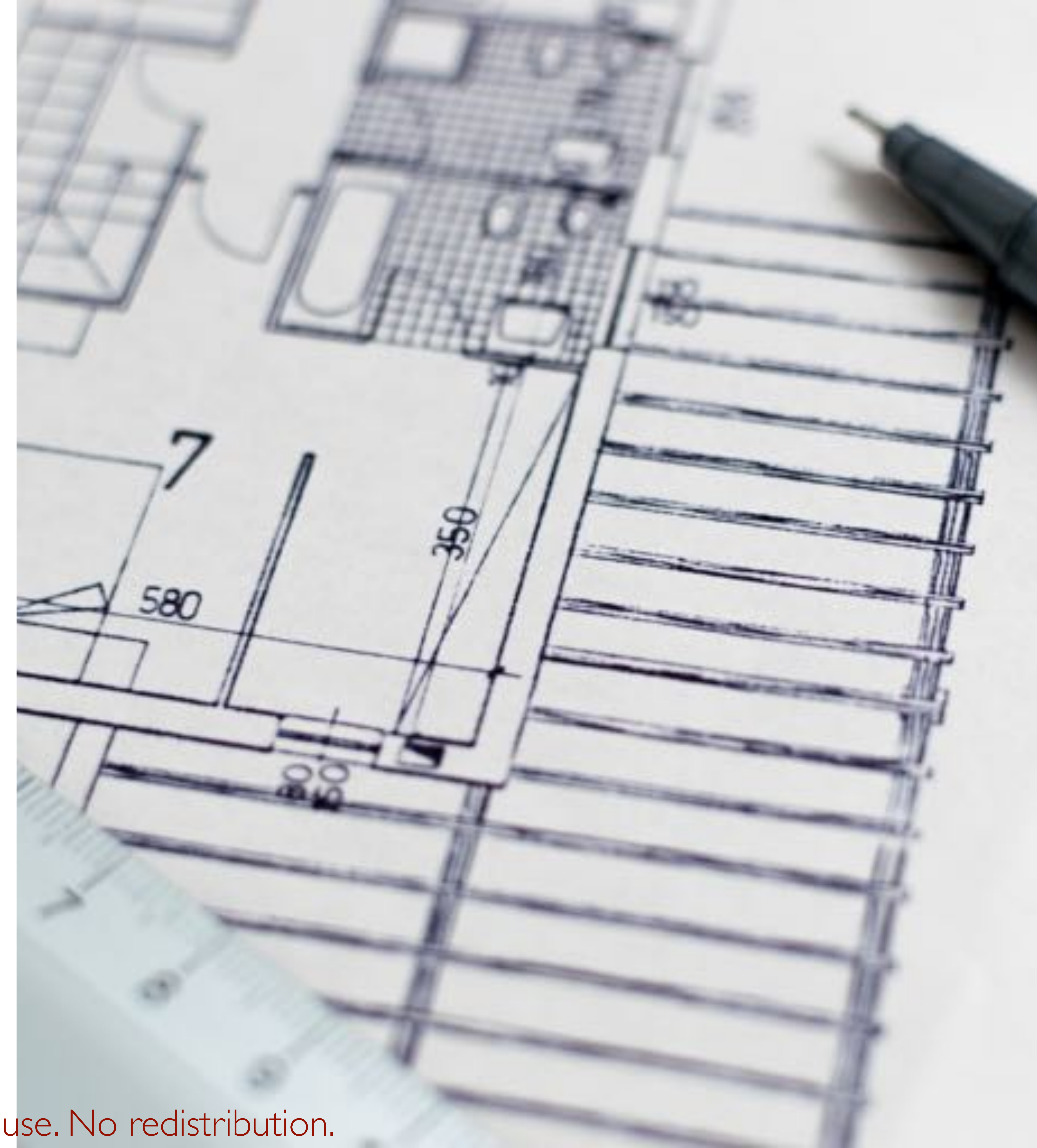
forced_alignment.py

```
aligner = ForcedAlignment()
aligner.prepare_training(file_id_list_name, wav_dir, lab_dir, work_dir, multiple_speaker)
aligner.train_hmm(7, 32)
aligner.align(work_dir, lab_align_dir)
```

Design choices: linguistic features

- Features
  - traditional front end (e.g., Festival)
  - data-driven features (e.g., Ossian)
  - best of both worlds?
  - ‘raw text’ (possibly normalised)

- Feature engineering
  - positional features as 1-of-K *or* numerical
  - sparse (1-of-K) *vs* dense (embeddings)



What next?

- We've prepared the input features
- Next
  - the output features
- After that
  - regression from input to output

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**
- Input
  - linguistic features
- Output
  - acoustic features

Orientation

- <u>Defining the problem of TTS</u>
  - **sequence-to-sequence regression**
- <u>Input</u>
  - linguistic features
- <u>Output</u>
  - acoustic features

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**

- Input
  - linguistic features

- Output
  - acoustic features

Requirements

- can be extracted from the waveform
- suitable for modelling
- can reconstruct the waveform

Agenda

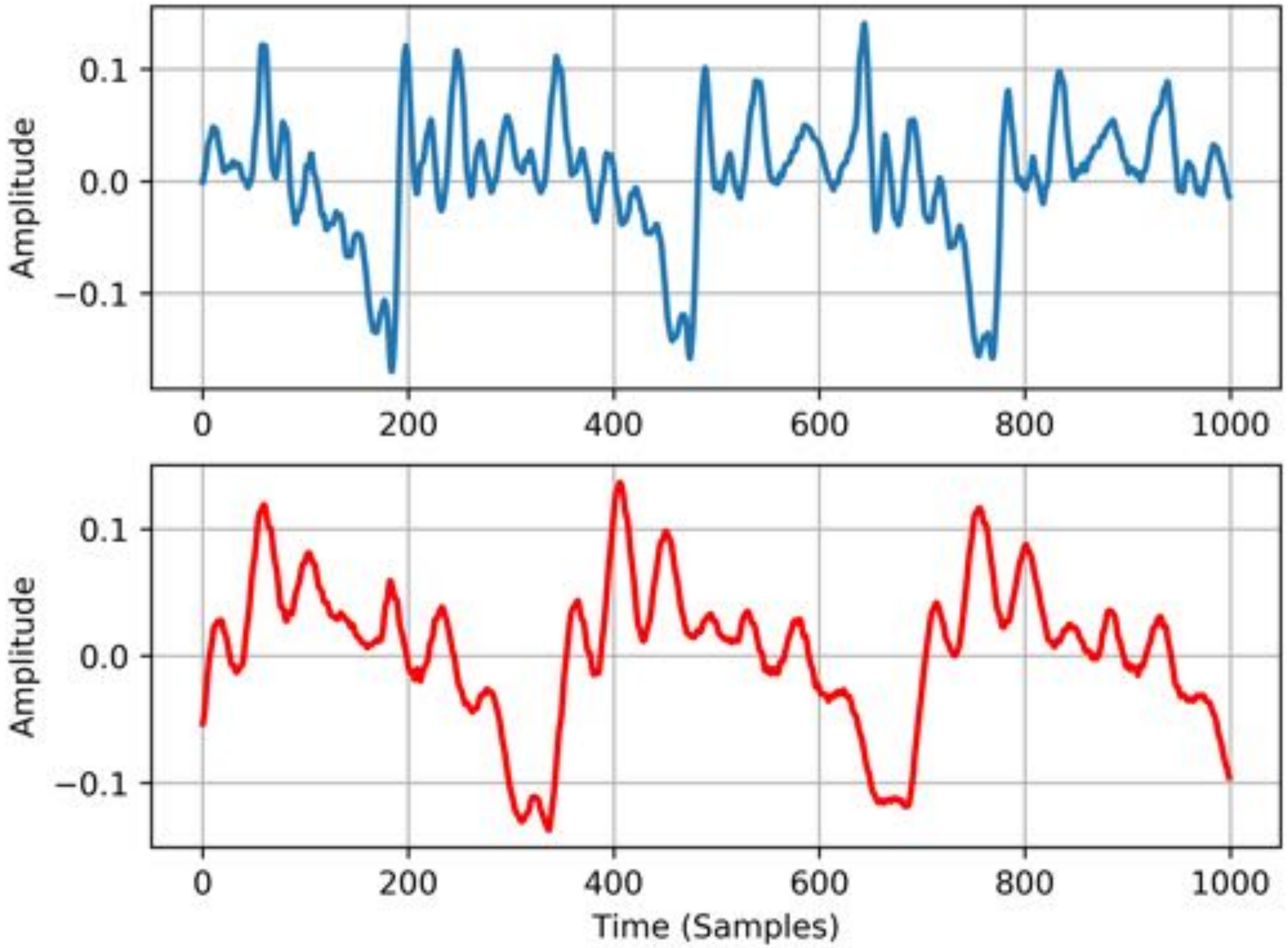
	Topic	**Presenter**
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	**Felipe Espic**
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

Acoustic feature extraction & engineering

Felipe Espic

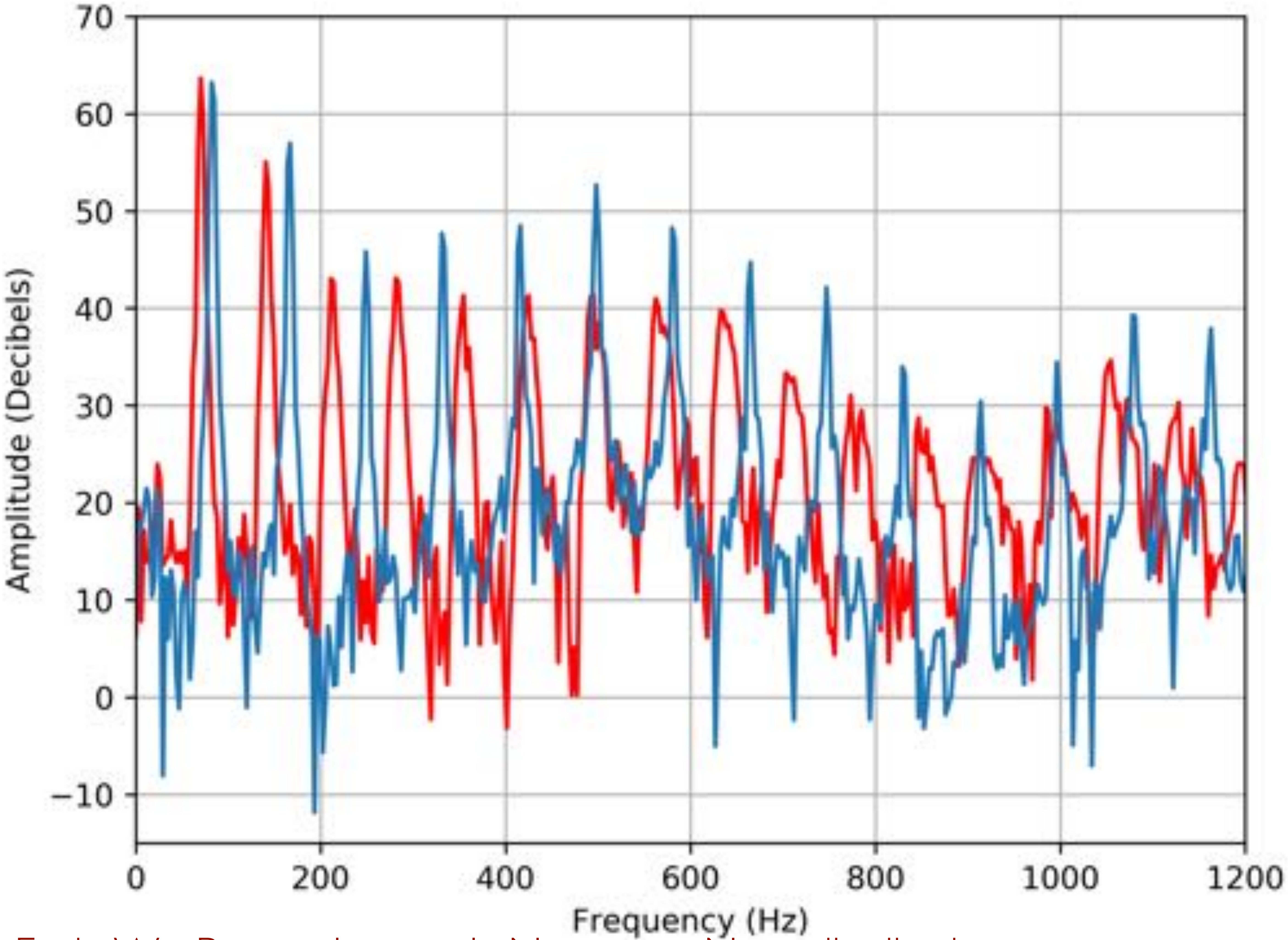
Why we use acoustic feature extraction - waveform

- Phoneme /a:/



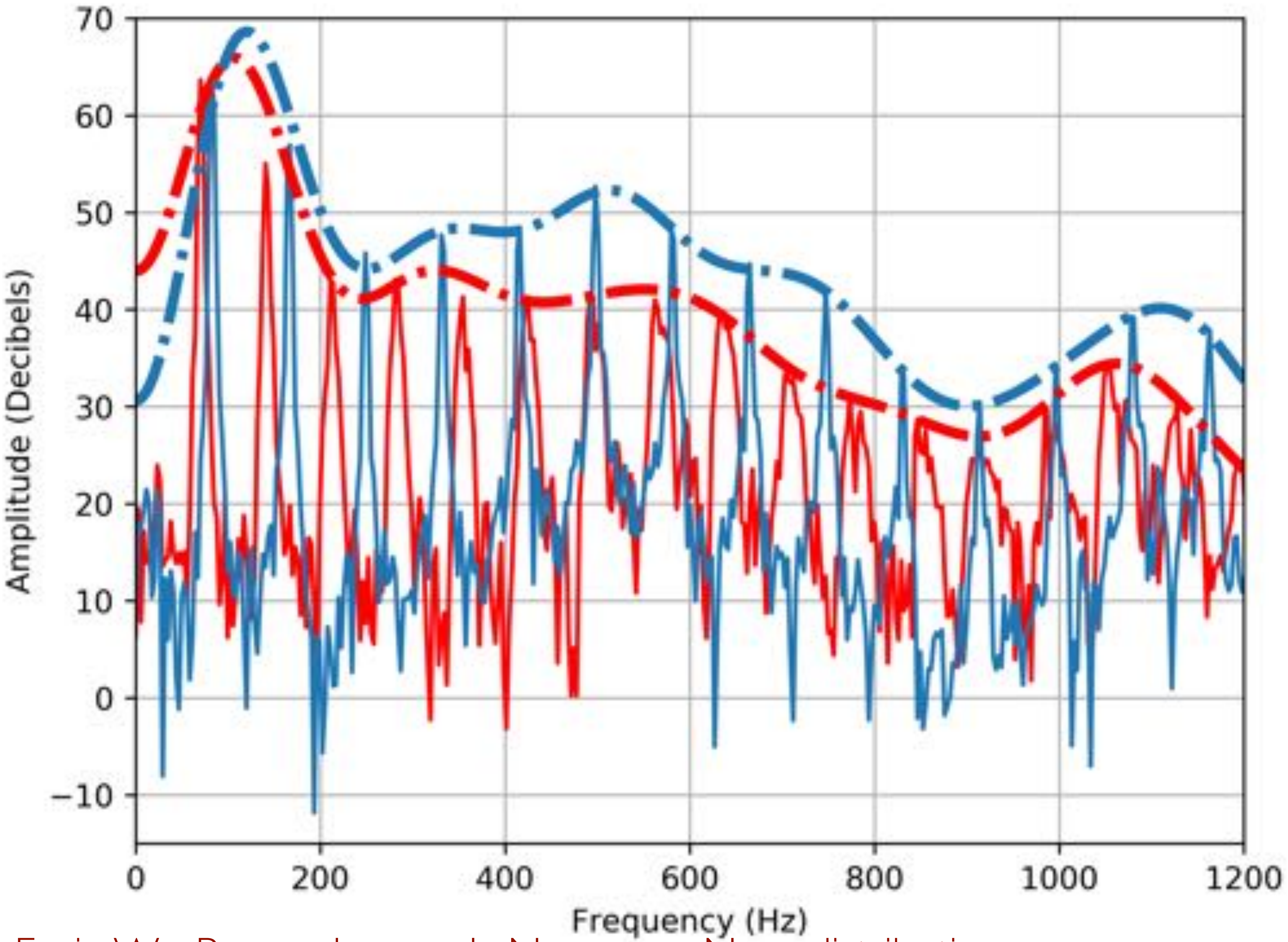
Why we use acoustic feature extraction - magnitude spectrum

- Phoneme /a:/



Why we use acoustic feature extraction - magnitude spectrum

- Phoneme /a:/



Terminology

- Spectral Envelope

- F0

- Aperiodic energy

Terminology

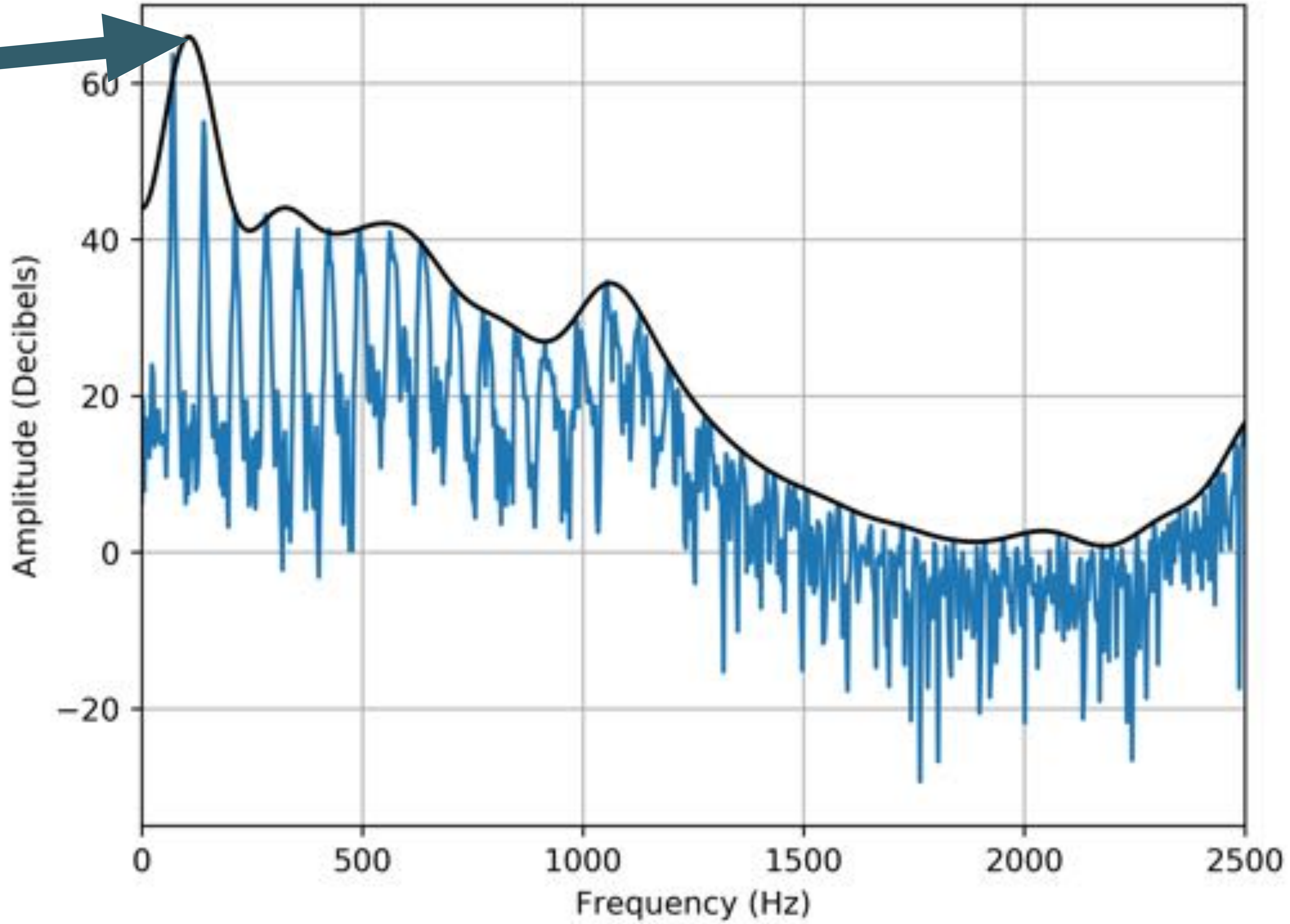
- Spectral Envelope

- F0

- Aperiodic energy

Terminology

- Spectral Envelope
- F0
- Aperiodic energy

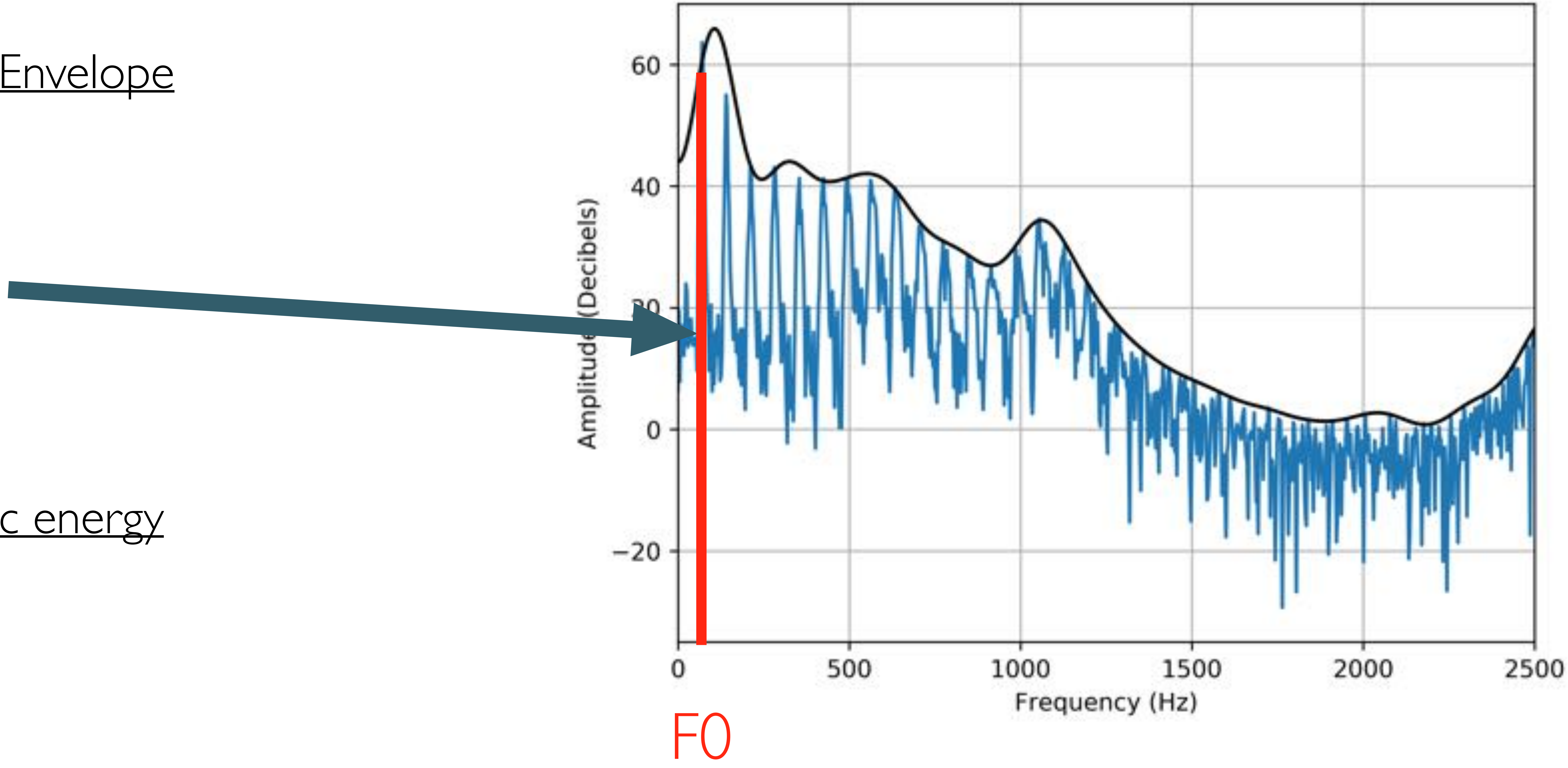


Terminology

- Spectral Envelope

- F0

- Aperiodic energy

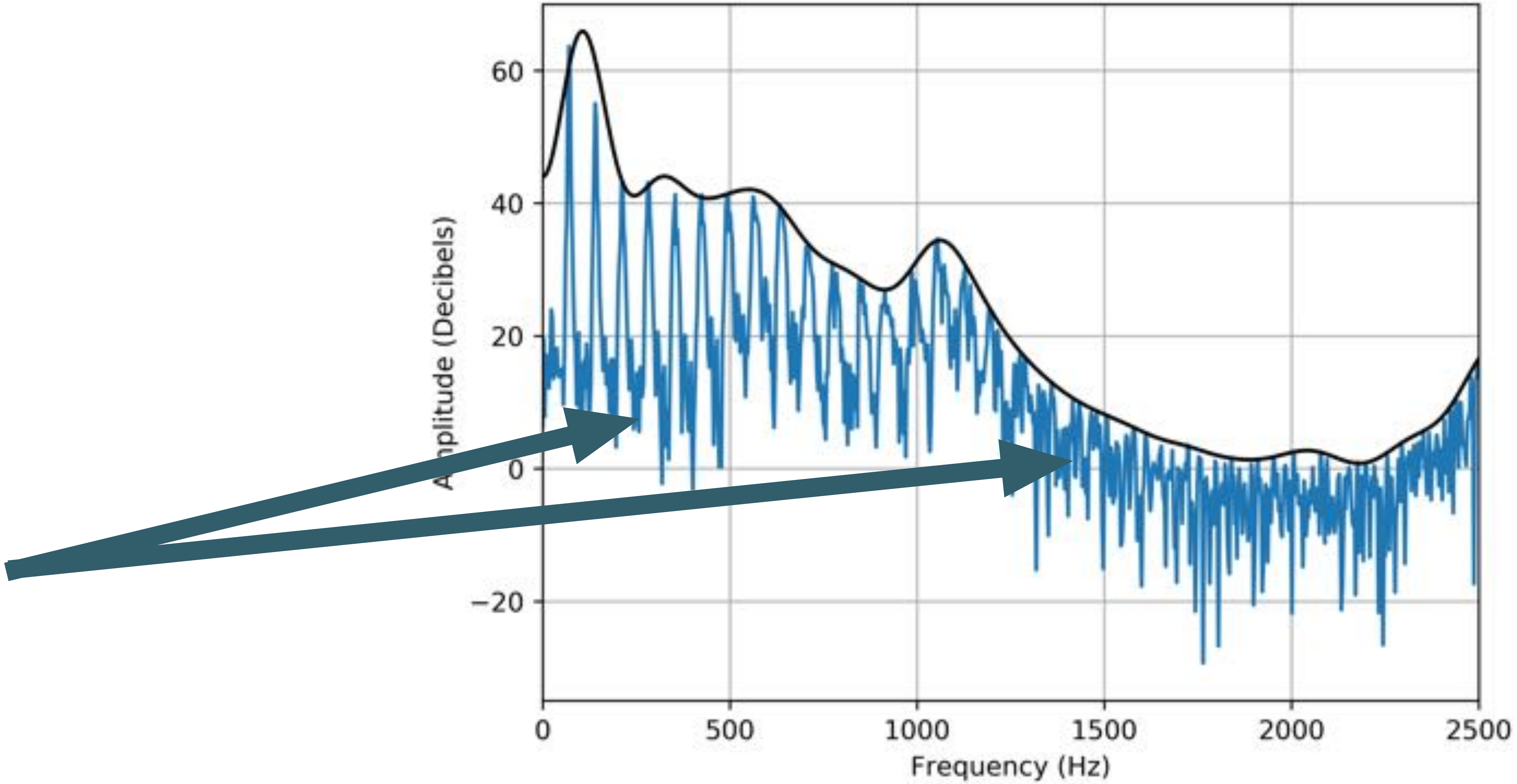


Terminology

- Spectral Envelope

- F0

- Aperiodic energy



A typical vocoder: WORLD

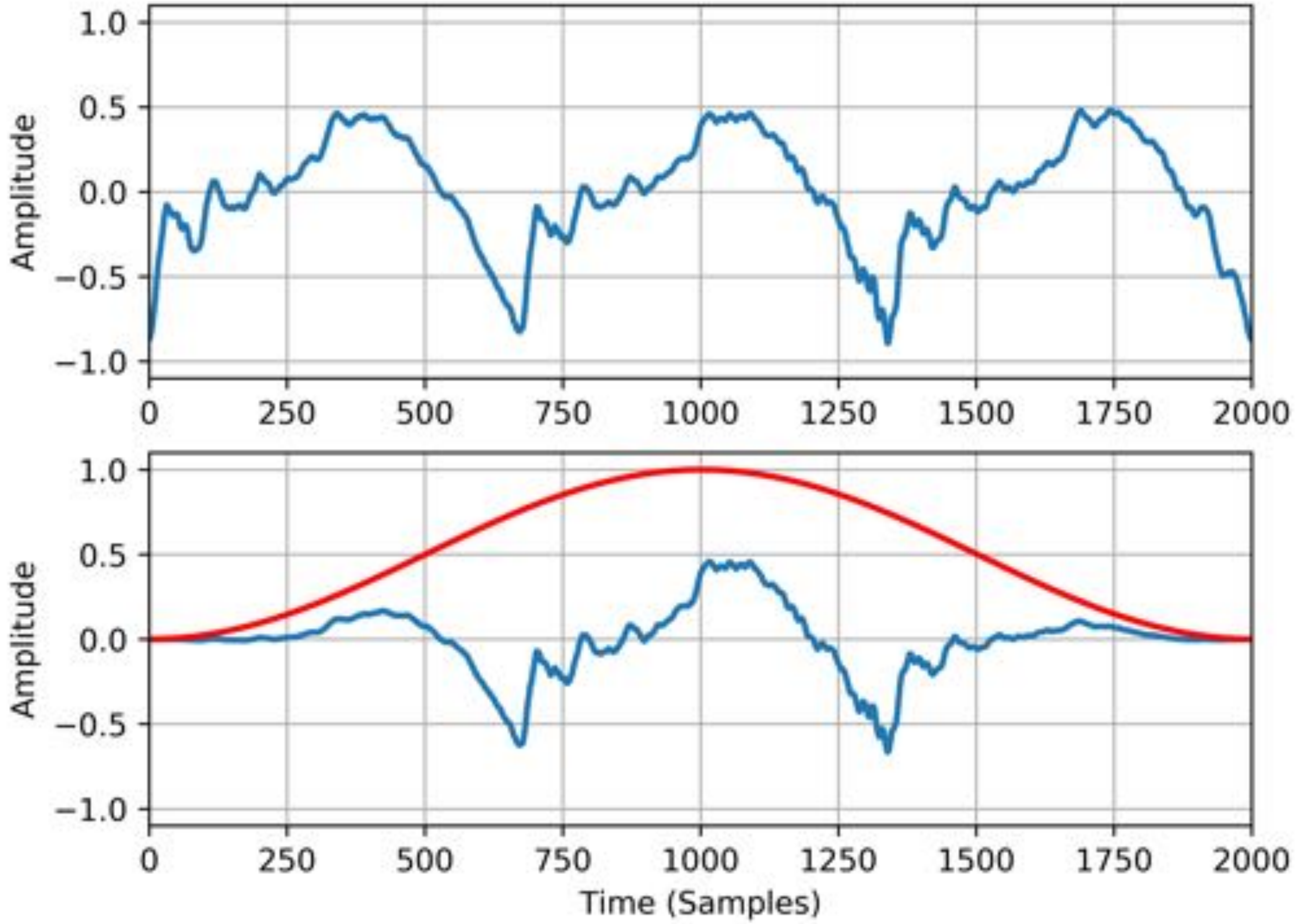
- Developed by Masanori Morise since 2009

- Free and Open Source (modified BSD licence)

- Speech Features:
  - **Spectral Envelope** (estimated using CheapTrick)
  - **F0** (estimated using DIO)
  - **Band aperiodicities** (estimated using D4C)

WORLD: spectral envelope estimation

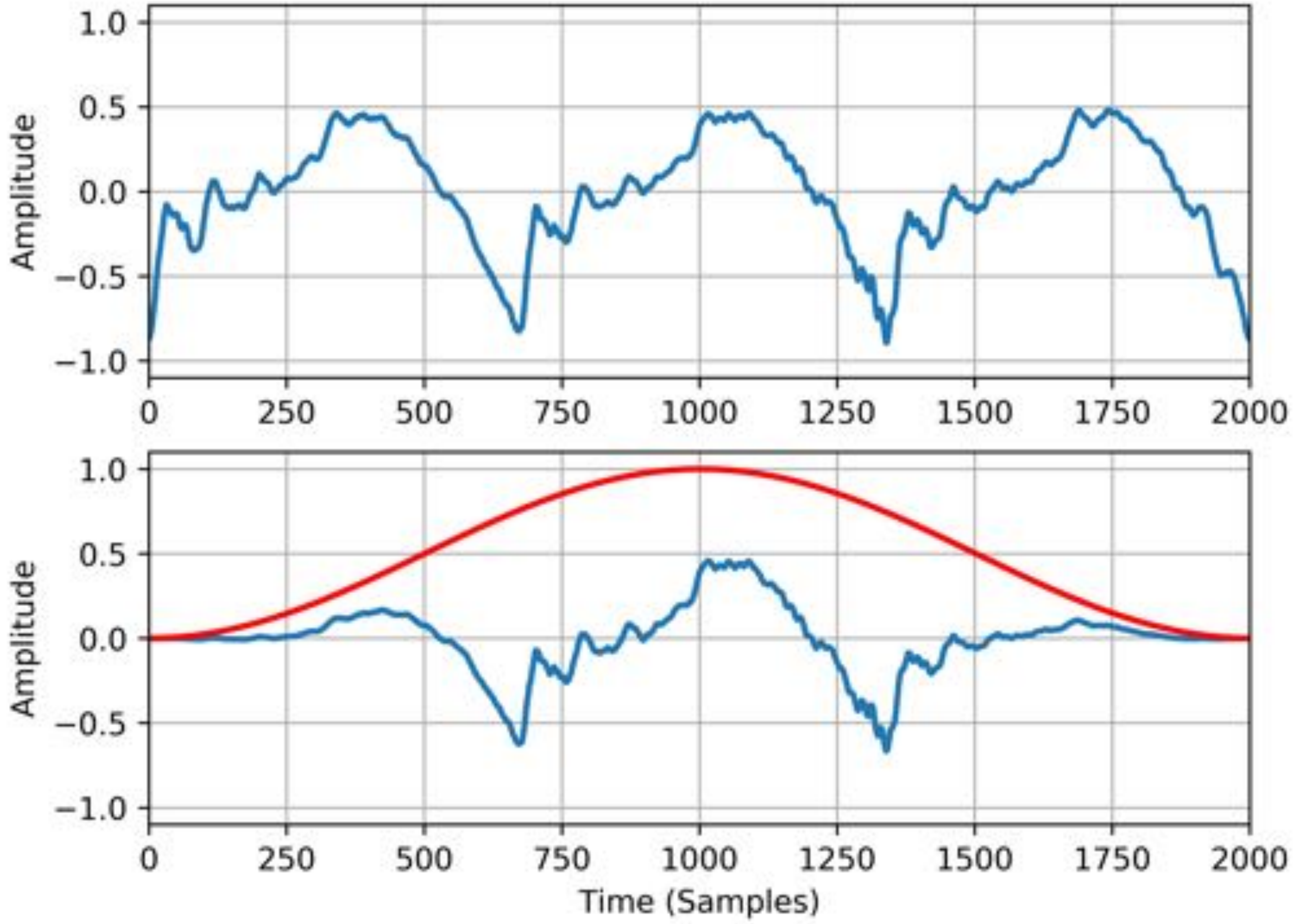
- Hanning window length 3 T0



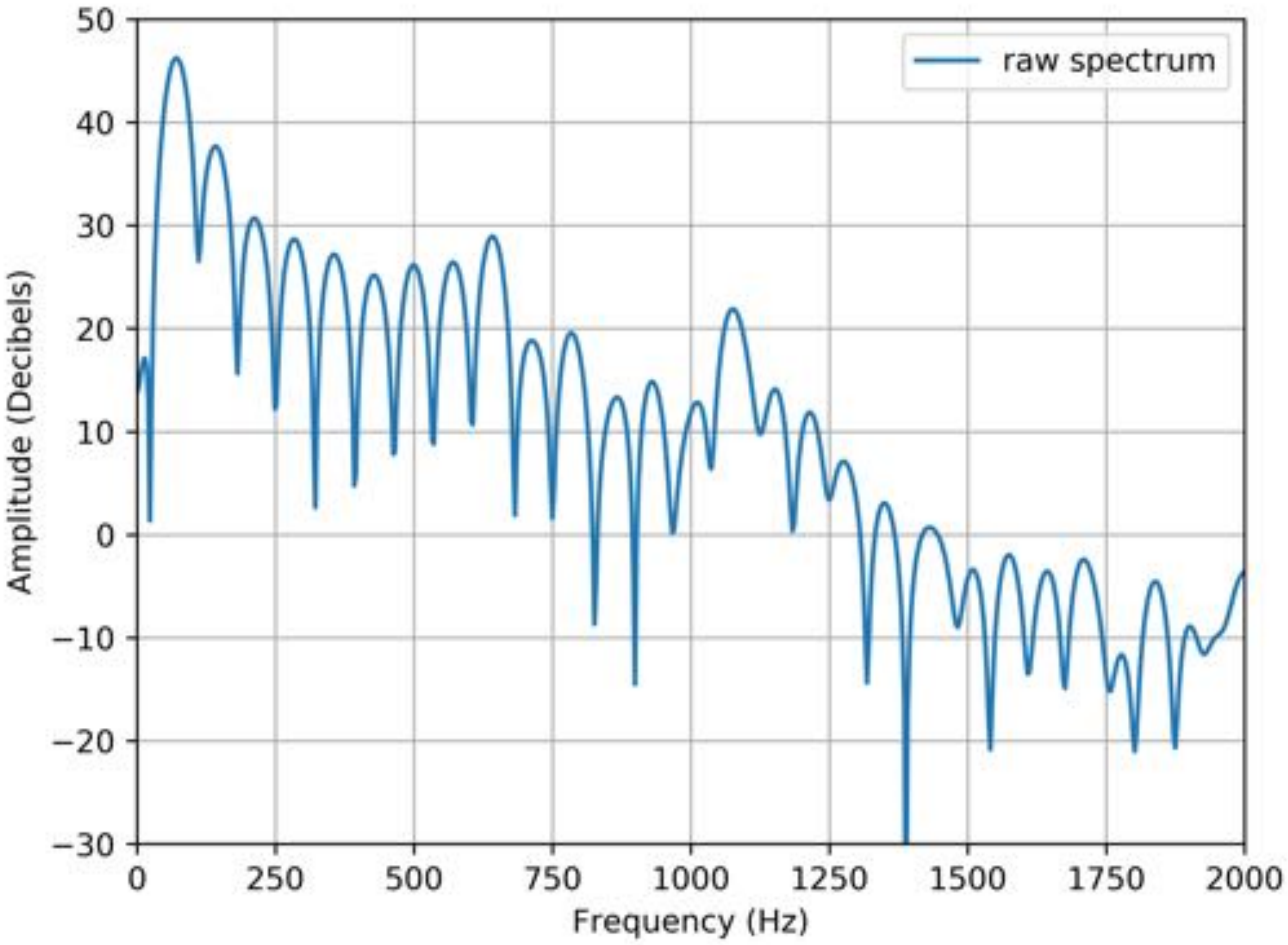
WORLD: spectral envelope estimation

- Hanning window length 3 T0

Power is temporally stable

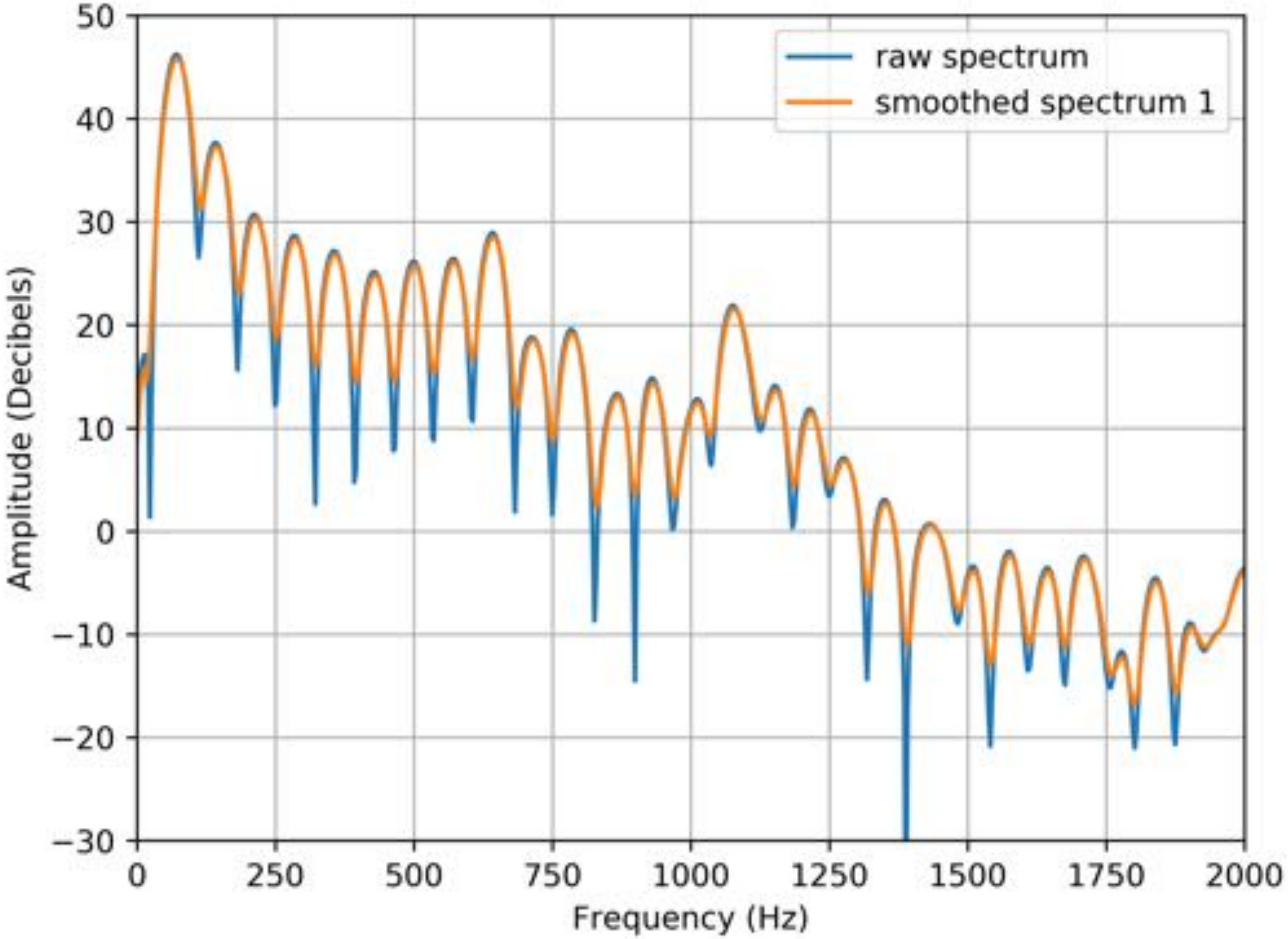


WORLD: spectral envelope estimation



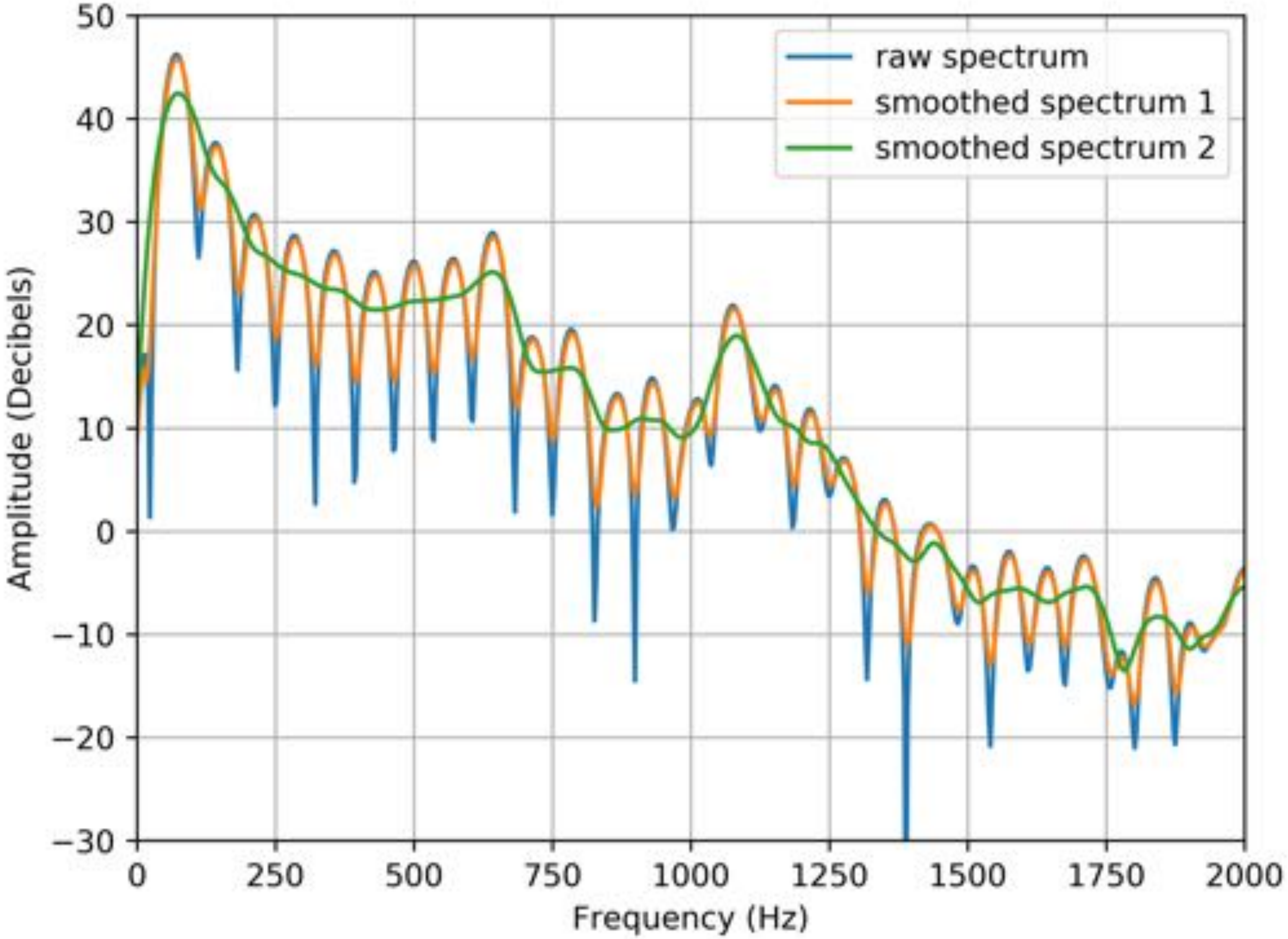
WORLD: spectral envelope estimation

- Apply a moving average filter
  - length (2/3) F0



WORLD: spectral envelope estimation

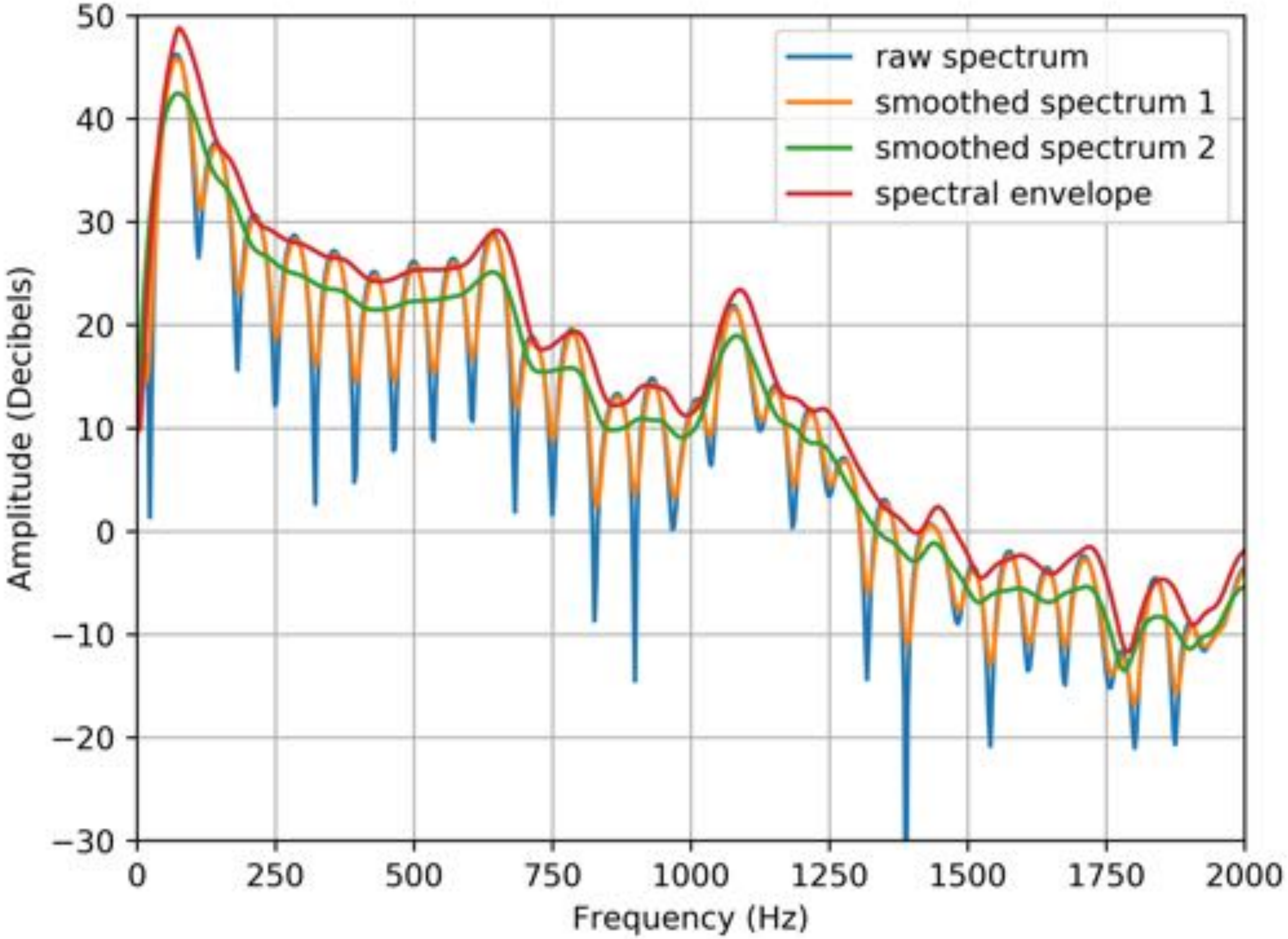
- Apply another moving average filter
  - length 2 F0



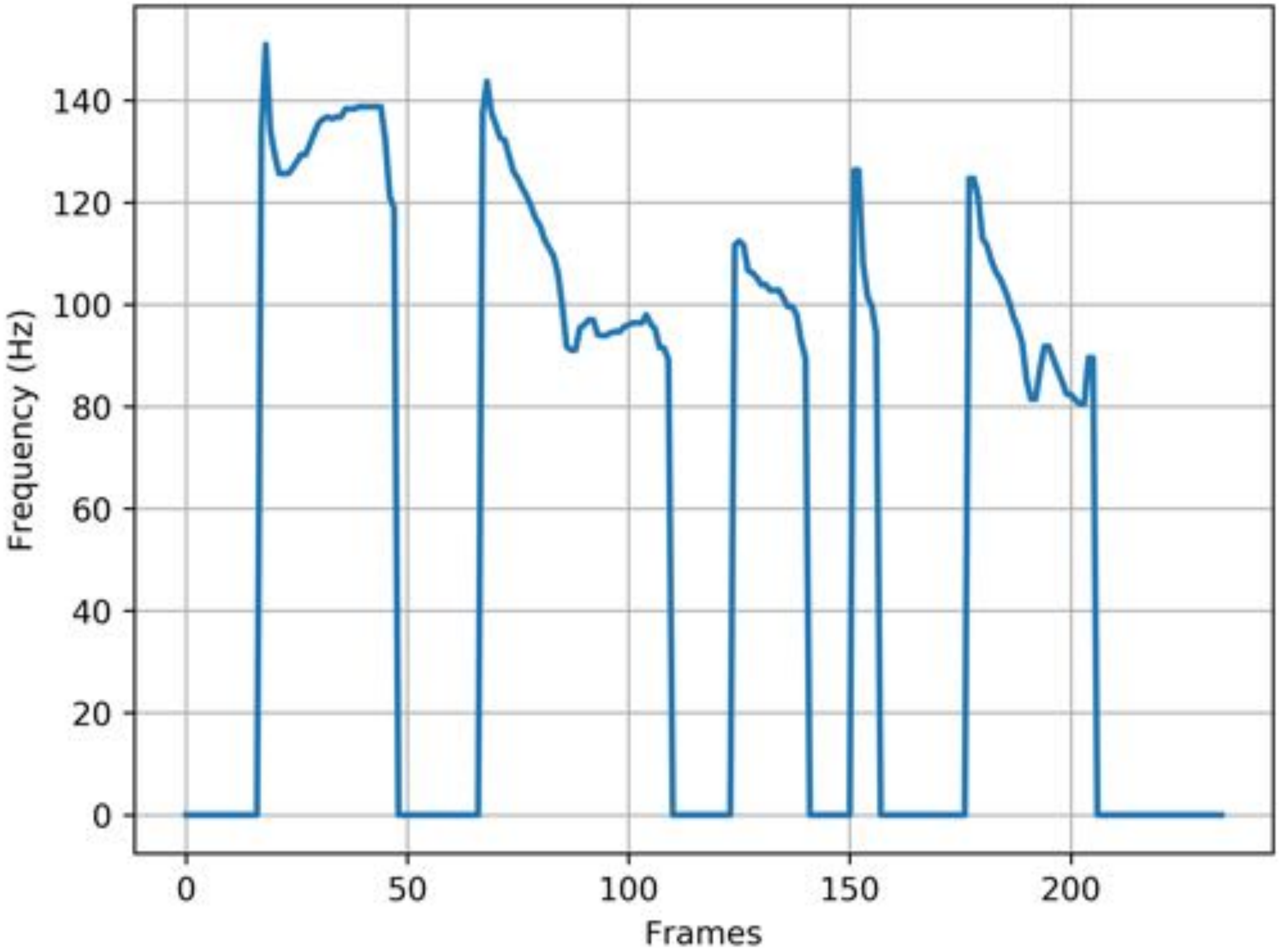
WORLD: spectral envelope estimation

- SpEnv = q0 logSp(F) + q1 logSp(F+F0)+q1 logSp(F-F0)

- *actually done in the cepstral domain*
- *illustrated here in the spectral domain*



WORLD: F0 estimation

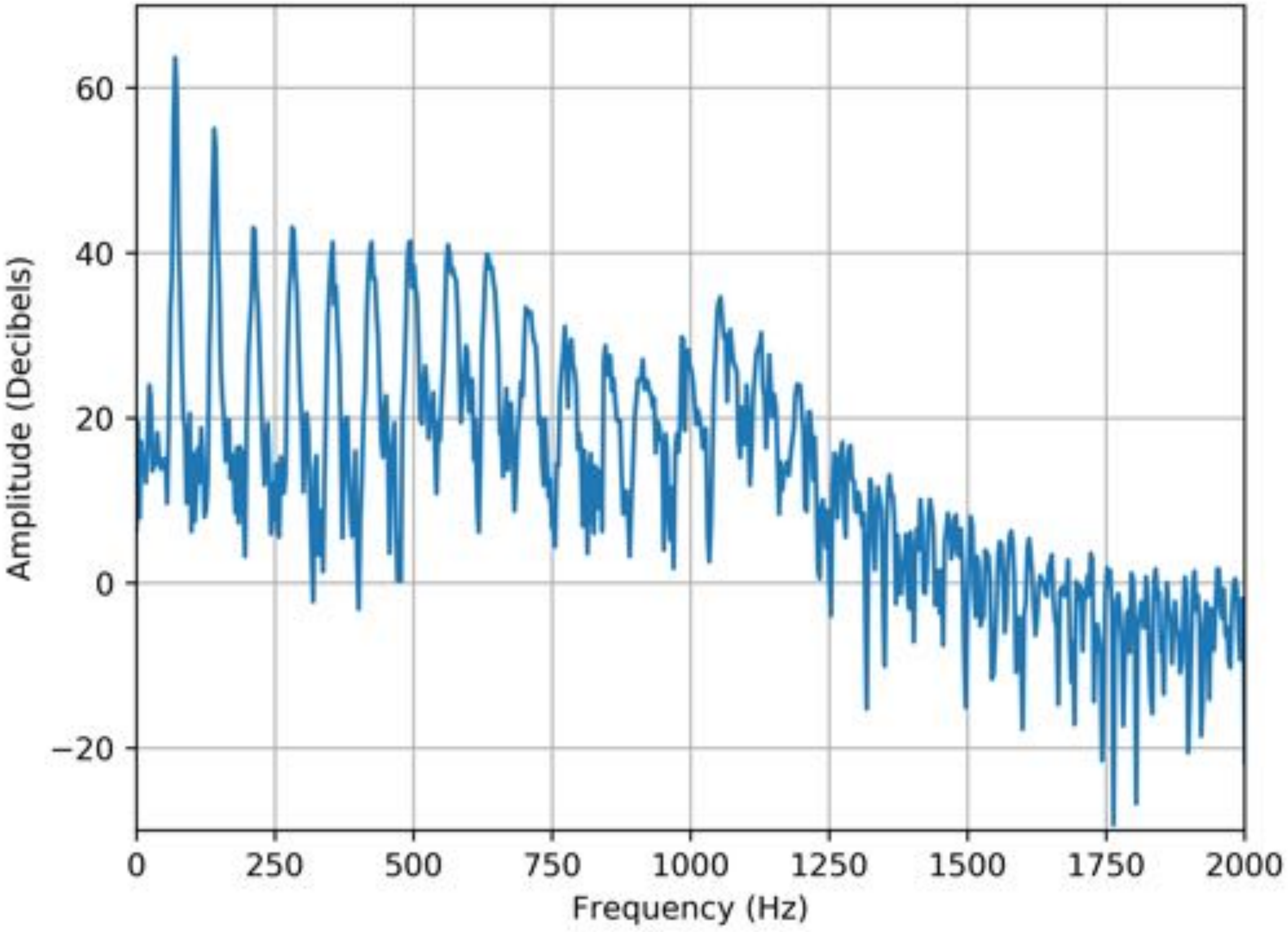


WORLD: band aperiodicities

- The **ratio** between aperiodic and periodic energy, averaged over certain frequency bands

- i.e., total power / sine wave power

- In the example, this ratio is
  - lowest in band **a**
  - more in band **b**
  - highest in band **c**

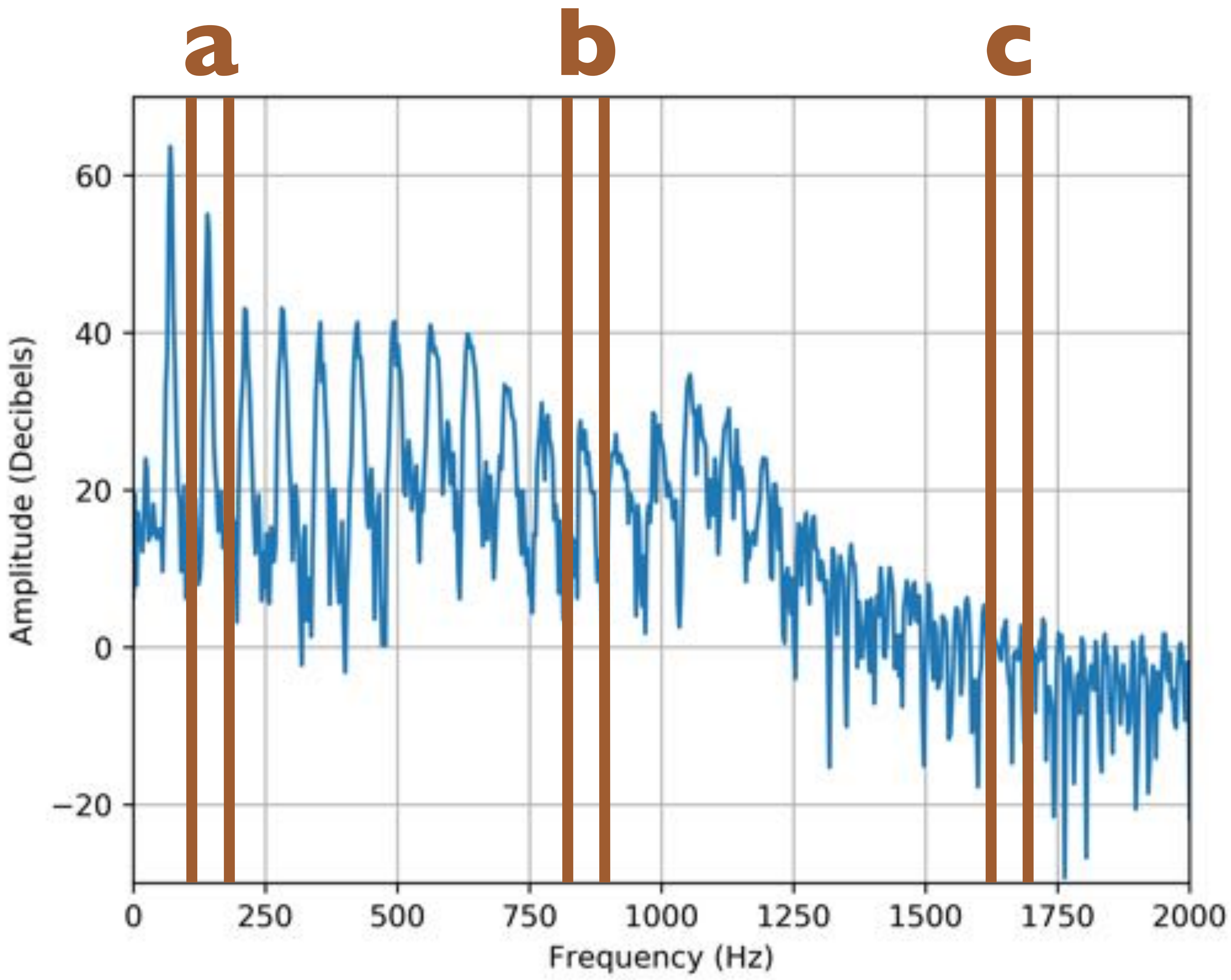


WORLD: band aperiodicities

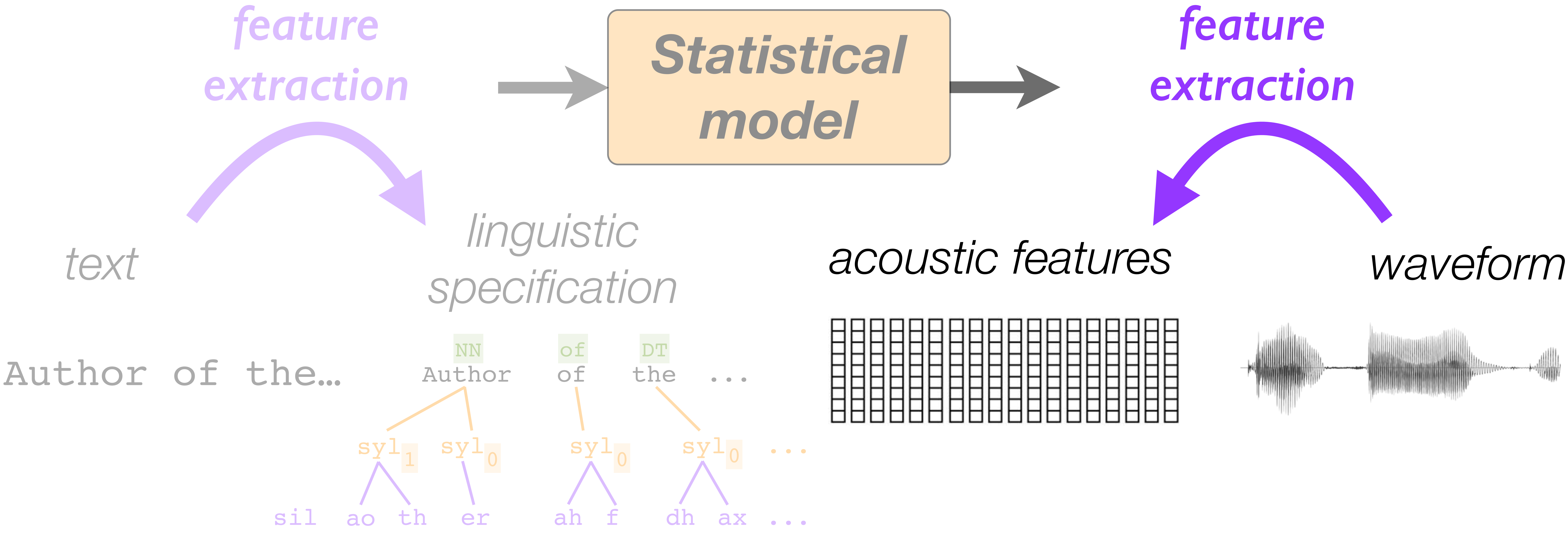
- The **ratio** between aperiodic and periodic energy, averaged over certain frequency bands

- i.e., total power / sine wave power

- In the example, this ratio is
  - lowest in band **a**
  - more in band **b**
  - highest in band **c**

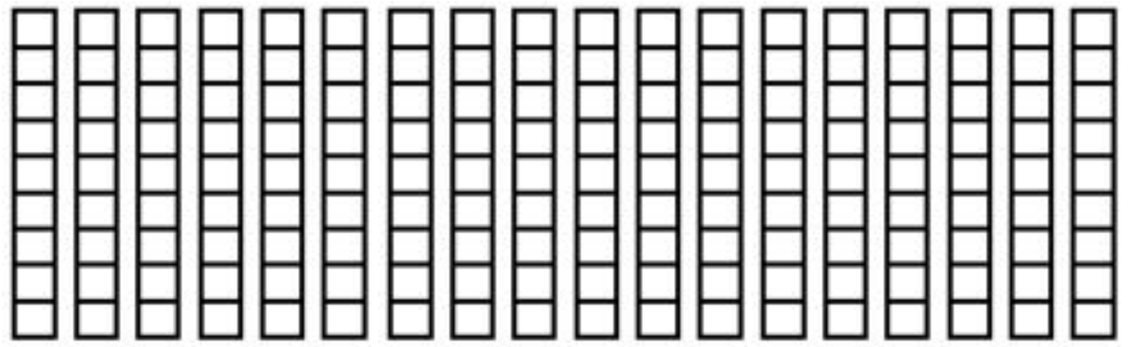


Acoustic feature extraction

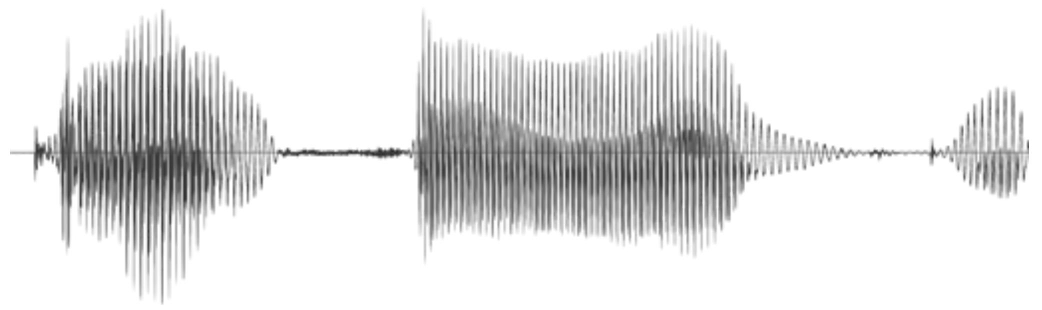


Acoustic feature extraction & engineering

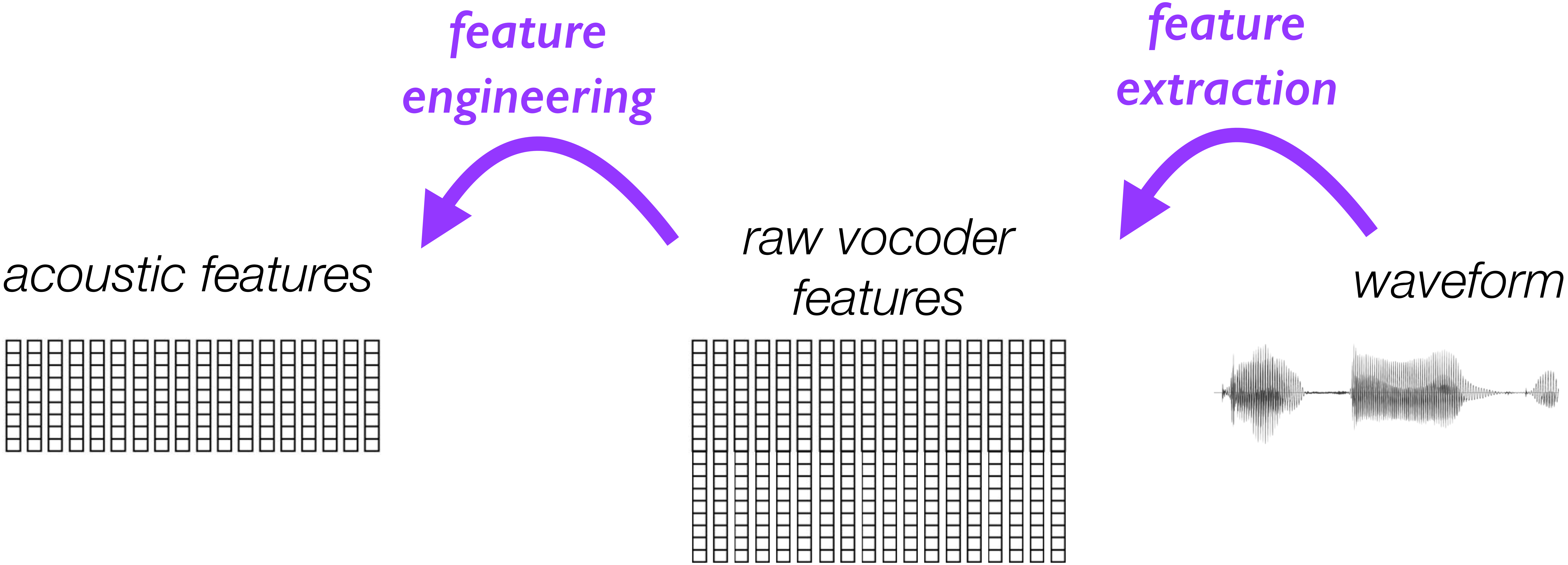
acoustic features



waveform



Acoustic feature extraction & engineering



feature engineering

feature extraction

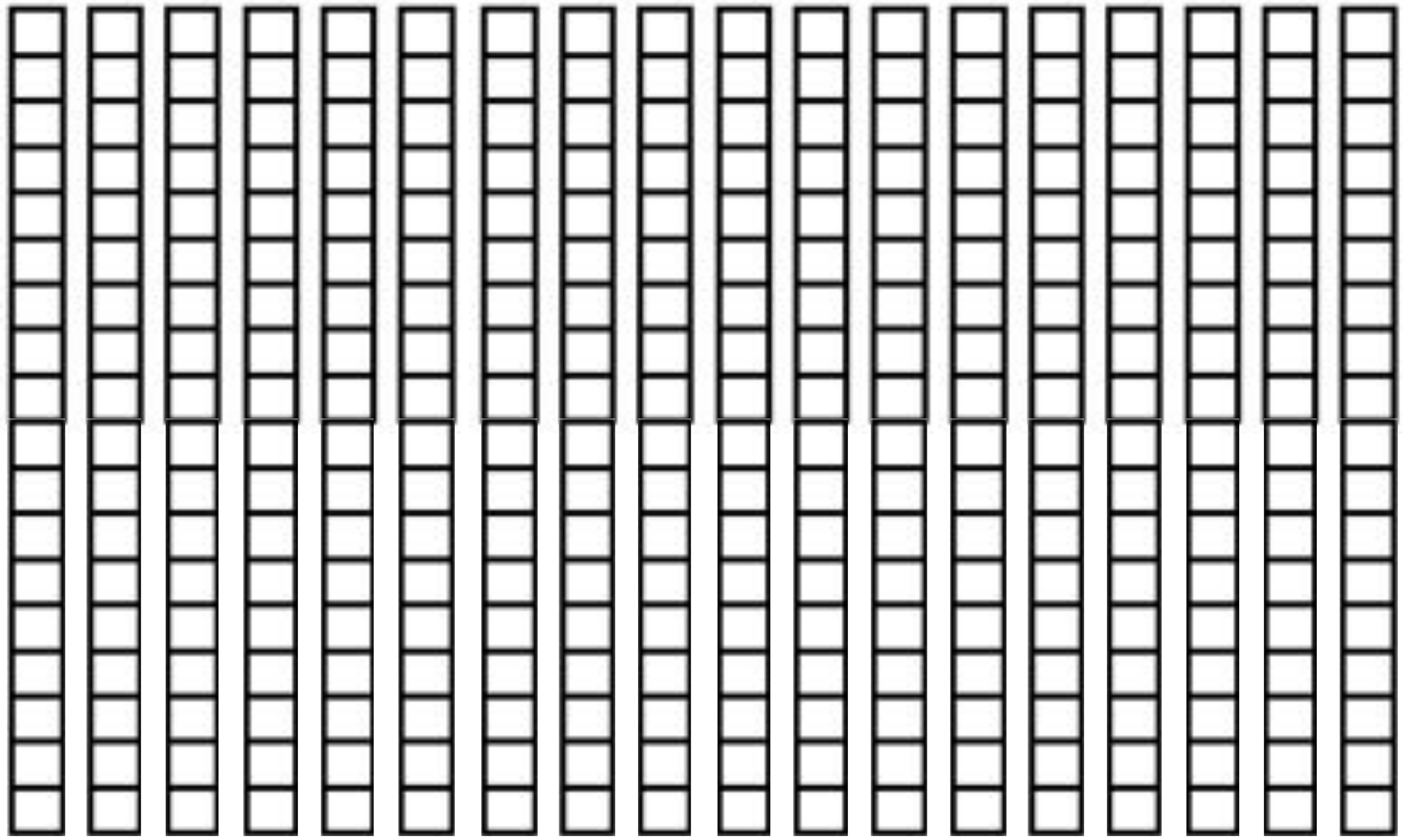
acoustic features

raw vocoder features

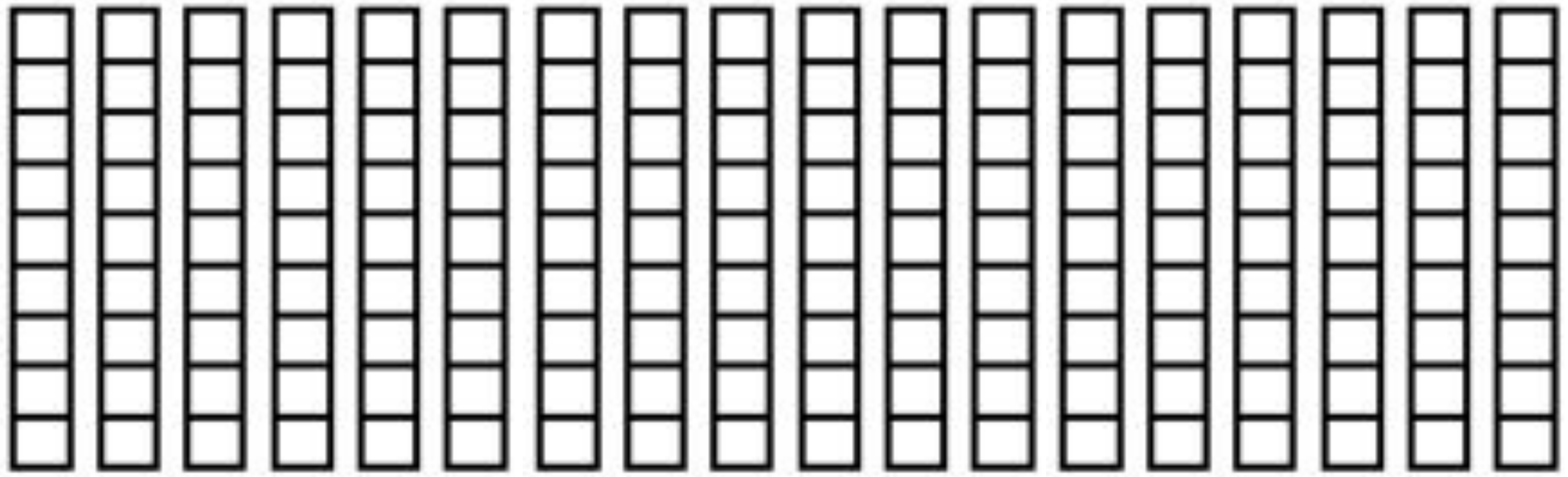
waveform

Acoustic feature engineering

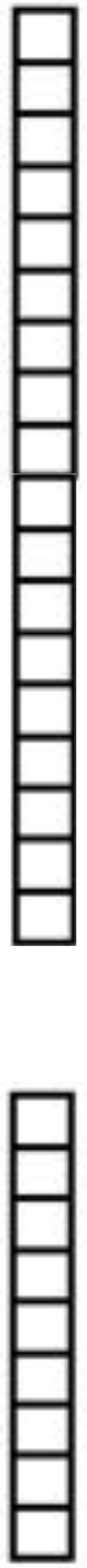
raw vocoder features



acoustic features

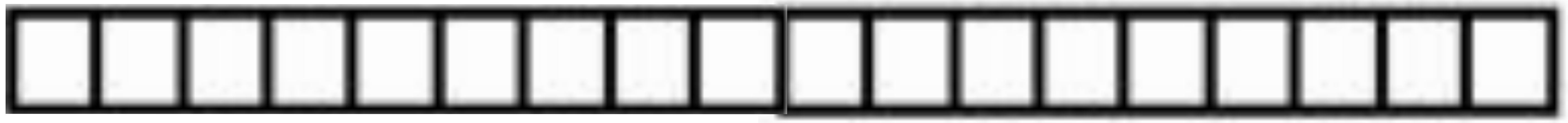


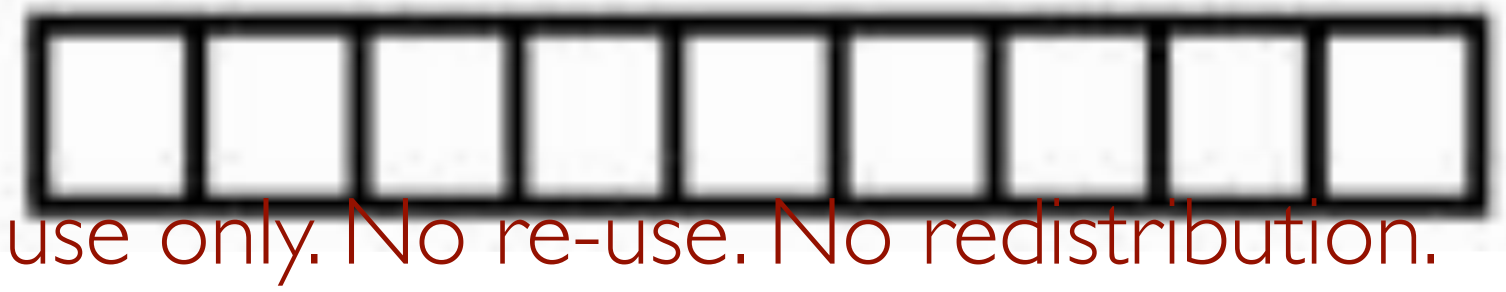
Acoustic feature engineering

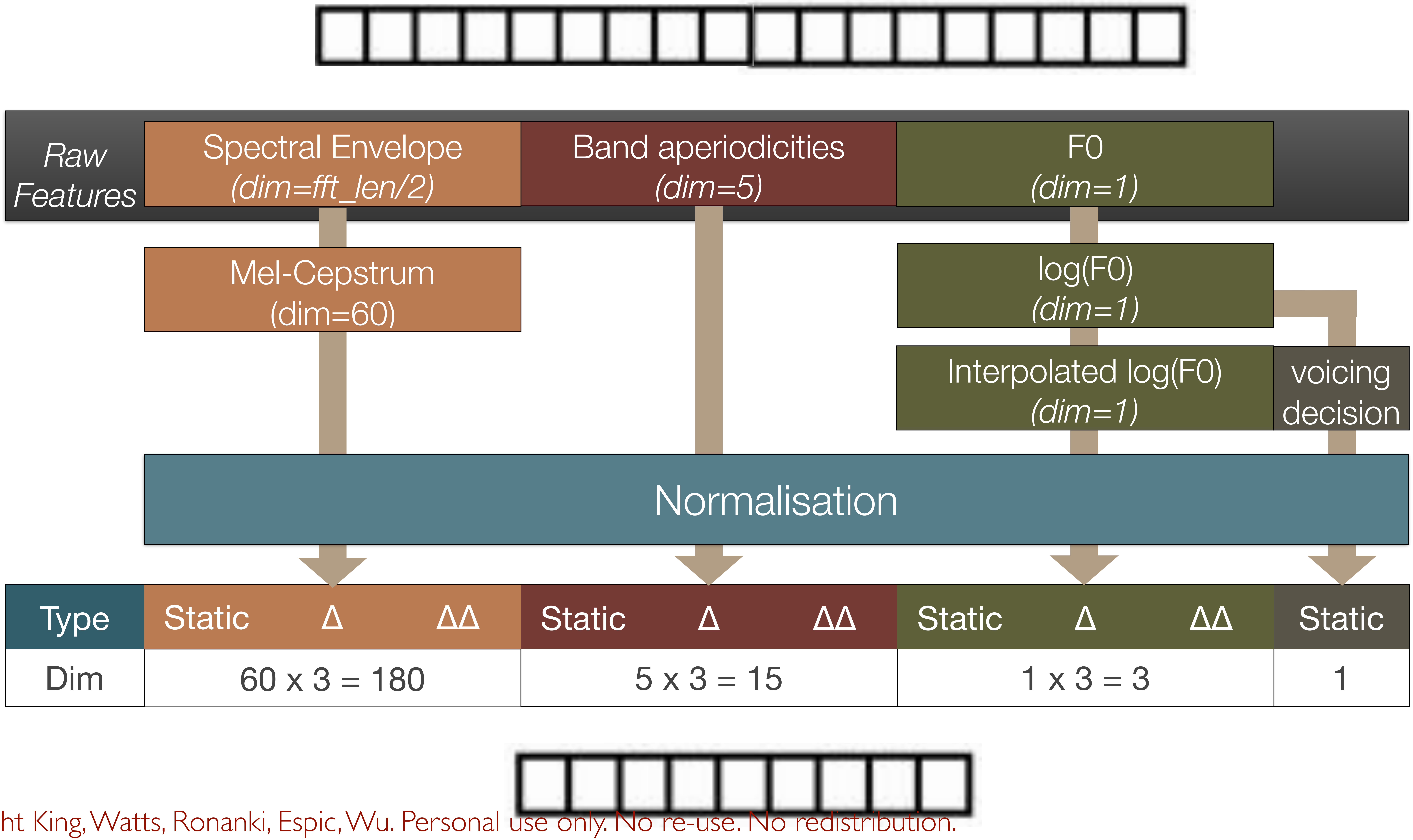


raw vocoder features

acoustic features







Raw Features

Spectral Envelope *(dim=fft_len/2)*

Band aperiodicities *(dim=5)*

F0 *(dim=1)*

Mel-Cepstrum (dim=60)

log(F0) *(dim=1)*

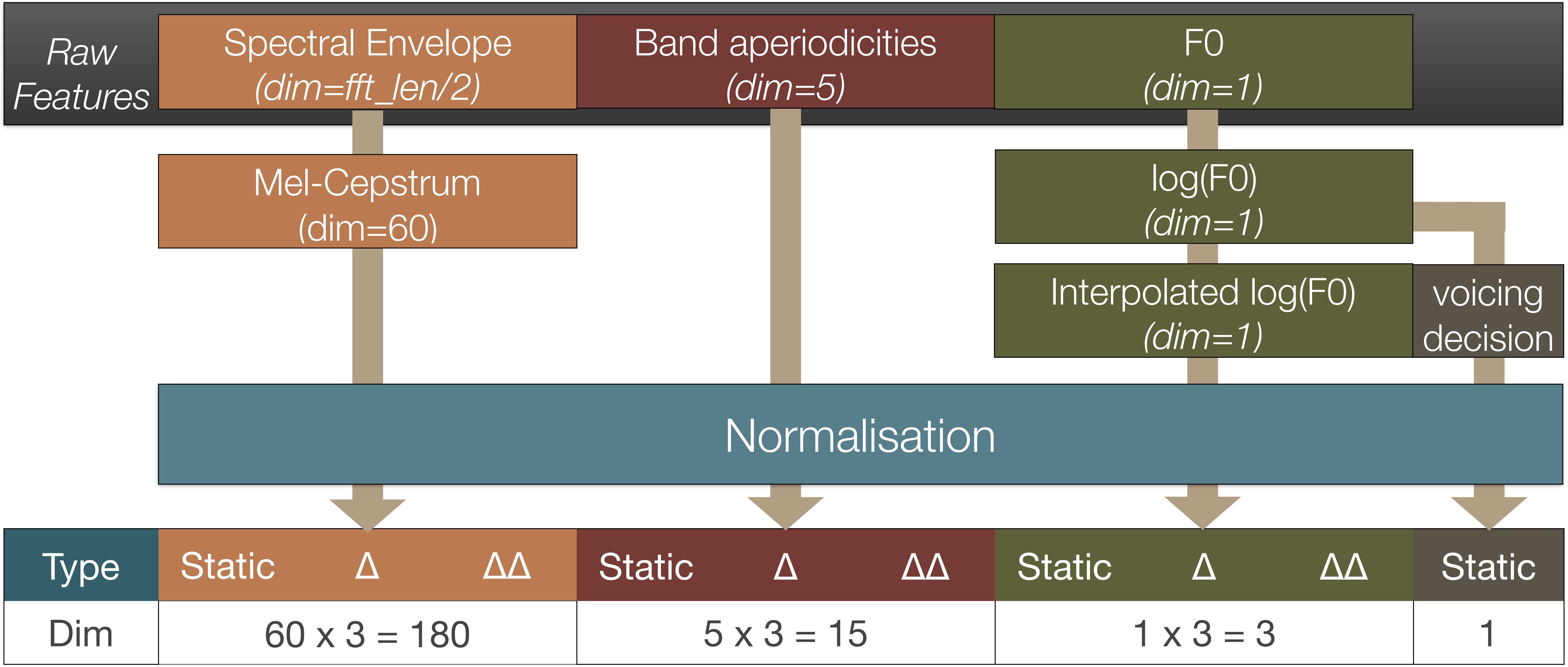
Interpolated log(F0) *(dim=1)*

voicing decision

Normalisation

Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering



Raw Features

Spectral Envelope
(dim=fft_len/2)

Band aperiodicities
(dim=5)

F0
(dim=1)

Mel-Cepstrum
(dim=60)

log(F0)
(dim=1)

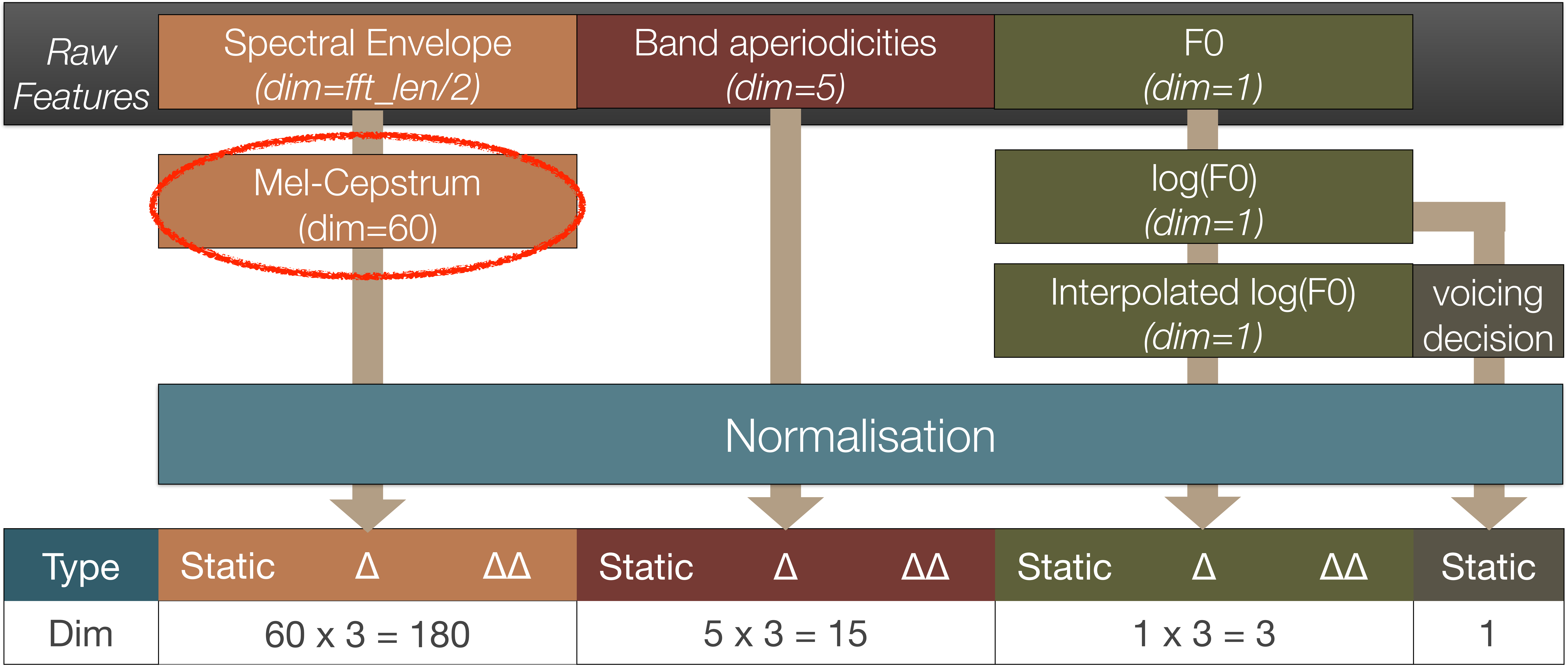
Interpolated log(F0)
(dim=1)

voicing decision

Normalisation

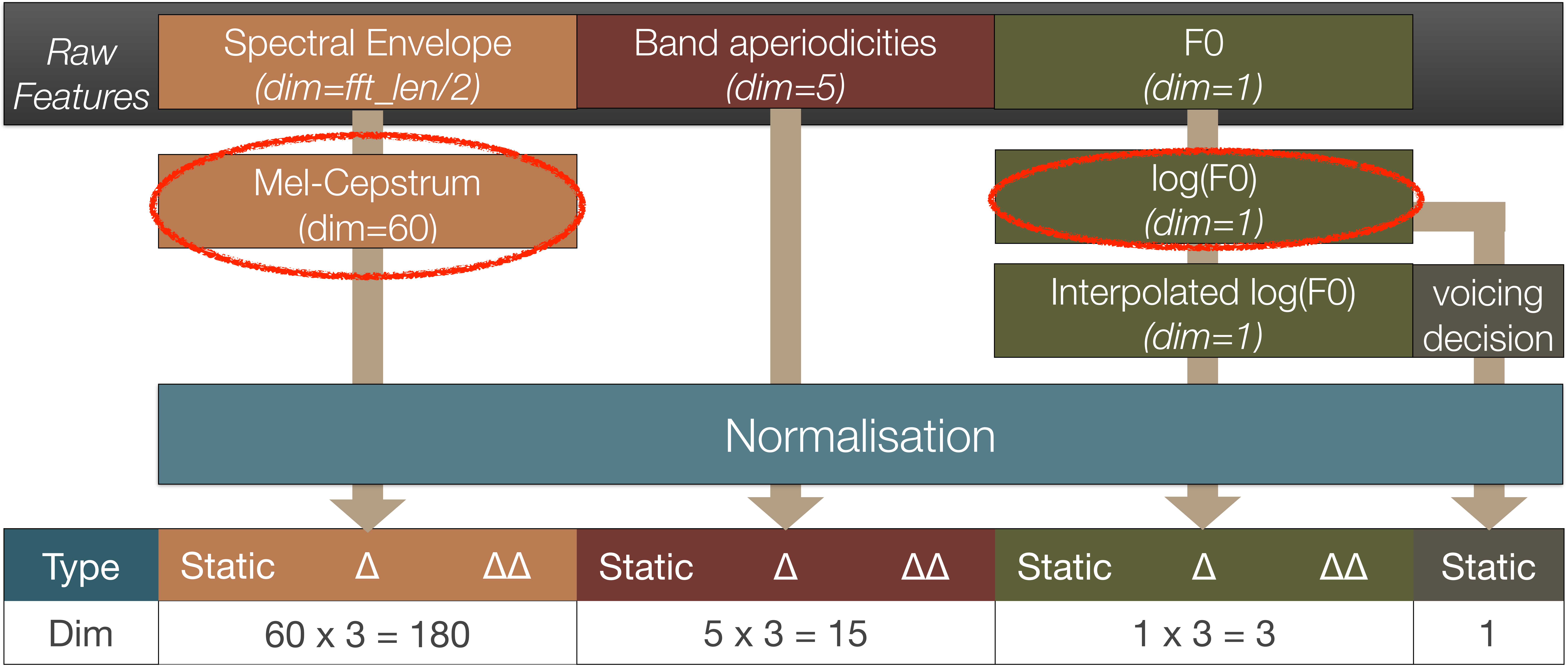
Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering



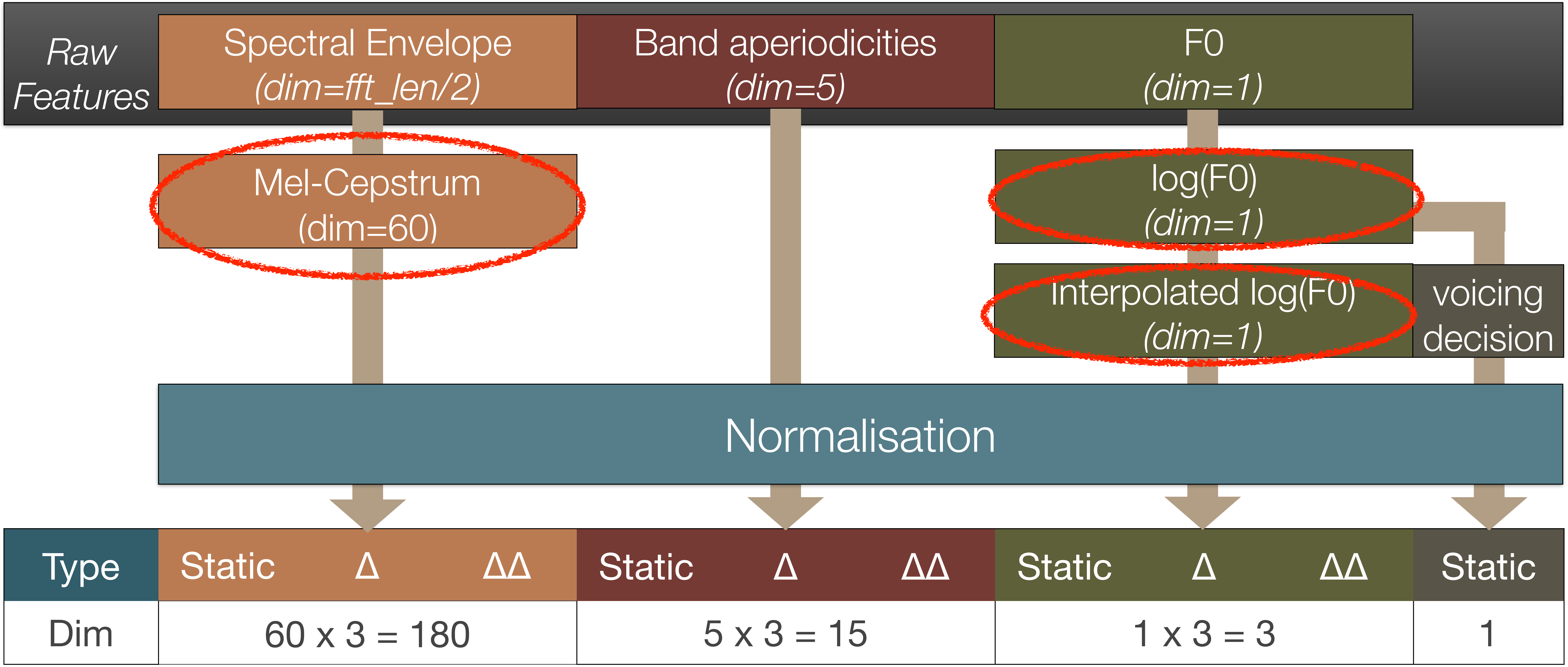
Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering


Raw Features
Spectral Envelope (dim=fft_len/2)
Band aperiodicities (dim=5)
F0 (dim=1)
Mel-Cepstrum (dim=60)
log(F0) (dim=1)
Interpolated log(F0) (dim=1)
voicing decision
Normalisation

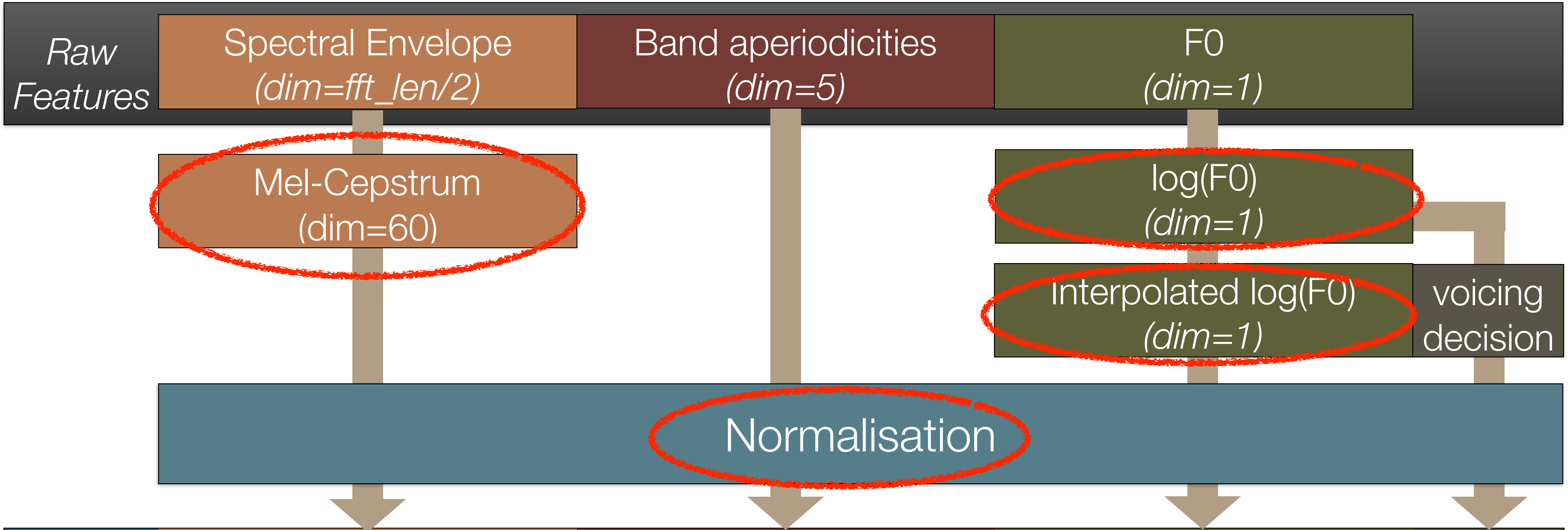
Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering



Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering



Raw Features

Spectral Envelope
(dim=fft_len/2)

Band aperiodicities
(dim=5)

F0
(dim=1)

Mel-Cepstrum
(dim=60)

log(F0)
(dim=1)

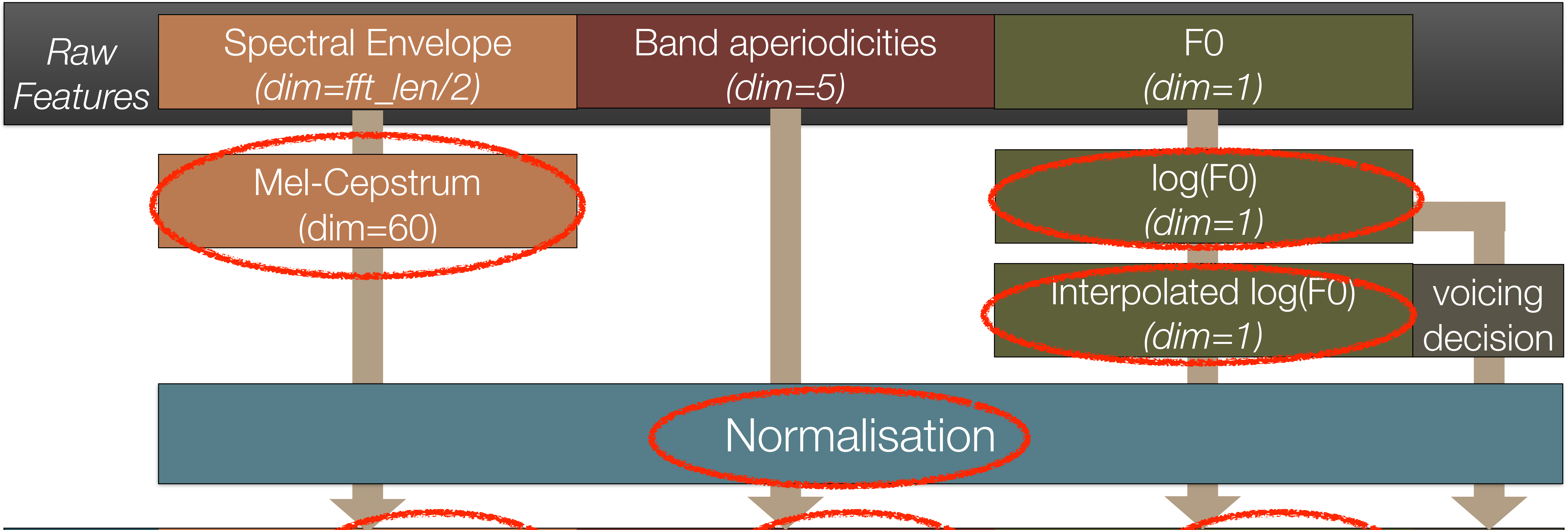
Interpolated log(F0)
(dim=1)

voicing decision

Normalisation

Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

Acoustic feature engineering



Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

03_prepare_acoustic_features.sh

```
python ${MerlinDir}/misc/scripts/vocoder/${Vocoder,,}/
extract_features_for_merlin.py ${MerlinDir} ${wav_dir} ${feat_dir} $SamplingFreq
```

extract_features_for_merlin.py

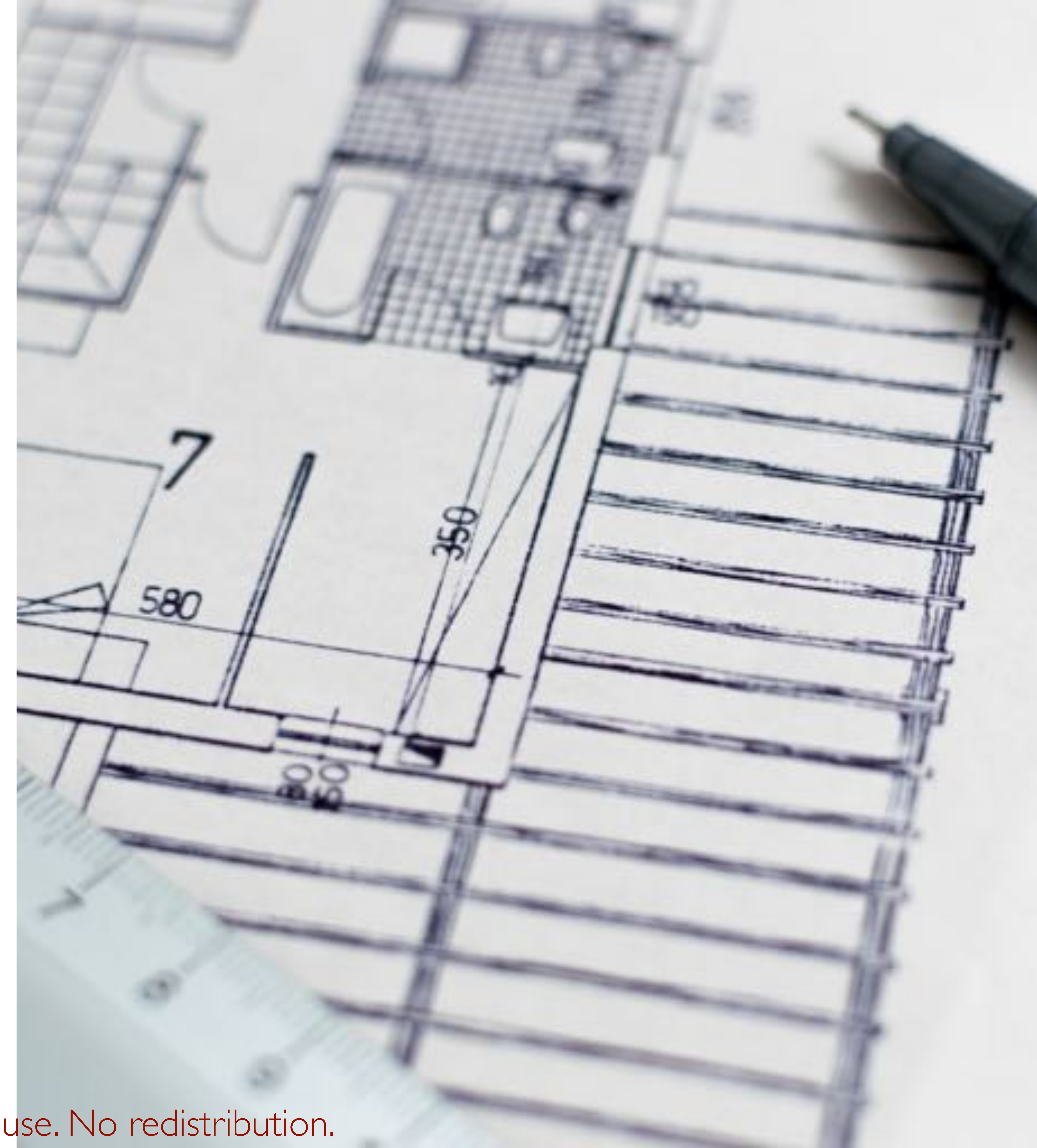
```
# tools directory
world = os.path.join(merlin_dir, "tools/bin/WORLD")
sptk  = os.path.join(merlin_dir, "tools/bin/SPTK-3.9")

if fs == 16000:
    nFFTHalf = 1024
    alpha = 0.58

elif fs == 48000:
    nFFTHalf = 2048
    alpha = 0.77

mcsize=59

world_analysis_cmd = "%s %s %s %s %s" % (os.path.join(world, 'analysis'), \
                                         filename,
                                         os.path.join(f0_dir, file_id + '.f0'), \
                                         os.path.join(sp_dir, file_id + '.sp'), \
                                         os.path.join(bap_dir, file_id + '.bapd'))
os.system(world_analysis_cmd)

### convert f0 to lf0 ###
sptk_x2x_da_cmd = "%s +da %s > %s" % (os.path.join(sptk, 'x2x'), \
```

extract_features_for_merlin.py

```
                      os.path.join(f0_dir, file_id + '.f0a'), \
                      os.path.join(sptk, 'sopr') + ' -magic 0.0 -LN -MAGIC
-1.0E+10', \

                      os.path.join(lf0_dir, file_id + '.lf0'))
os.system(sptk_x2x_af_cmd)

### convert sp to mgc ###
sptk_x2x_df_cmd1 = "%s +df %s | %s | %s >%s" % (os.path.join(sptk, 'x2x'), \
                        os.path.join(sp_dir, file_id + '.sp'), \
                        os.path.join(sptk, 'sopr') + ' -R -m 32768.0', \
                        os.path.join(sptk, 'mcep') + ' -a ' + str(alpha
' -m ' + str(

                          mcsize) + ' -l ' + str(
                          nFFTHalf) + ' -e 1.0E-8 -j 0 -f 0.0 -q 3 ',
                        os.path.join(mgc_dir, file_id + '.mgc'))
os.system(sptk_x2x_df_cmd1)

### convert bapd to bap ###
sptk_x2x_df_cmd2 = "%s +df %s > %s " % (os.path.join(sptk, "x2x"), \
                    os.path.join(bap_dir, file_id + ".bapd"), \
                    os.path.join(bap_dir, file_id + '.bap'))
os.system(sptk_x2x_df_cmd2)
```

Design choices: acoustic features

- fixed framerate *or* pitch synchronous
- cepstrum *or* spectrum
- linear *or* warped frequency (e.g., Mel)
- order
- interpolate F0
- phase modelling
  - no: e.g., Tacotron
  - yes: e.g., Espic, Valentini-Botinhao, King, Interspeech 2017



Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**

- Input
  - linguistic features

- Output
  - acoustic features

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**
- Input
  - linguistic features
- Output
  - acoustic features

Orientation

- Defining the problem of TTS
  - **sequence-to-sequence regression**

- Input
  - linguistic features ✓

- Output
  - acoustic features ✓

Agenda

	Topic	**Presenter**
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	**Zhizheng Wu**
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

What next?

- we spent **a lot of time** preparing the input and output features
- but that reflects the reality of building a DNN-based system

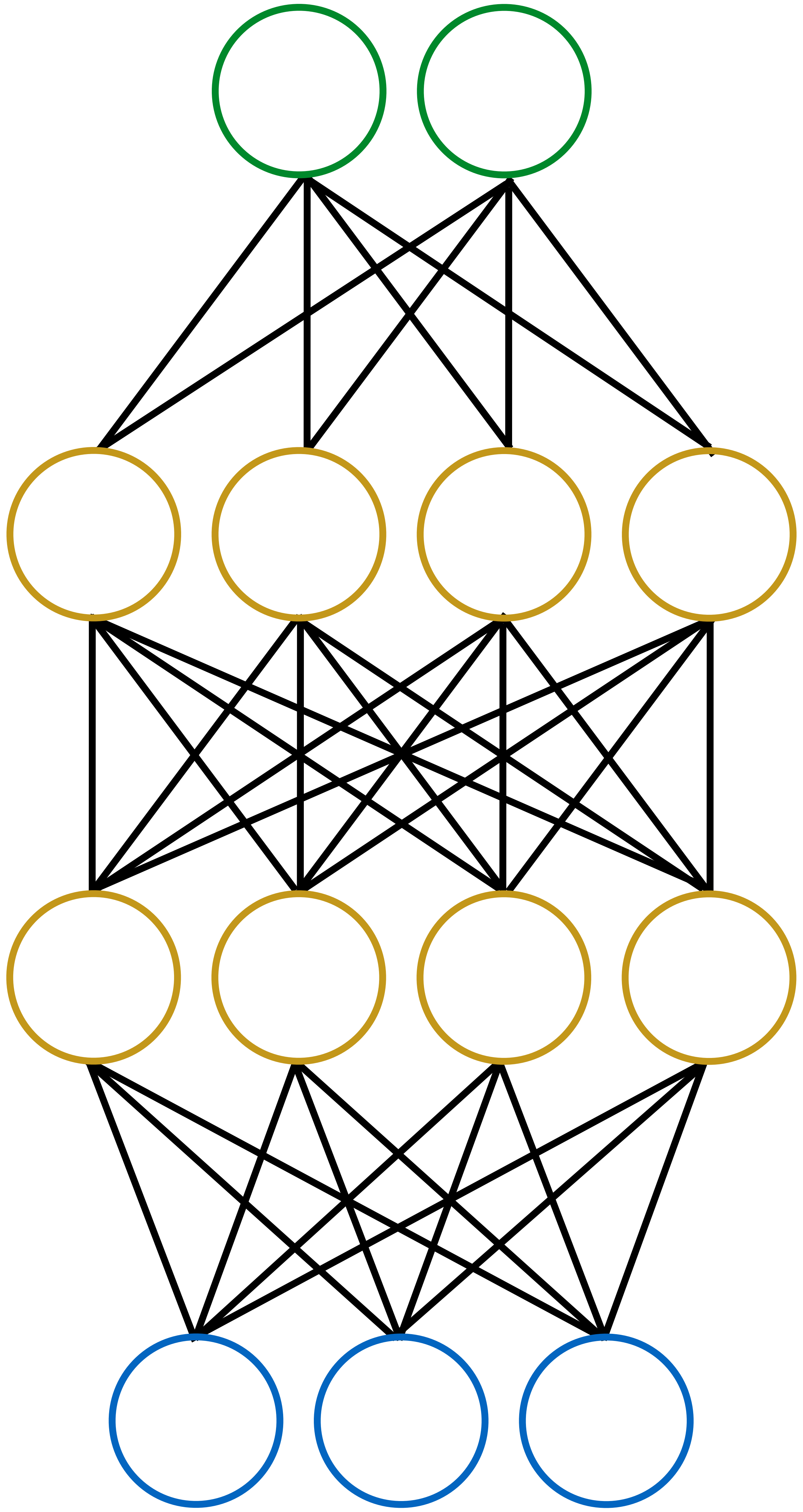
- Next
  - actually doing the regression !

Regression

Zhizheng Wu

Feed-forward

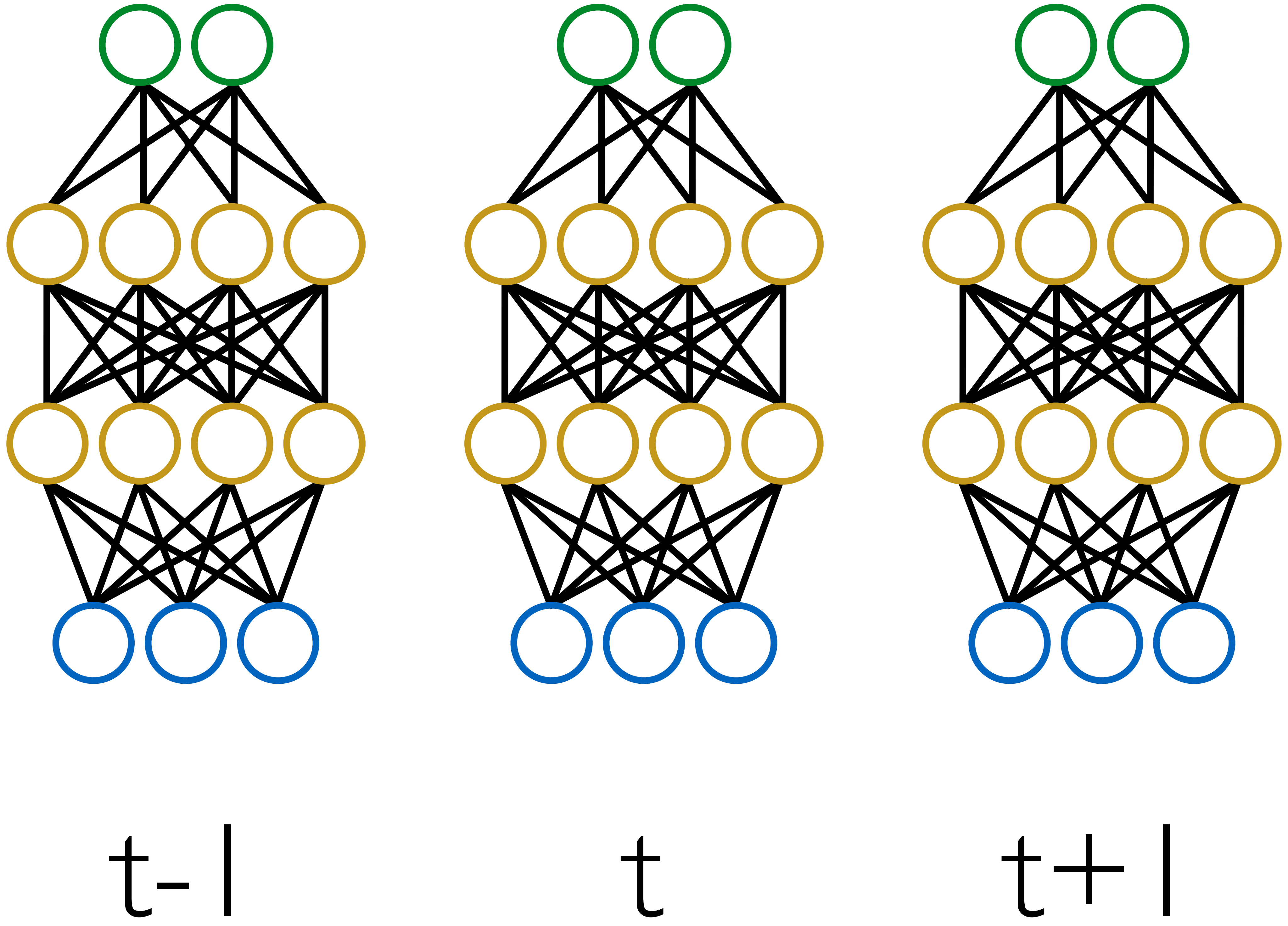
- Conceptually straightforward
- **For each input frame**
  - perform regression to corresponding output features
- To provide wider input context, could simply stack several frames together
  - although, remember that the linguistic features already span several timescales



Recurrent (naive version)

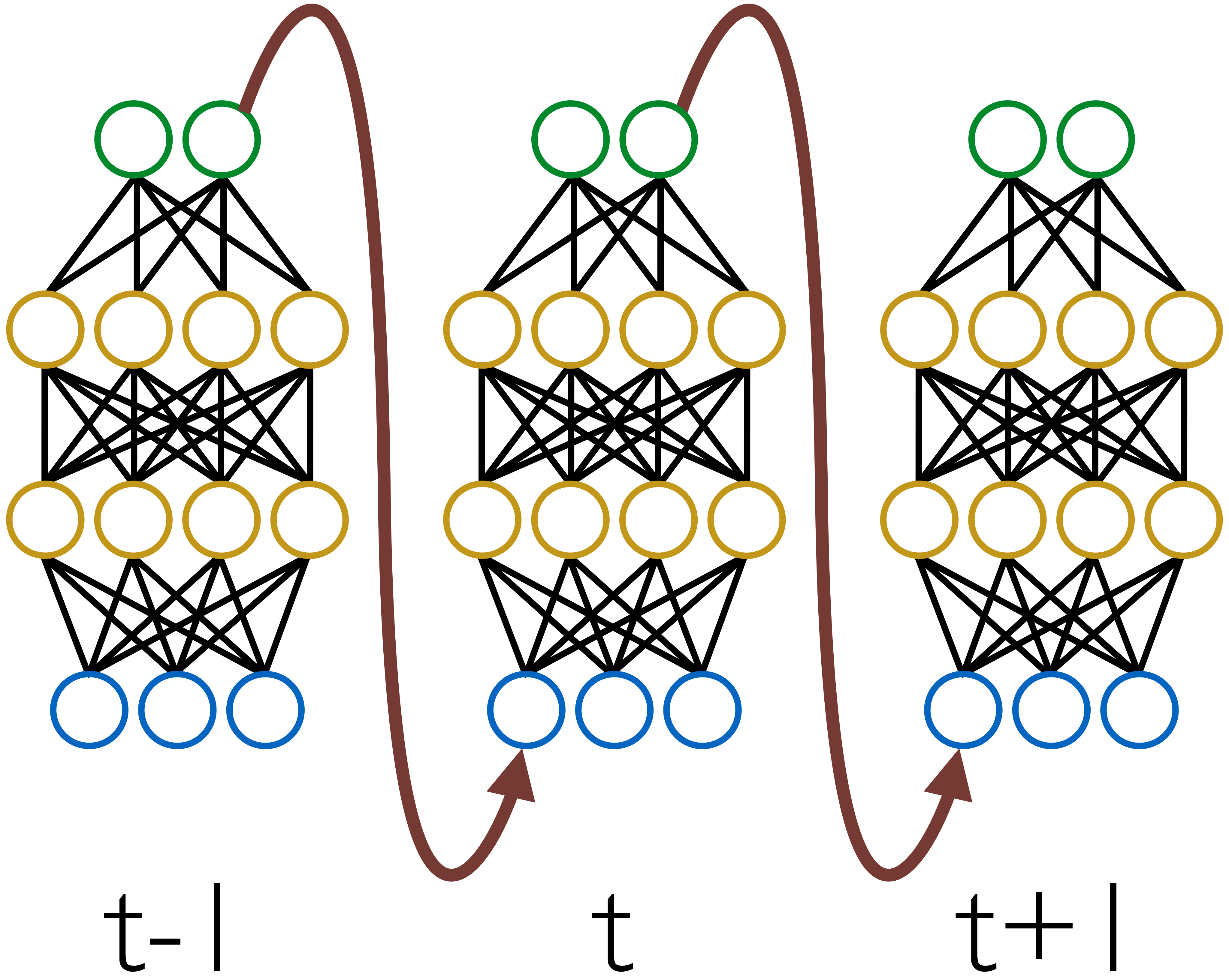
- Pass some of the outputs (or hidden layer activations) forwards in time, typically to the next time step
- A kind of **memory**
- Provides "infinite" left context

- (could also pass information backwards in time)



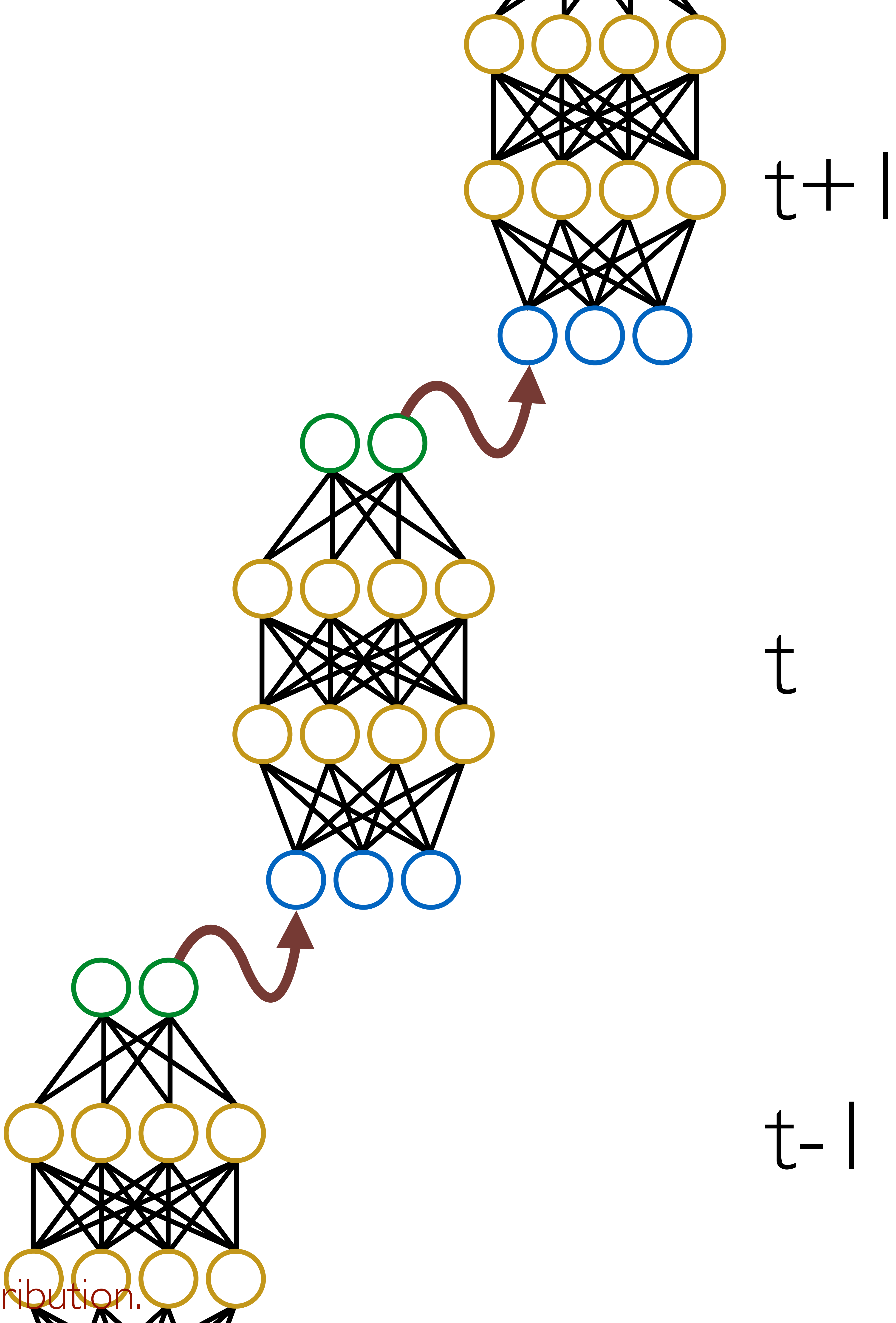
Recurrent (naive version)

- 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)



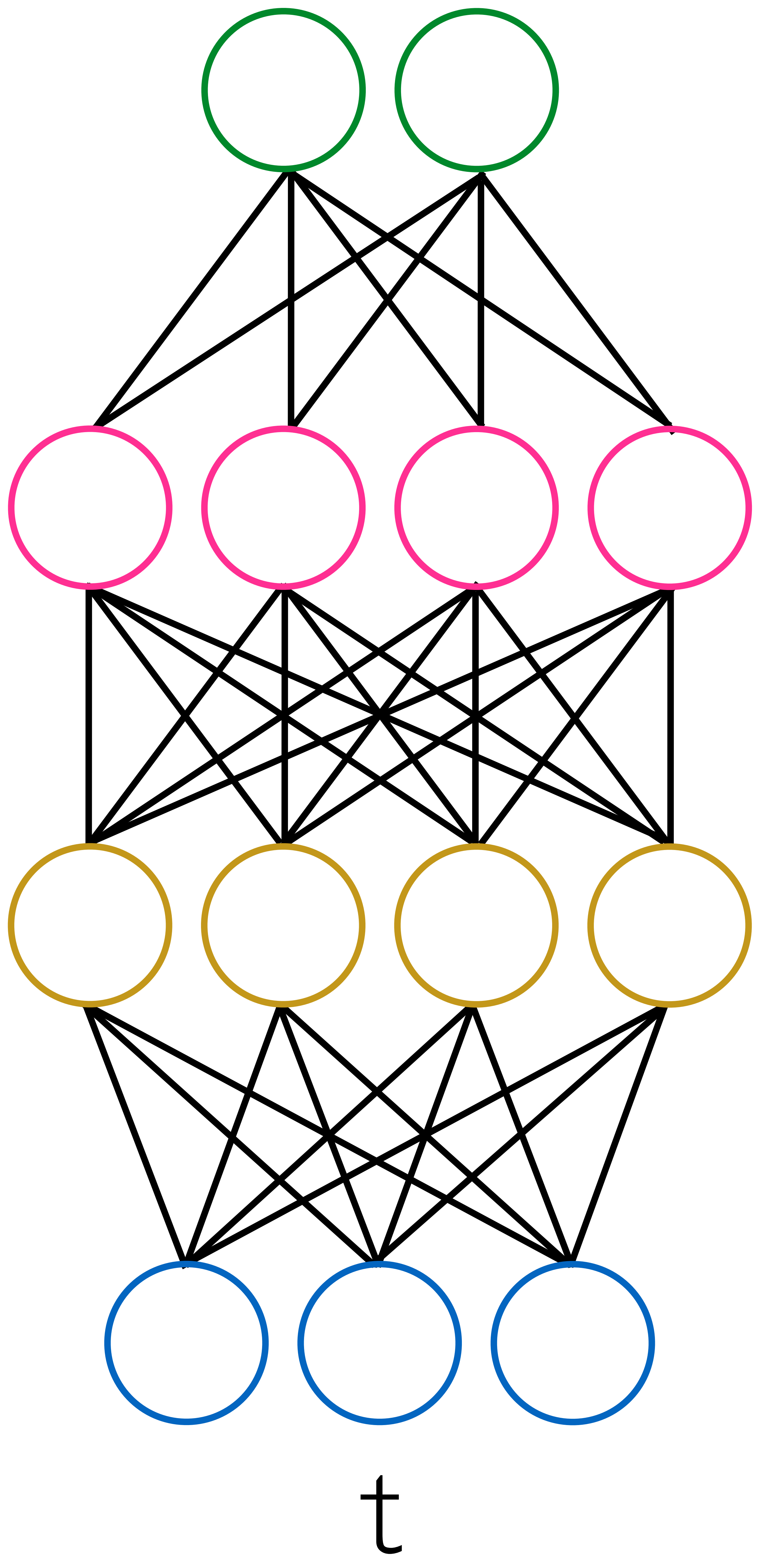
Recurrent

- 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



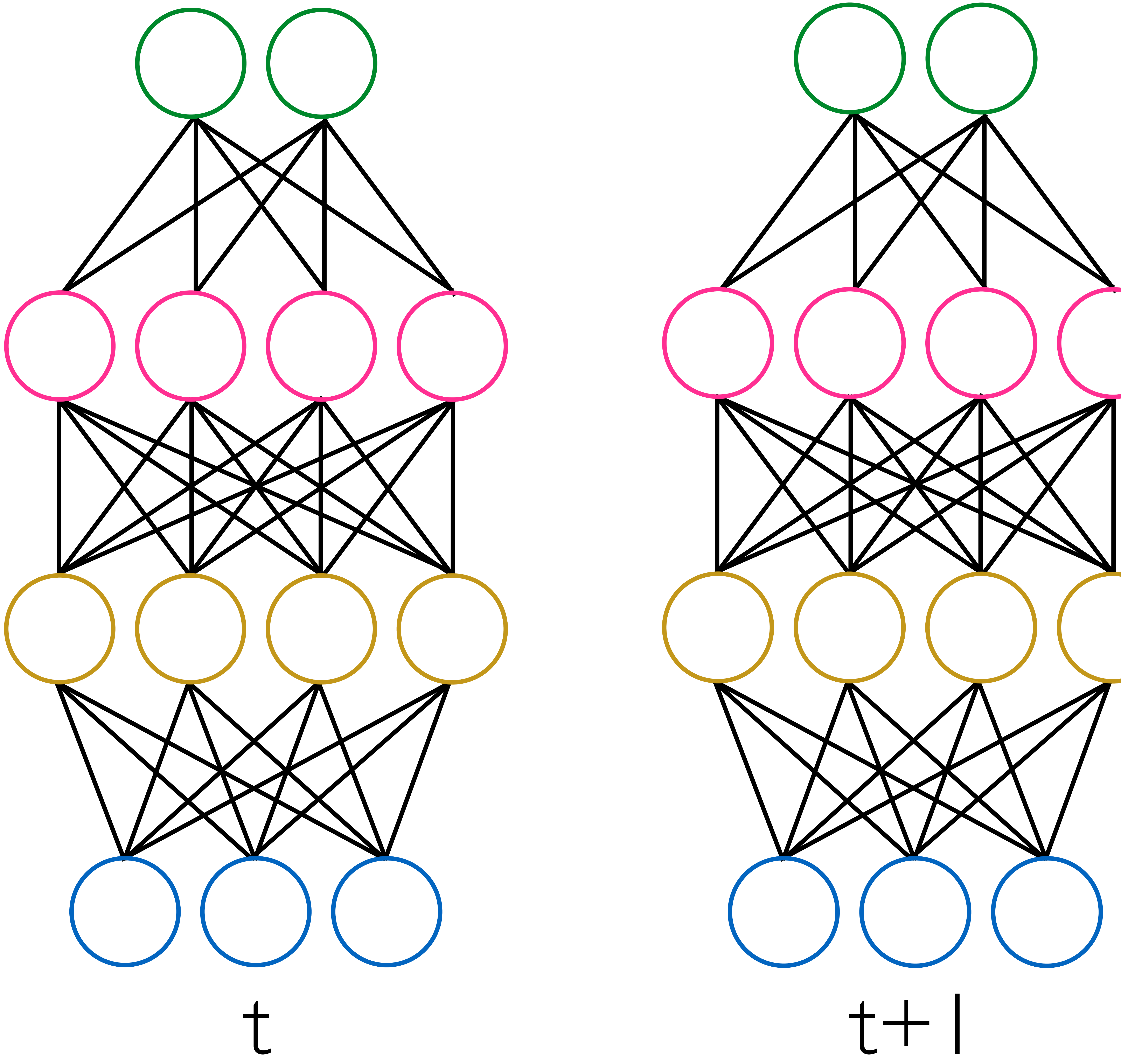
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- <u>Conceptually</u>
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**



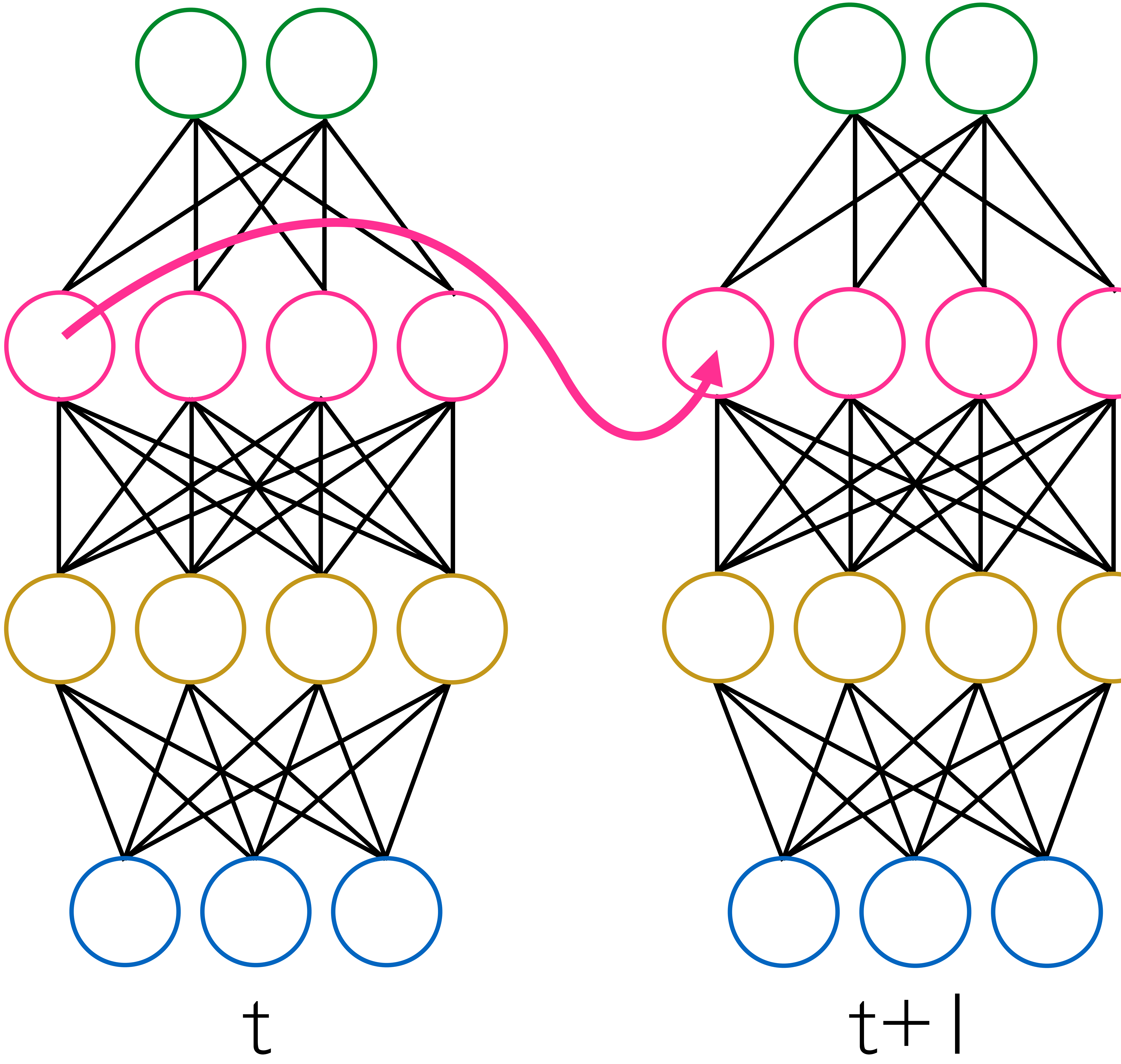
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- Conceptually
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**



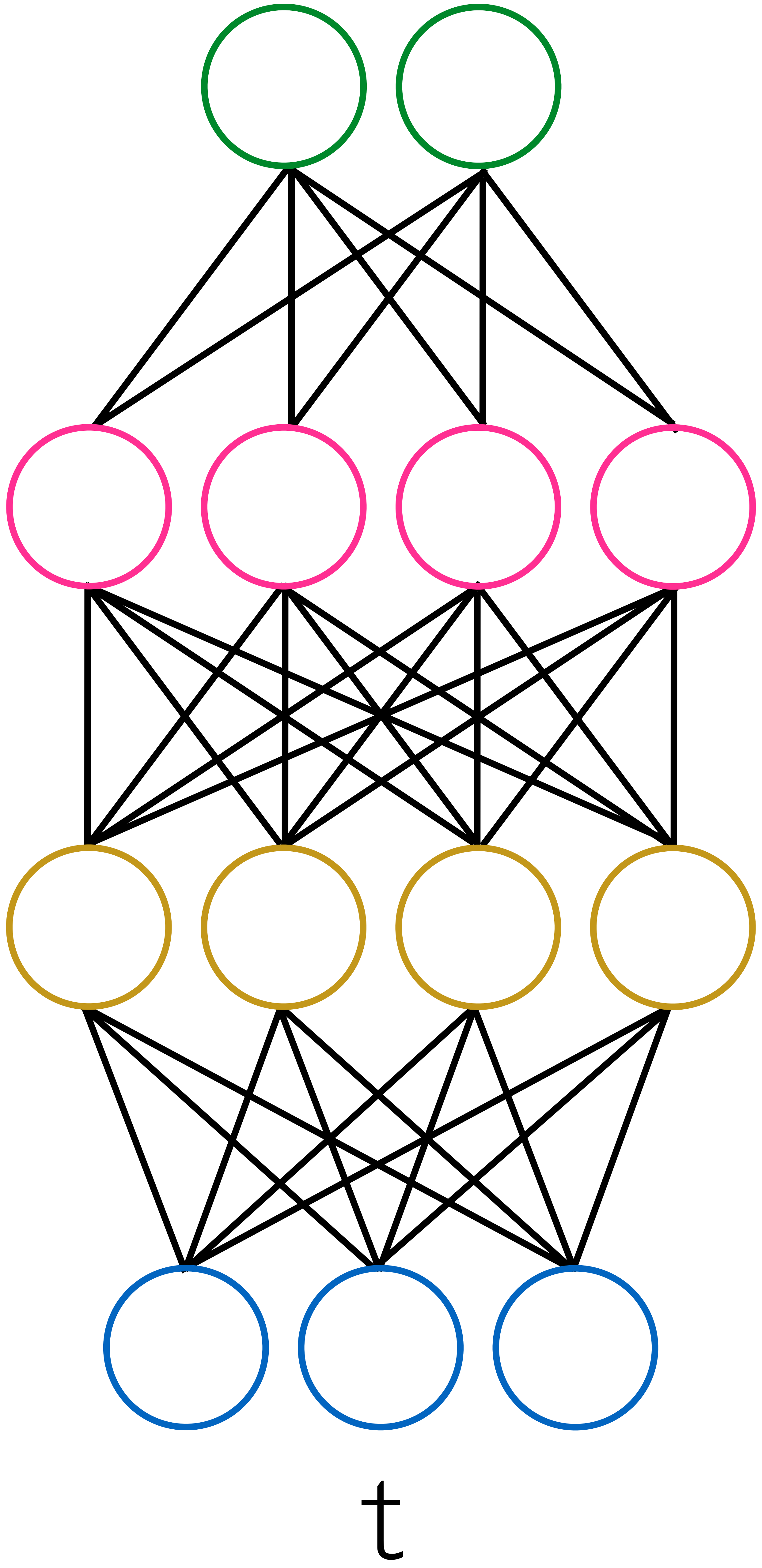
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- Conceptually
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**



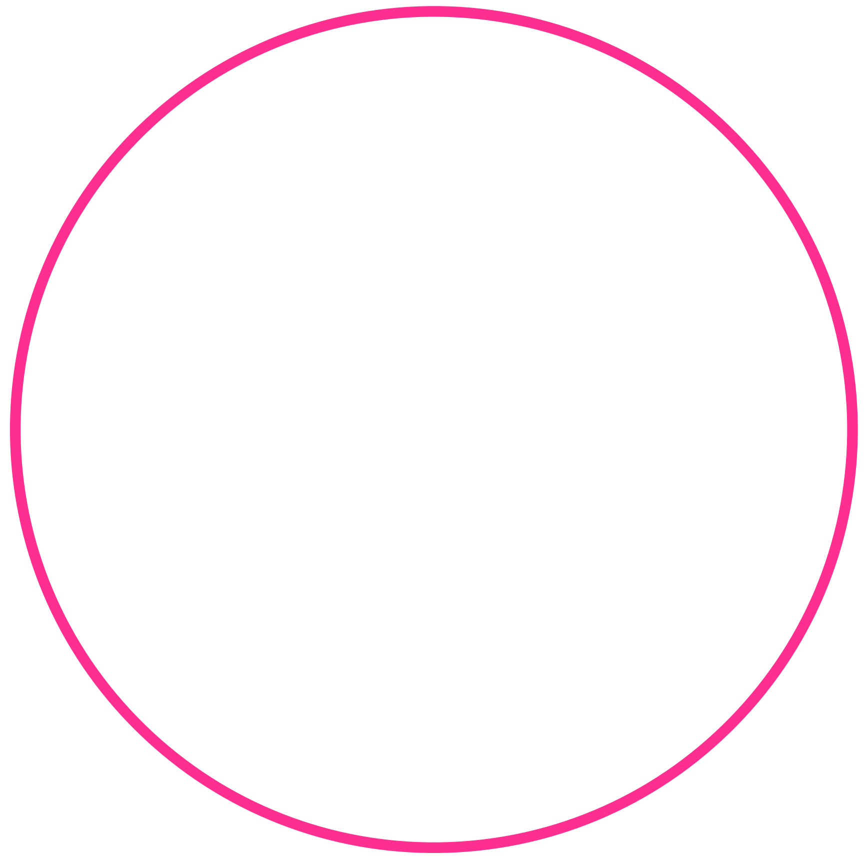
Long short-term memory (a type of recurrence)

- Solves the vanishing gradient problem by using "gates" to control the flow of information
- Conceptually
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn 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
- Conceptually
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn 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>
  - Special LSTM units
    - learn when to **remember**
    - remember information for any number of time steps
    - learn when to **forget**

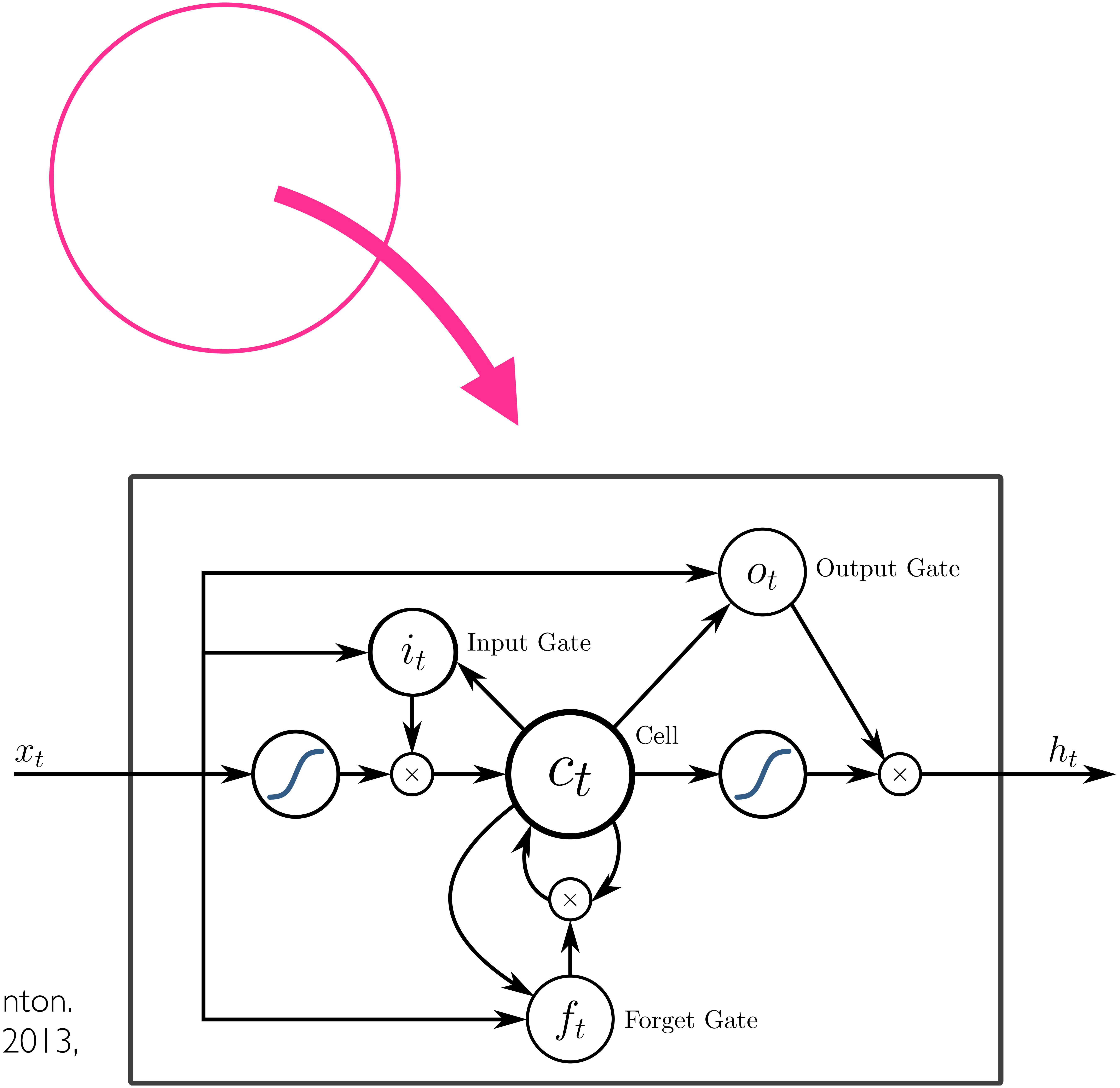


Figure from Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks" ICASSP 2013, redrawn as SVG by Eddie Antonio Santos

Orientation

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

Sequence-to-sequence

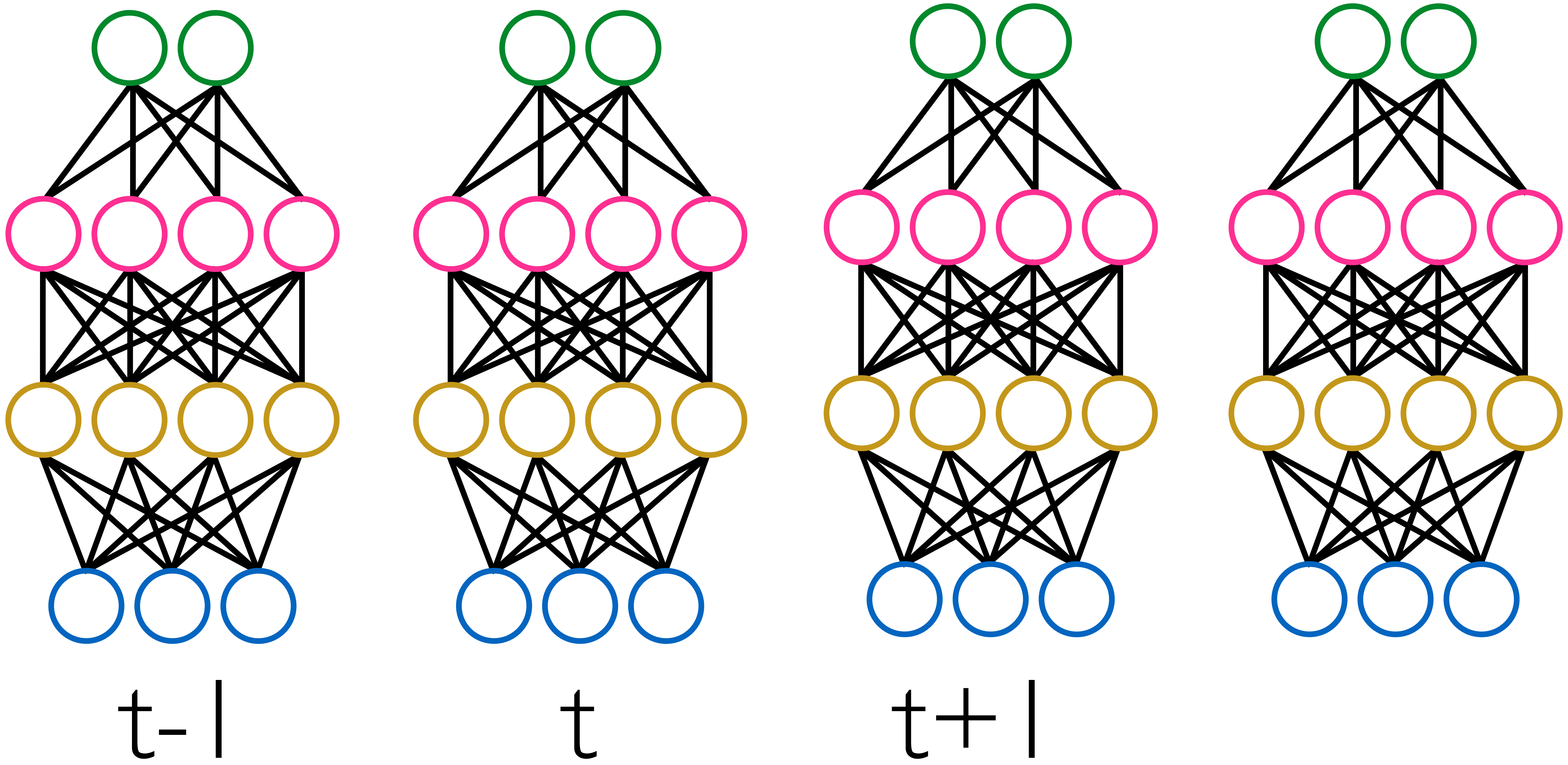
- Next step is to integrate the alignment mechanism into the network itself
- Now, length of input sequence may be **different** to length of output sequence

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

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

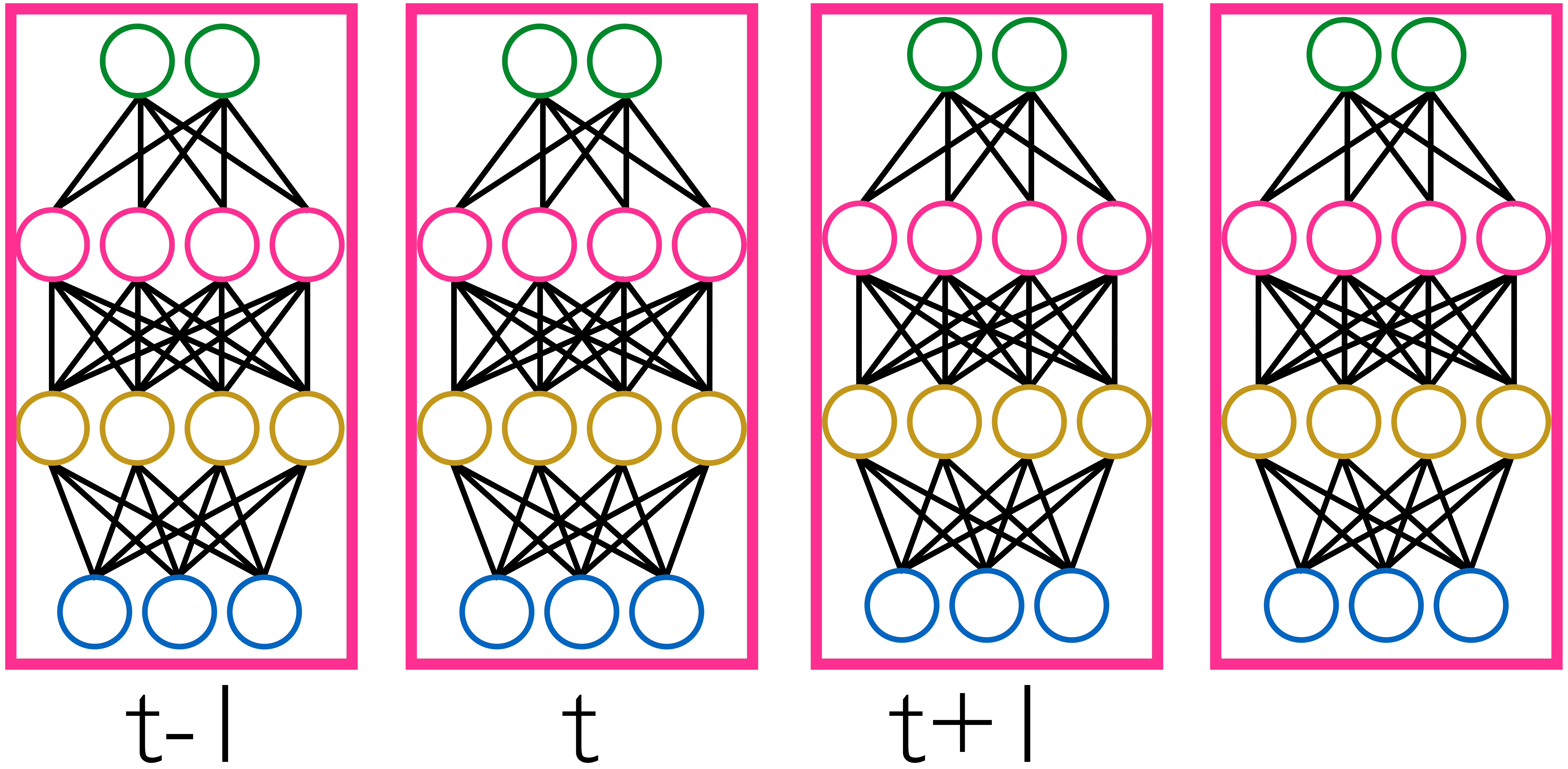
Sequence-to-sequence (just conceptually)

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

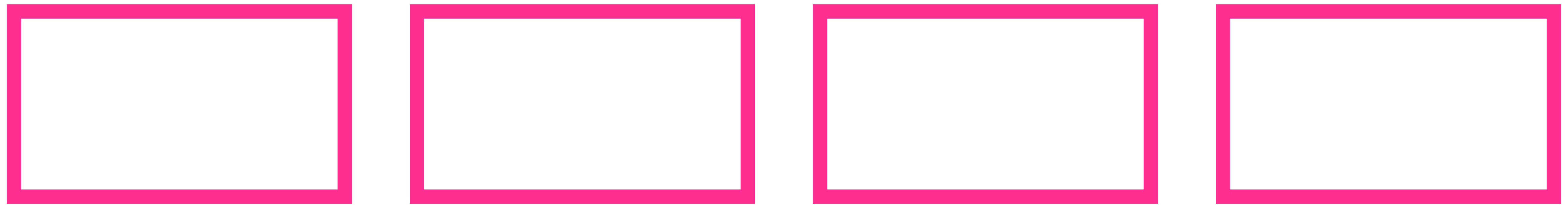


Sequence-to-sequence (just conceptually)

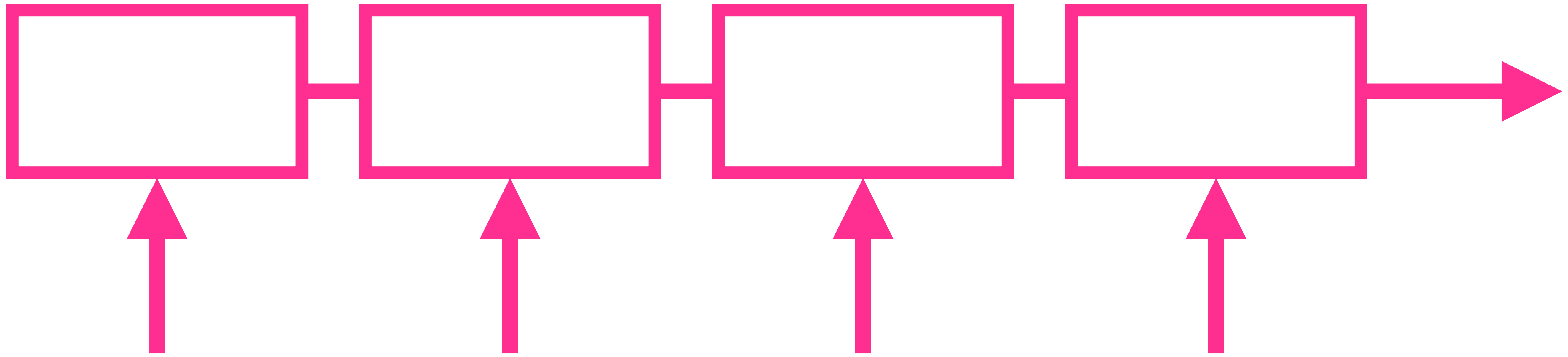
- The **encoder**
- A recurrent network that "reads" the entire input sequence and "summarises" or "memorises" it using a fixed-length representation

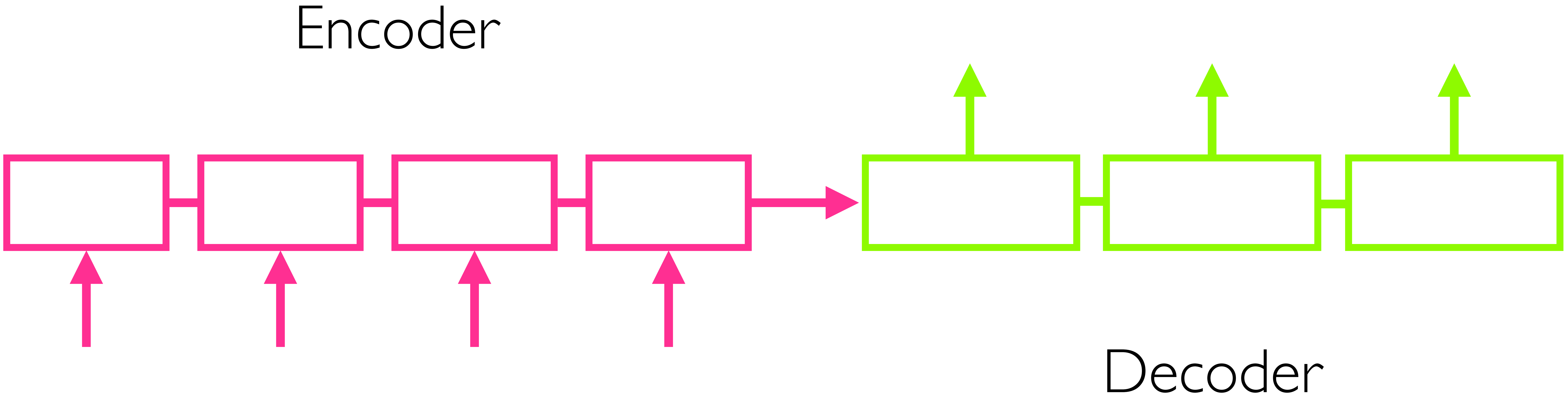


Encoder



Encoder




Encoder
Decoder

Alignment in sequence-to-sequence models

- Basic model, as presented, has **no alignment** between input and output
- Get better performance by adding **“attention”**
  - decoder has access to the input sequence
  - decoder typically also has access to its own output at previous time step
- **Alignment is like ASR. Doing ASR with vocoder features does not work well!**
  - so, we expect better performance by using ASR-style acoustic features (just for the alignment part of the model)

- This is as far as we want to go in this tutorial, regarding different DNN topologies for TTS
- The field is fast moving

04_prepare_conf_files.sh

```
echo "preparing config files for acoustic, duration models..."
./scripts/prepare_config_files.sh $global_config_file

echo "preparing config files for synthesis..."
./scripts/prepare_config_files_for_synthesis.sh $global_config_file
```

05_train_duration_model.sh

```
./scripts/submit.sh ${MerlinDir}/src/run_merlin.py $duration_conf_file
```

config files

```
[DEFAULT]

Merlin: <path to Merlin root directory>

TOPLEVEL: <path where experiments are created>

[Paths]

# where to place work files
work: <path where data, log, models and generated data are stored and created>

# where to find the data
data: %(work)s/data

# where to find intermediate directories
inter_data: %(work)s/inter_module

# list of file basenames, training and validation in a single list
file_id_list: %(data)s/file_id_list.scp
test_id_list: %(data)s/test_id_list.scp

in_mgc_dir: %(data)s/mgc
in_bap_dir: %(data)s/bap
```

config files

```
[Labels]
enforce_silence: False
silence_pattern: ['*-sil+*']
# options: state_align or phone_align
label_type: state_align
label_align: <path to labels>
question_file_name: <path to questions set>

add_frame_features: True

# options: full, coarse_coding, minimal_frame, state_only, frame_only, none
subphone_feats: full

[Outputs]
# dX should be 3 times X
mgc      : 60
dmgc     : 180
bap      : 1
dbap     : 3
lf0      : 1
dlf0     : 3

[Waveform]
```

config files

```
[Outputs]
# dX should be 3 times X
mgc     : 60
dmgc    : 180
bap     : 1
dbap    : 3
lf0     : 1
dlf0    : 3

[Waveform]
test_synth_dir: None
# options: WORLD or STRAIGHT
vocoder_type: WORLD
samplerate: 16000
framelength: 1024
# Frequency warping coefficient used to compress the spectral envelope into MGC (or MCEP)
fw_alpha: 0.58
minimum_phase_order: 511
use_cep_ap: True

[Architecture]
switch_to_keras: False
hidden_layer_size  : [1024, 1024, 1024, 1024, 1024, 1024]
```

config files

```
[Architecture]
switch_to_keras: False
hidden_layer_size  : [1024, 1024, 1024, 1024, 1024, 1024]
hidden_layer_type  : ['TANH', 'TANH', 'TANH', 'TANH', 'TANH', 'TANH']

model_file_name: feed_forward_6_tanh

#if RNN or sequential training is used, please set sequential_training to True.
sequential_training : False

dropout_rate : 0.0
batch_size   : 256

# options: -1 for exponential decay, 0 for constant learning rate, 1 for linear decay
lr_decay      : -1
learning_rate : 0.002

# options: sgd, adam, rprop
optimizer : sgd

warmup_epoch    : 10
training_epochs : 25
```

config files

```
[Processes]

# Main processes

AcousticModel : True
GenTestList : False

# sub-processes

NORMLAB  : True
MAKECMP  : True
NORMCMP  : True

TRAINDNN : True
DNNGEN   : True

GENWAV   : True
CALMCD   : True
```

06_train_acoustic_model.sh

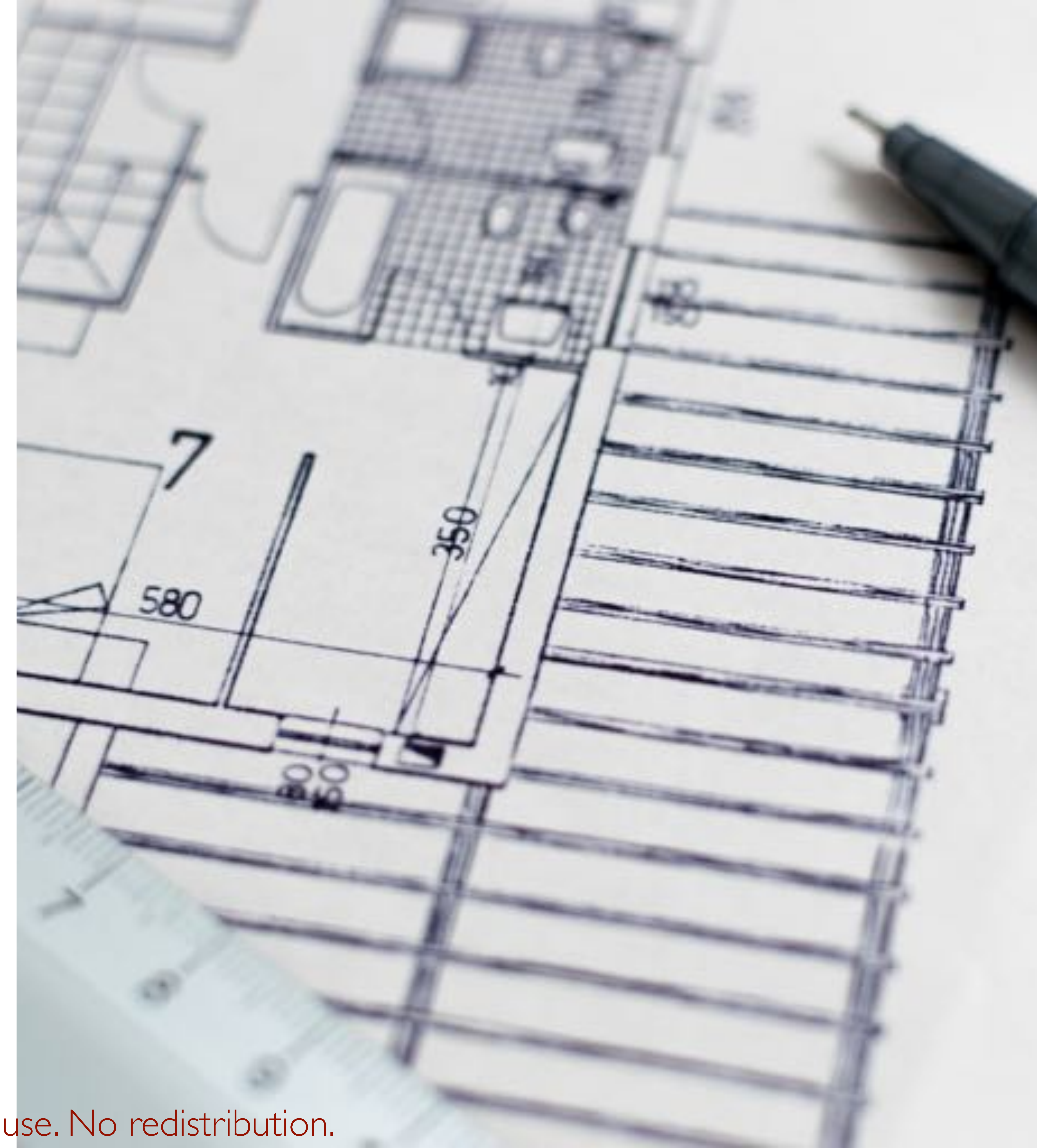
```
./scripts/submit.sh ${MerlinDir}/src/run_merlin.py $acoustic_conf_file
```

07_run_merlin.sh

```
inp_txt=$1
test_dur_config_file=$2
test_synth_config_file=$3

echo "preparing full-contextual labels using Festival frontend..."
lab_dir=$(dirname $inp_txt)
./scripts/prepare_labels_from_txt.sh $inp_txt $lab_dir $global_config_file

echo "synthesizing durations..."
./scripts/submit.sh ${MerlinDir}/src/run_merlin.py $test_dur_config_file

echo "synthesizing speech..."
./scripts/submit.sh ${MerlinDir}/src/run_merlin.py $test_synth_config_file
```

Design choices: acoustic model

- Straightforward, if the input and output sequences are **the same length and aligned**
  - feedforward
  - recurrent (e.g. LSTM) layer(s)
- Less straightforward, for **unaligned** input and output sequences
  - sequence-to-sequence
- **The only practical limitation is what your chosen backend can do** (e.g., Theano, Tensorflow)



Orientation

- What is the output of the regression?
  - acoustic features
  - **not** a speech waveform

so there is one more step

- Generating the waveform
  - input is the acoustic features
  - output is the speech waveform

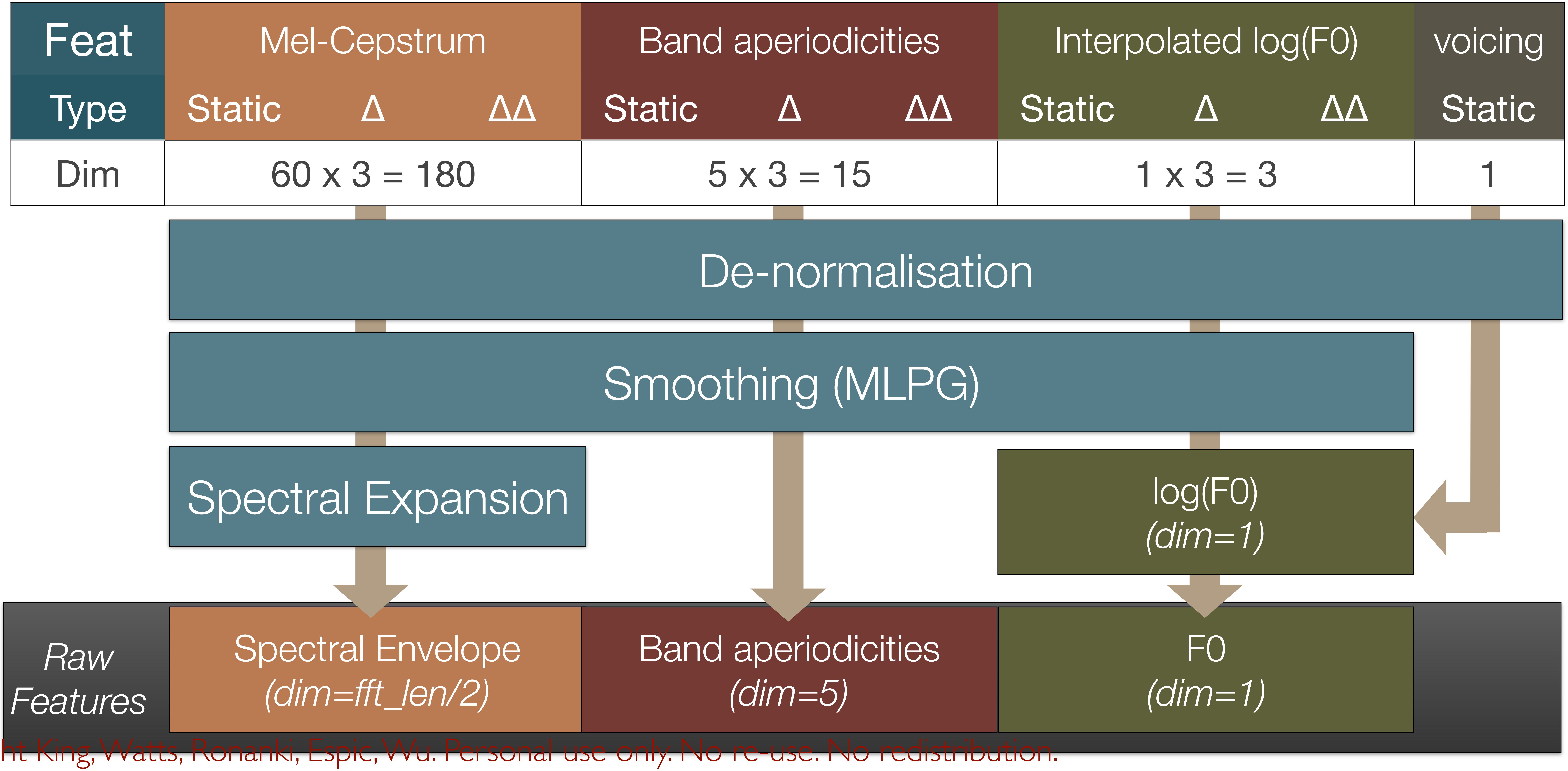
Agenda

	Topic	Presenter
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	**Felipe Espic**
	Recap and conclusion	Simon King
PART 3	Extensions	Zhizheng Wu

Waveform generation

Felipe Espic

From acoustic features back to raw vocoder features



Feat	Mel-Cepstrum			Band aperiodicities			Interpolated log(F0)			voicing
Type	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static	Δ	ΔΔ	Static
Dim	60 x 3 = 180			5 x 3 = 15			1 x 3 = 3			1

De-normalisation

Smoothing (MLPG)

Spectral Expansion

log(F0)
(dim=1)

Raw Features

Spectral Envelope
(dim=fft_len/2)

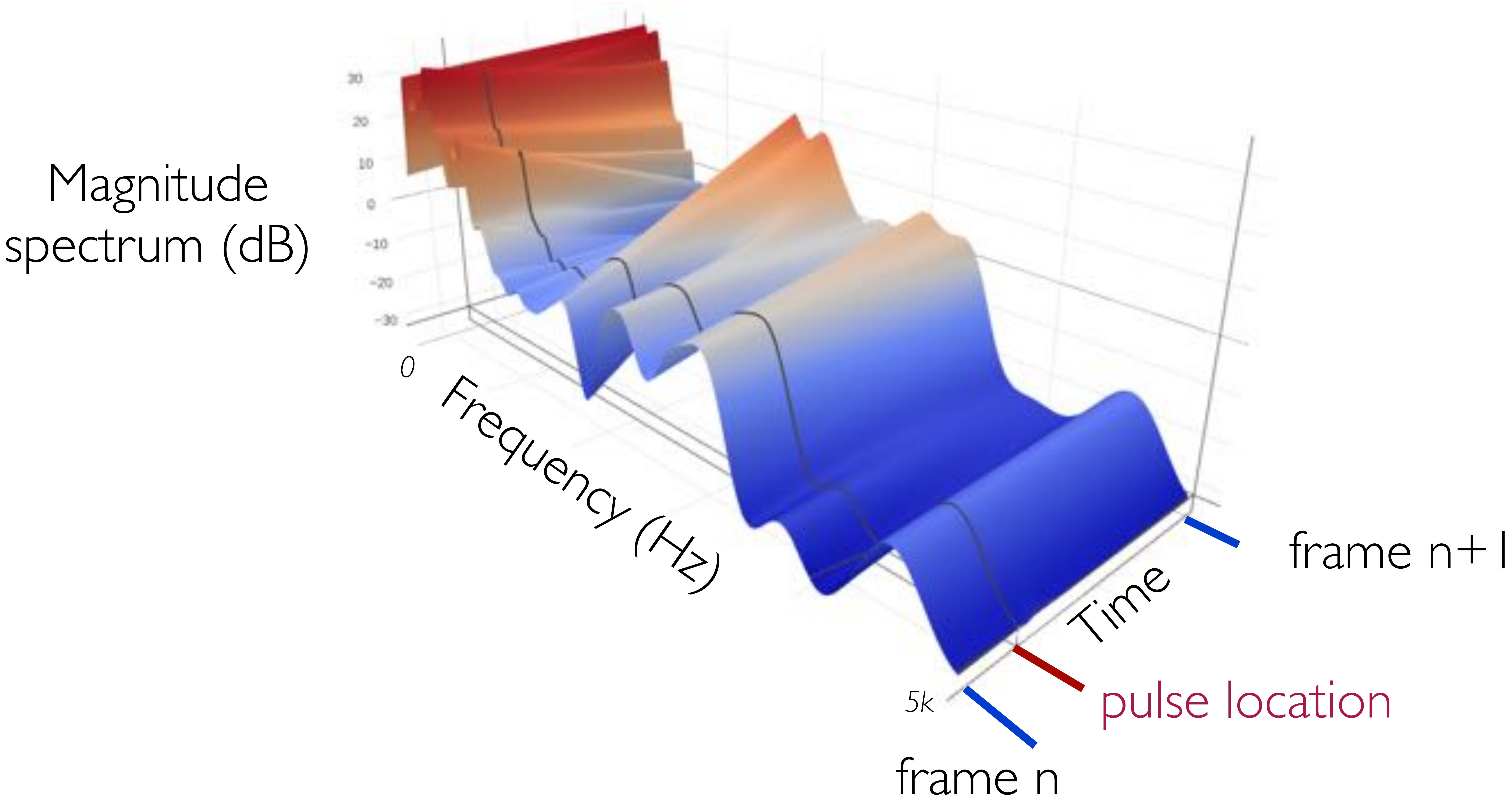
Band aperiodicities
(dim=5)

F0
(dim=1)

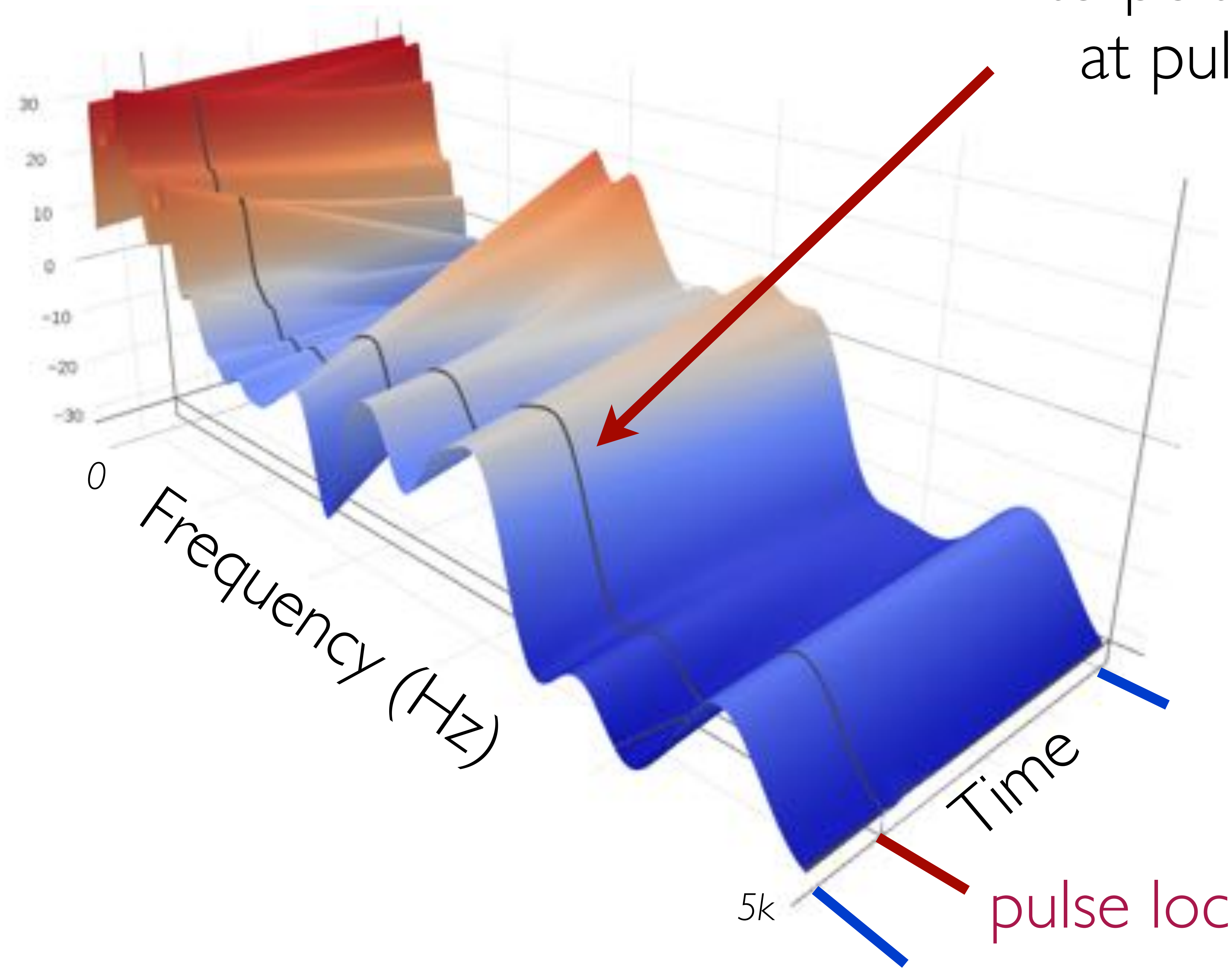
WORLD: periodic excitation using a pulse train

- Computation of pulse locations
  - Voiced segments: create one pulse every **fundamental period**, T0
    - calculate T0 from F0, which has been predicted by the acoustic model
  - Unvoiced segments: fixed rate T0 = 5ms

WORLD: obtain spectral envelope at exact pulse locations, by interpolation



WORLD: obtain spectral envelope at exact pulse locations, by interpolation



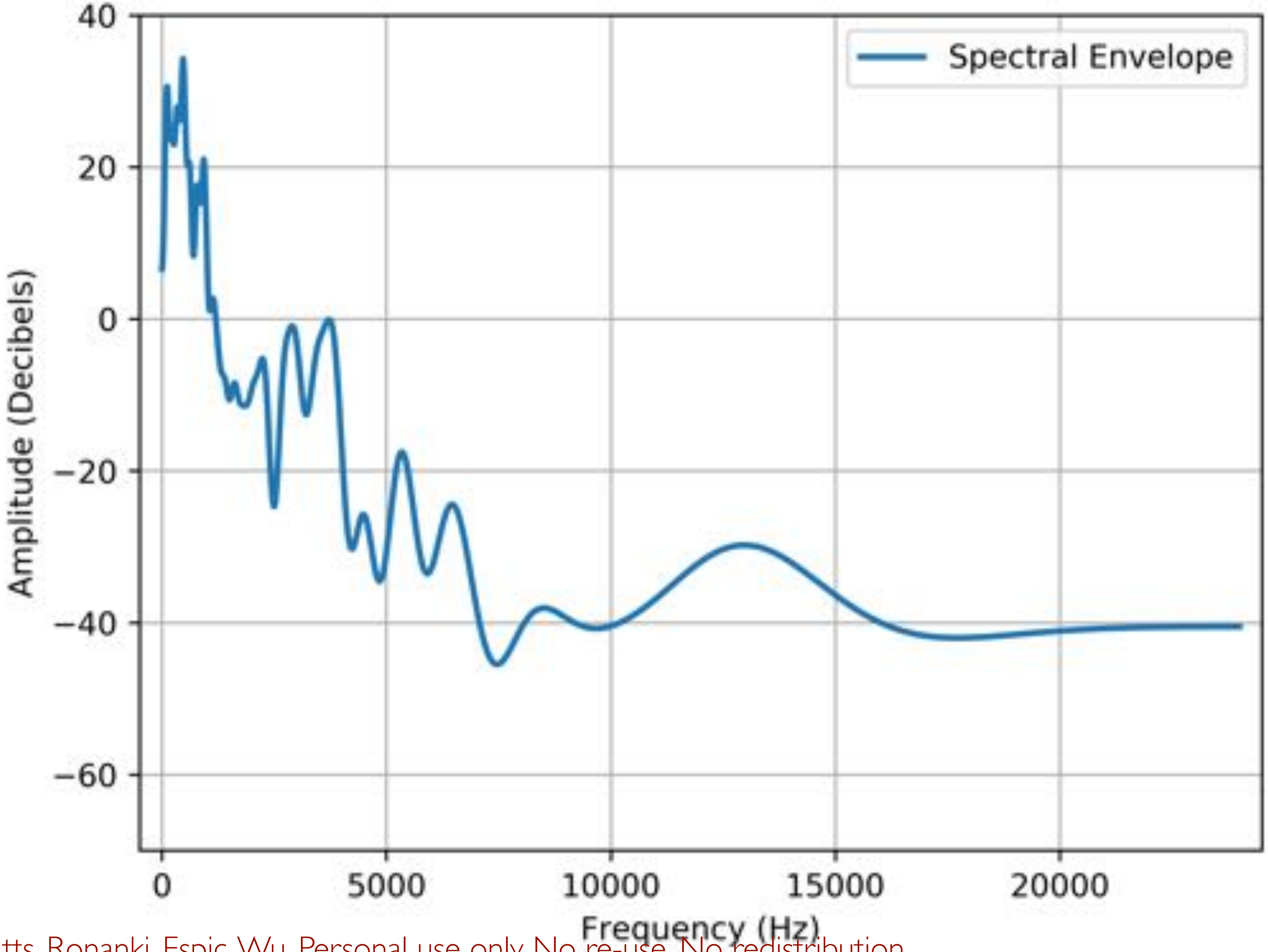
interpolated spectrum at pulse location

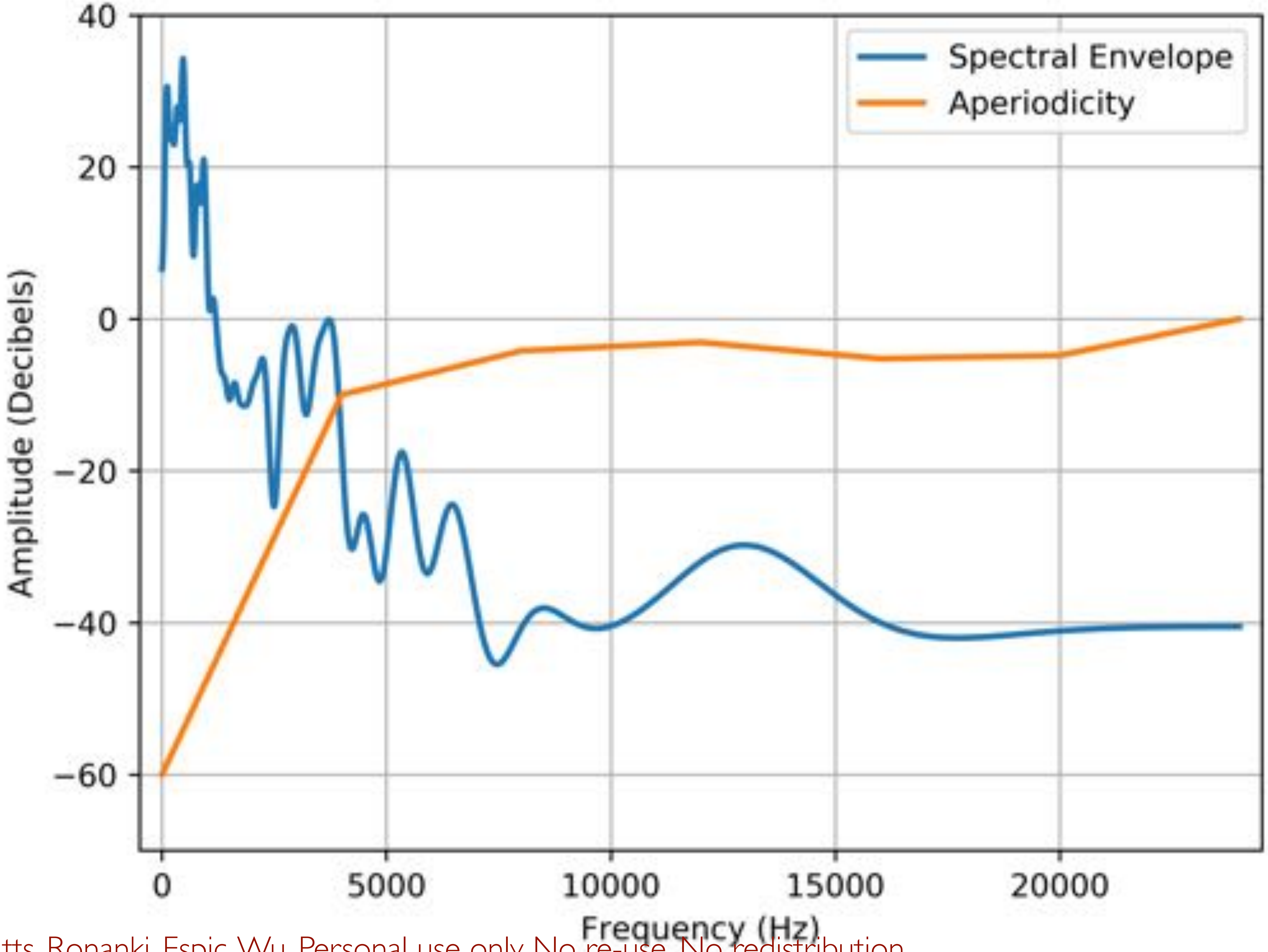
frame n+1

pulse location

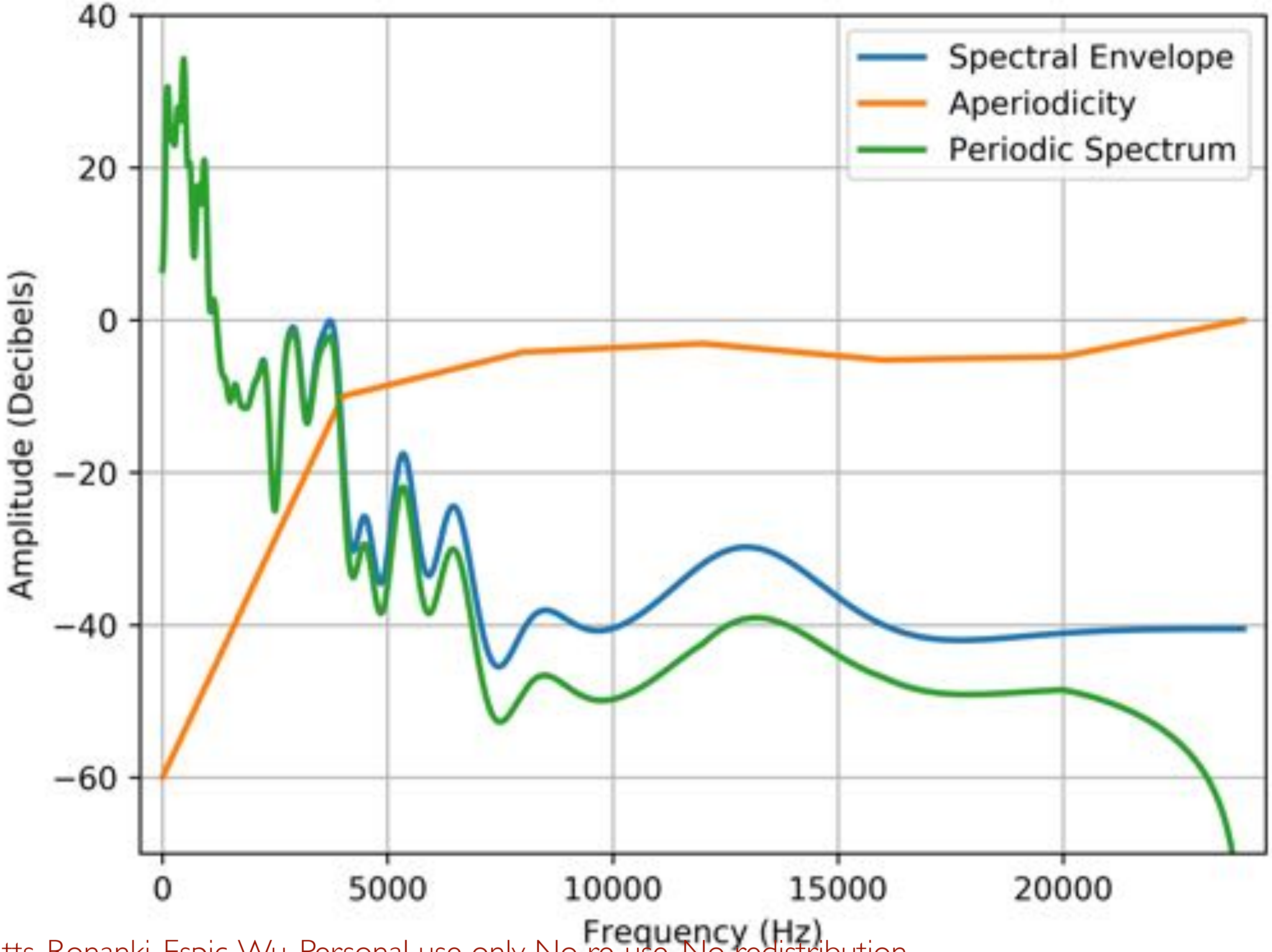
frame n

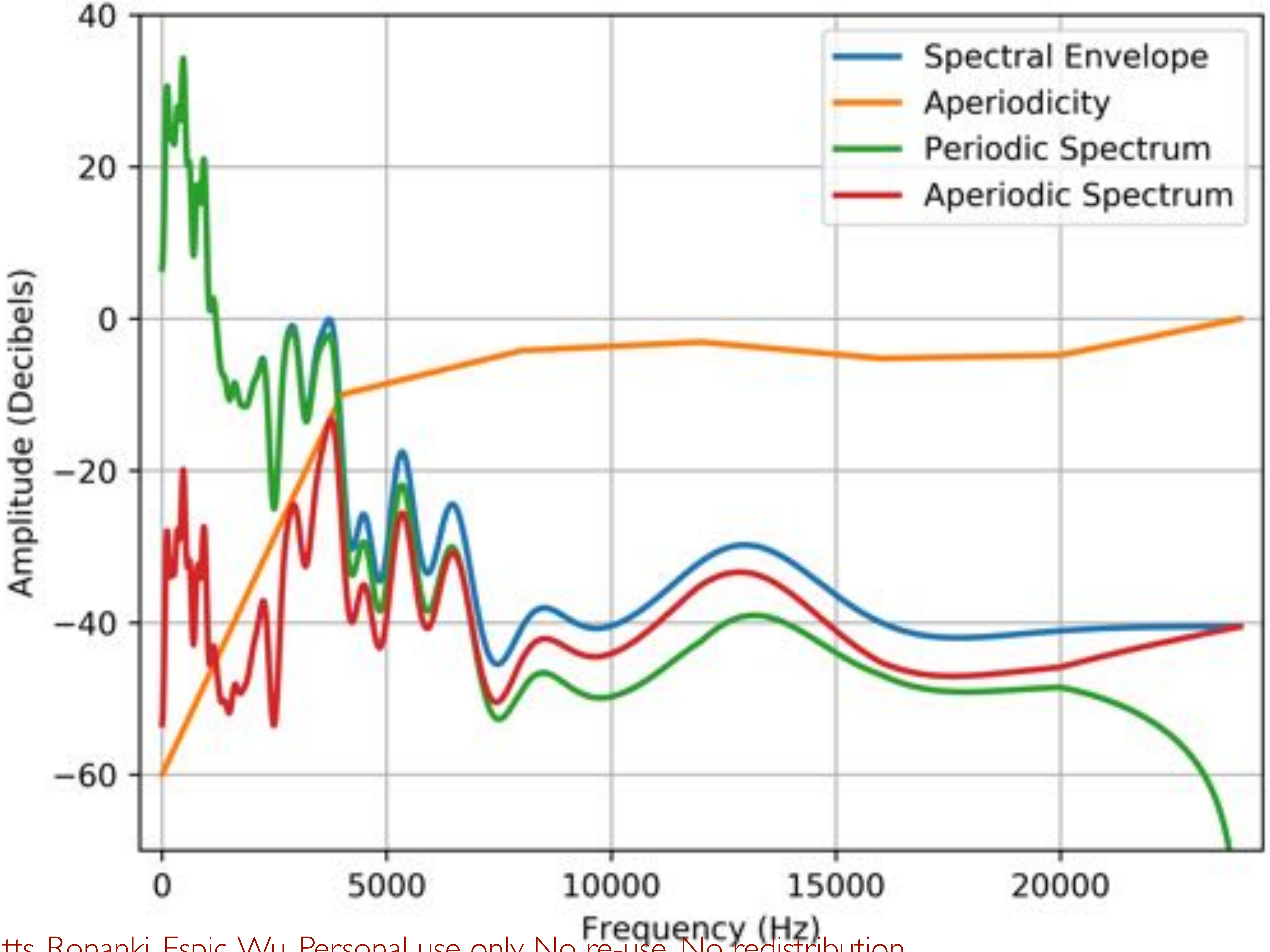
WORLD: reconstruct periodic and aperiodic magnitude spectra


Spectral Envelope
Amplitude (Decibels)
40
20
0
−20
−40
−60
0
5000
10000
15000
20000
Frequency (Hz)



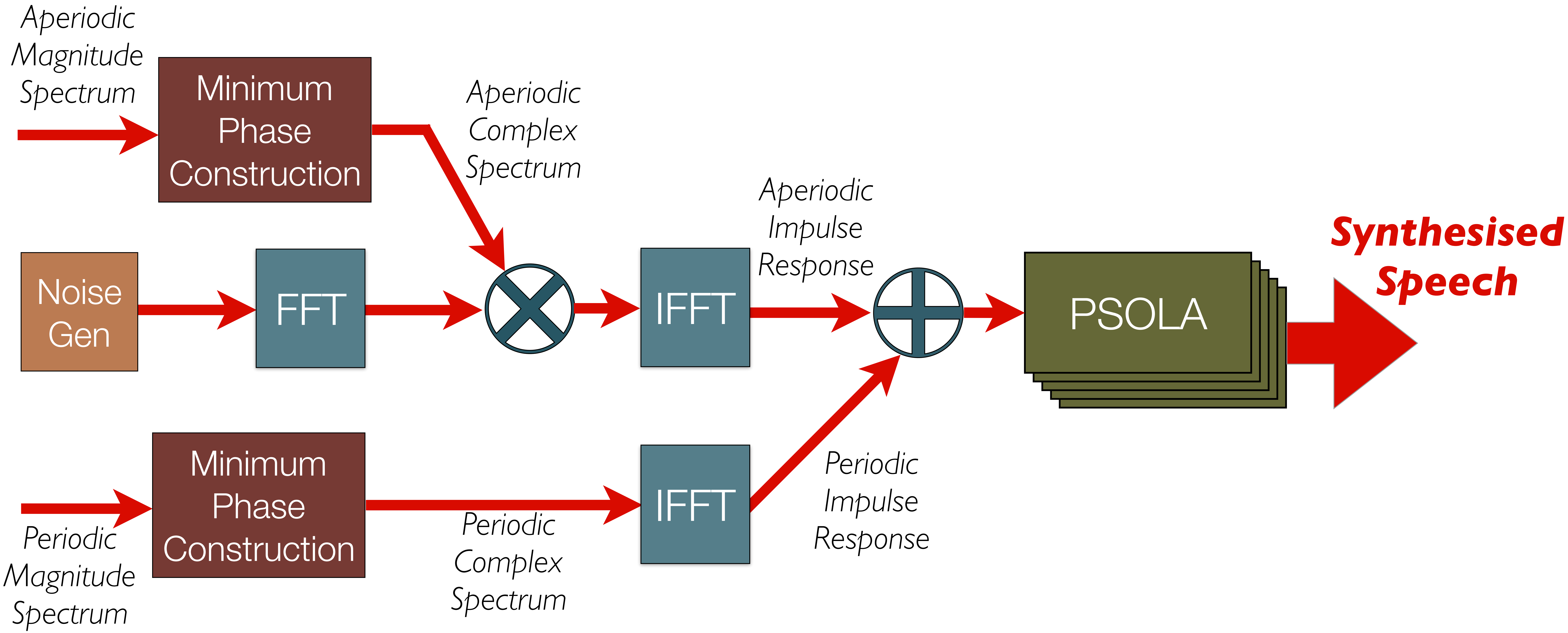
Spectral Envelope
Aperiodicity
Amplitude (Decibels)
Frequency (Hz)
40
20
0
−20
−40
−60
0
5000
10000
15000
20000


Spectral Envelope
Aperiodicity
Periodic Spectrum
40
20
0
−20
−40
−60
Amplitude (Decibels)
0
5000
10000
15000
20000
Frequency (Hz)



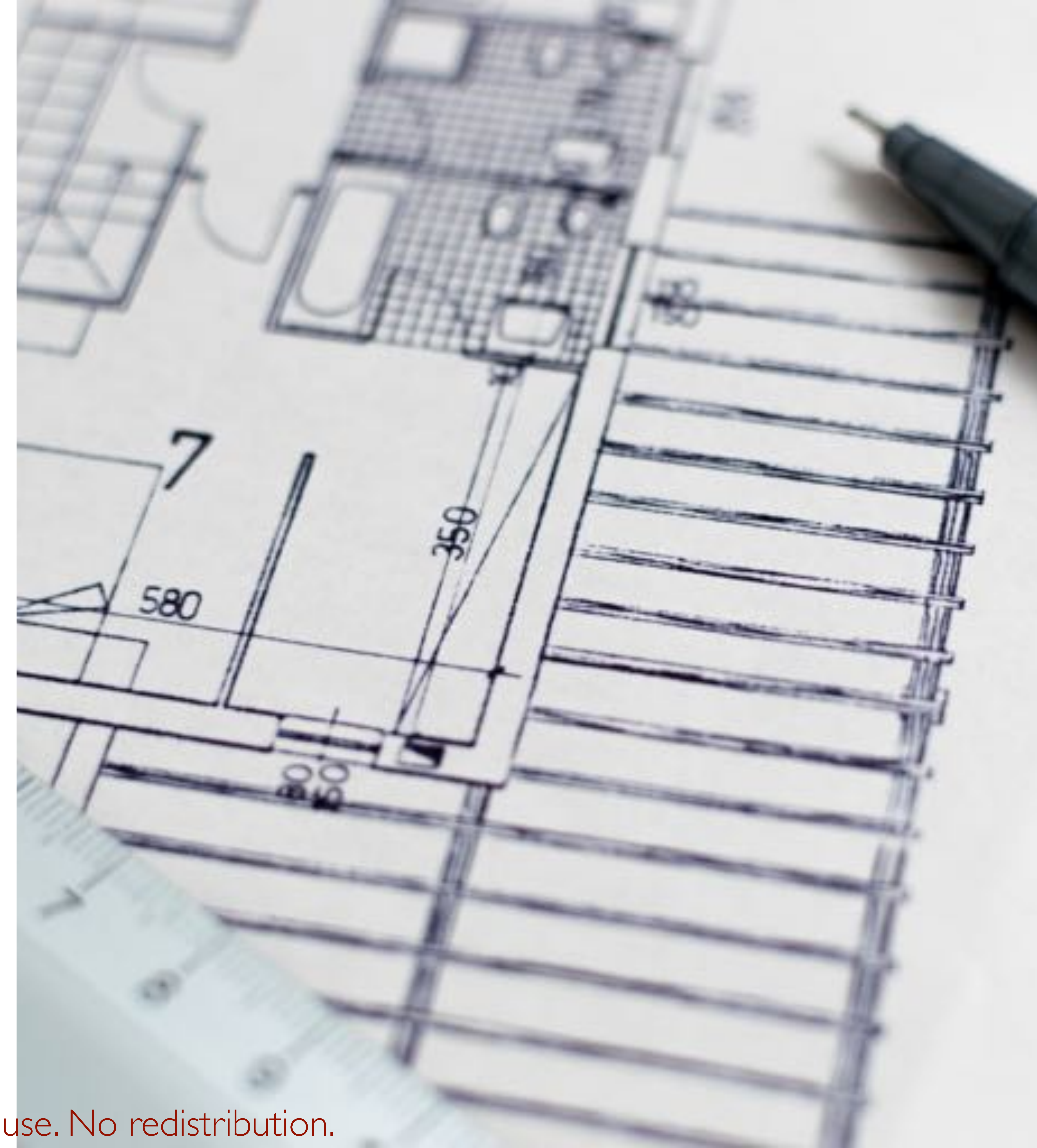
40
20
0
−20
−40
−60
Amplitude (Decibels)
0
5000
10000
15000
20000
Frequency (Hz)
Spectral Envelope
Aperiodicity
Periodic Spectrum
Aperiodic Spectrum

WORLD: generate waveform


Aperiodic Magnitude Spectrum
Minimum Phase Construction
Aperiodic Complex Spectrum
Noise Gen
FFT
IFFT
Aperiodic Impulse Response
PSOLA
Synthesised Speech
Minimum Phase Construction
Periodic Magnitude Spectrum
Periodic Complex Spectrum
IFFT
Periodic Impulse Response

Design choices: waveform generation

- fixed framerate *or* pitch synchronous
  - may be different from what you used in acoustic feature extraction
- cepstrum *or* spectrum
- source
  - pulse/noise *or* mixed *or* sampled
- phase
  - synthetic (e.g., pulse train + minimum phase filter) *or*
  - predict using acoustic model



Agenda

	Topic	Presenter
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	**Simon King**
PART 3	Extensions	Zhizheng Wu

Recap & conclusion

Simon King

Agenda

	Topic	**Presenter**
PART 1	From text to speech	Simon King
PART 2	The front end	Oliver Watts
	Linguistic feature extraction & engineering	Srikanth Ronanki
	Acoustic feature extraction & engineering	Felipe Espic
	Regression	Zhizheng Wu
	Waveform generation	Felipe Espic
	Recap and conclusion	Simon King
PART 3	**Extensions**	**Zhizheng Wu**

Extensions

Zhizheng Wu

Extensions

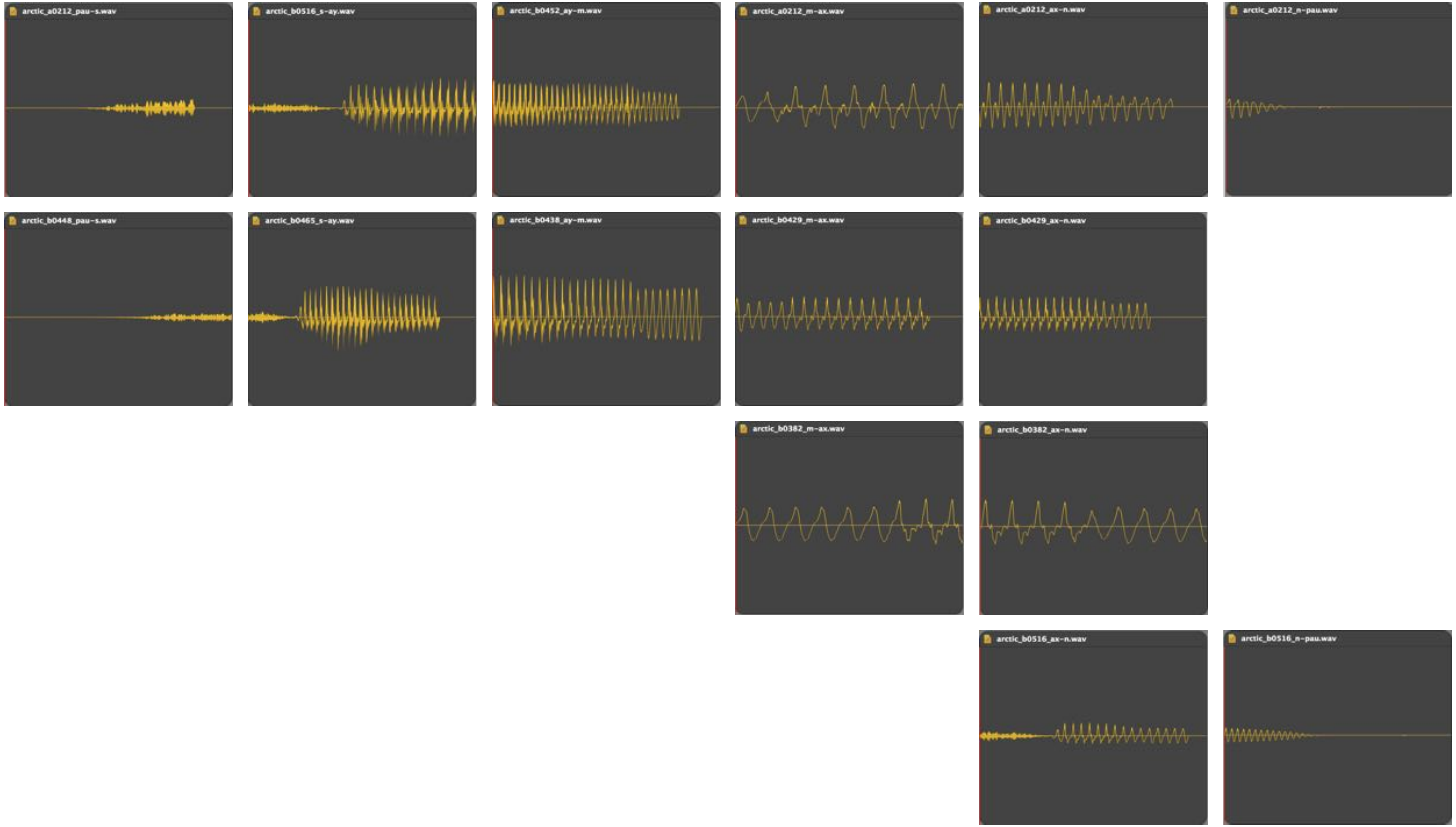
- Hybrid speech synthesis
  - make acoustic feature predictions with Merlin, then select units with Festival
- Voice conversion
  - input speech, instead of text
  - training data is aligned input and output speech (instead of phone labels and speech)
- Speaker adaptation
  - augmenting the input
  - adapting hidden layers
  - transforming the output

Extensions

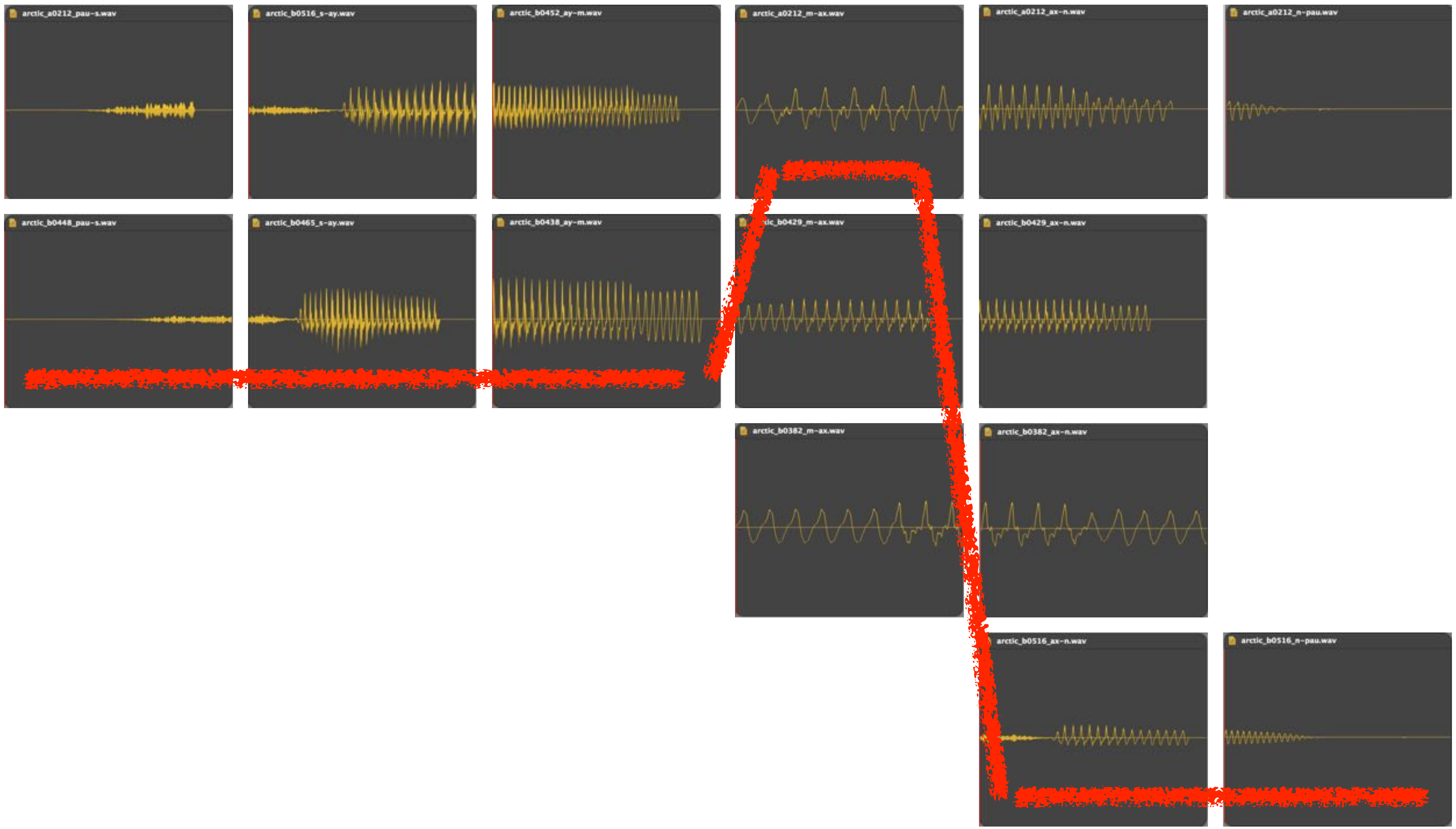
- Hybrid speech synthesis
  - make acoustic feature predictions with Merlin, then select units with Festival
- Voice conversion
  - input speech, instead of text
  - training data is aligned input and output speech (instead of phone labels and speech)
- Speaker adaptation
  - augmenting the input
  - adapting hidden layers
  - transforming the output

“Simon”

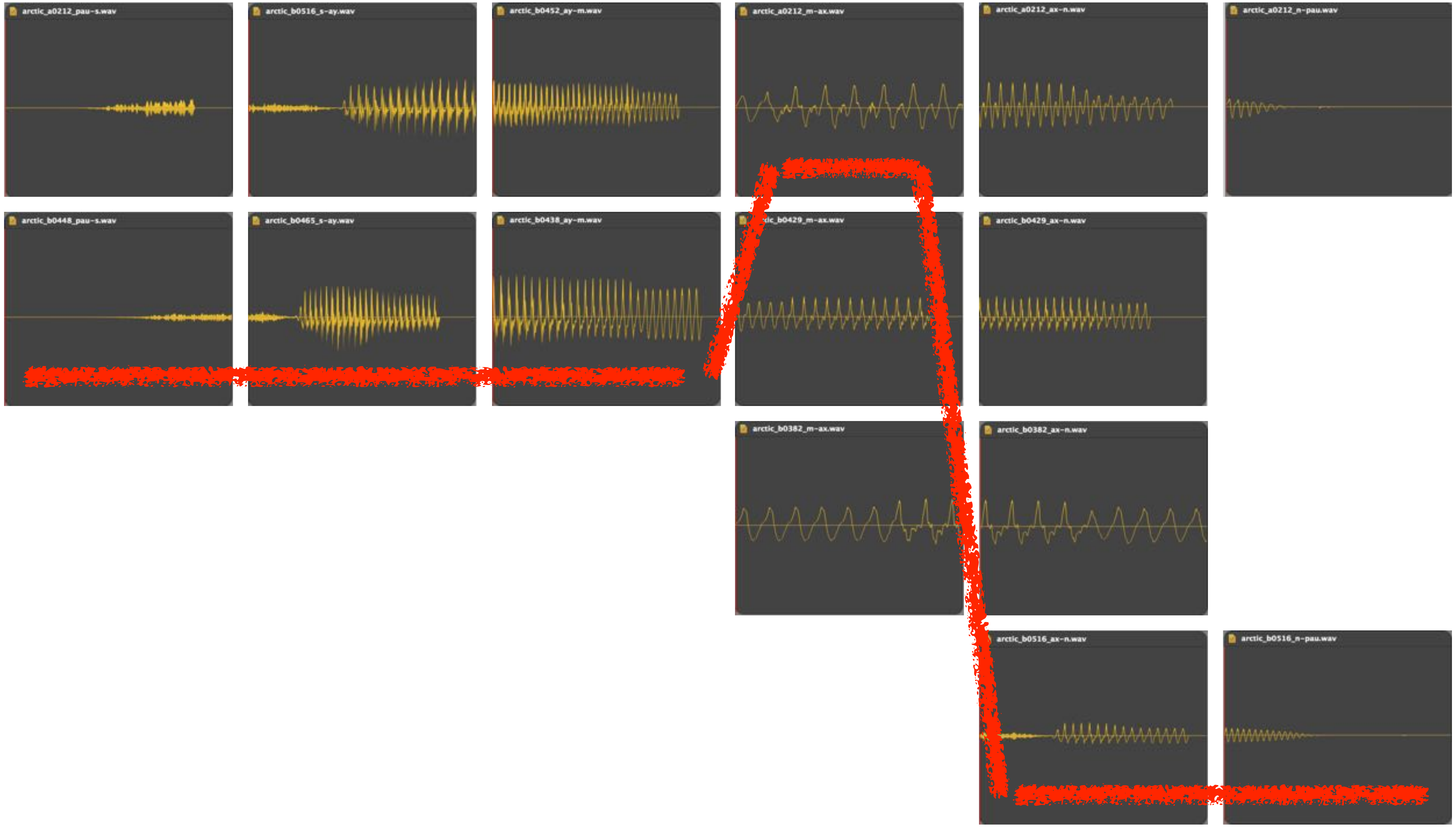
“Simon”


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

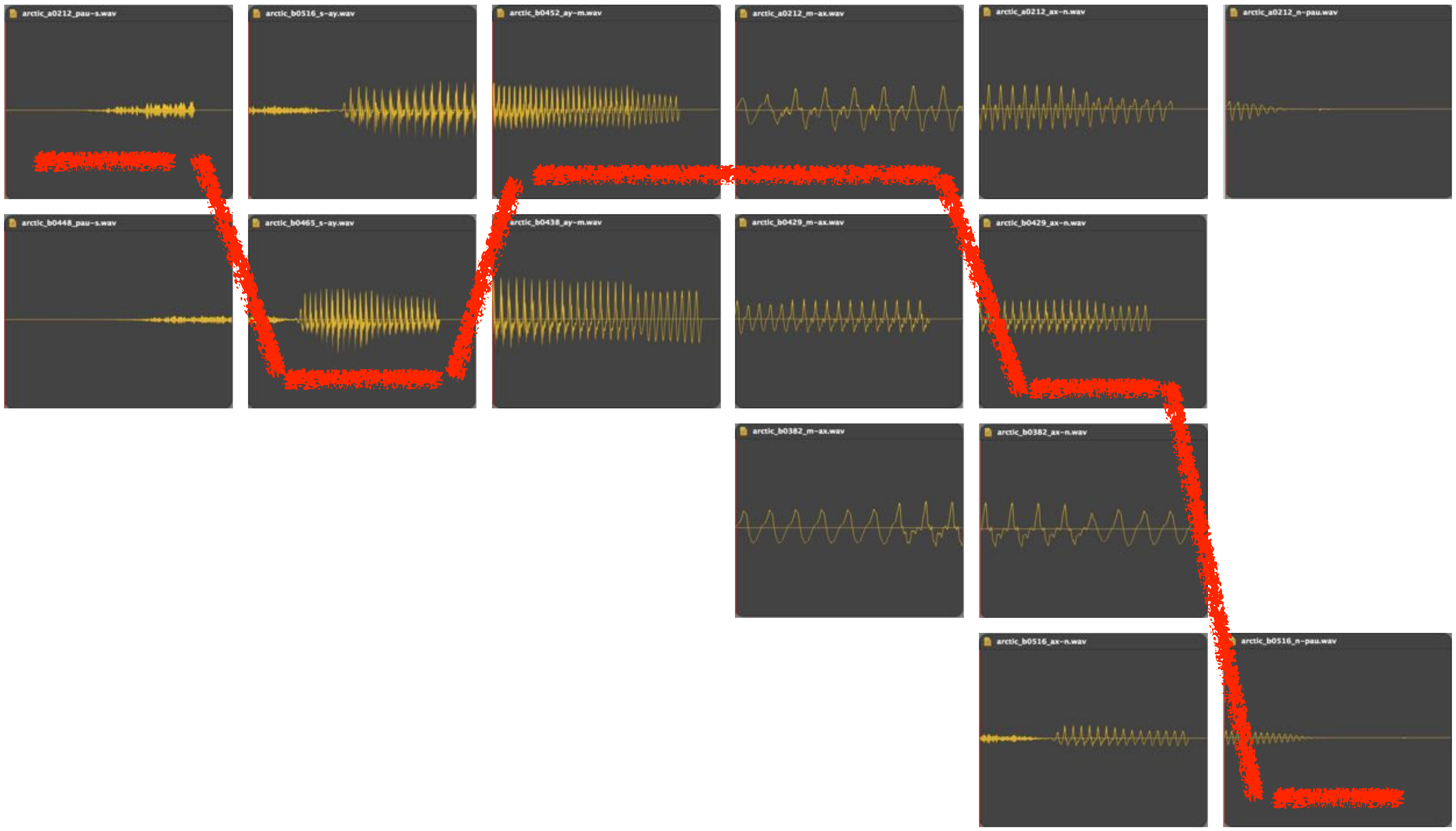
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
ic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

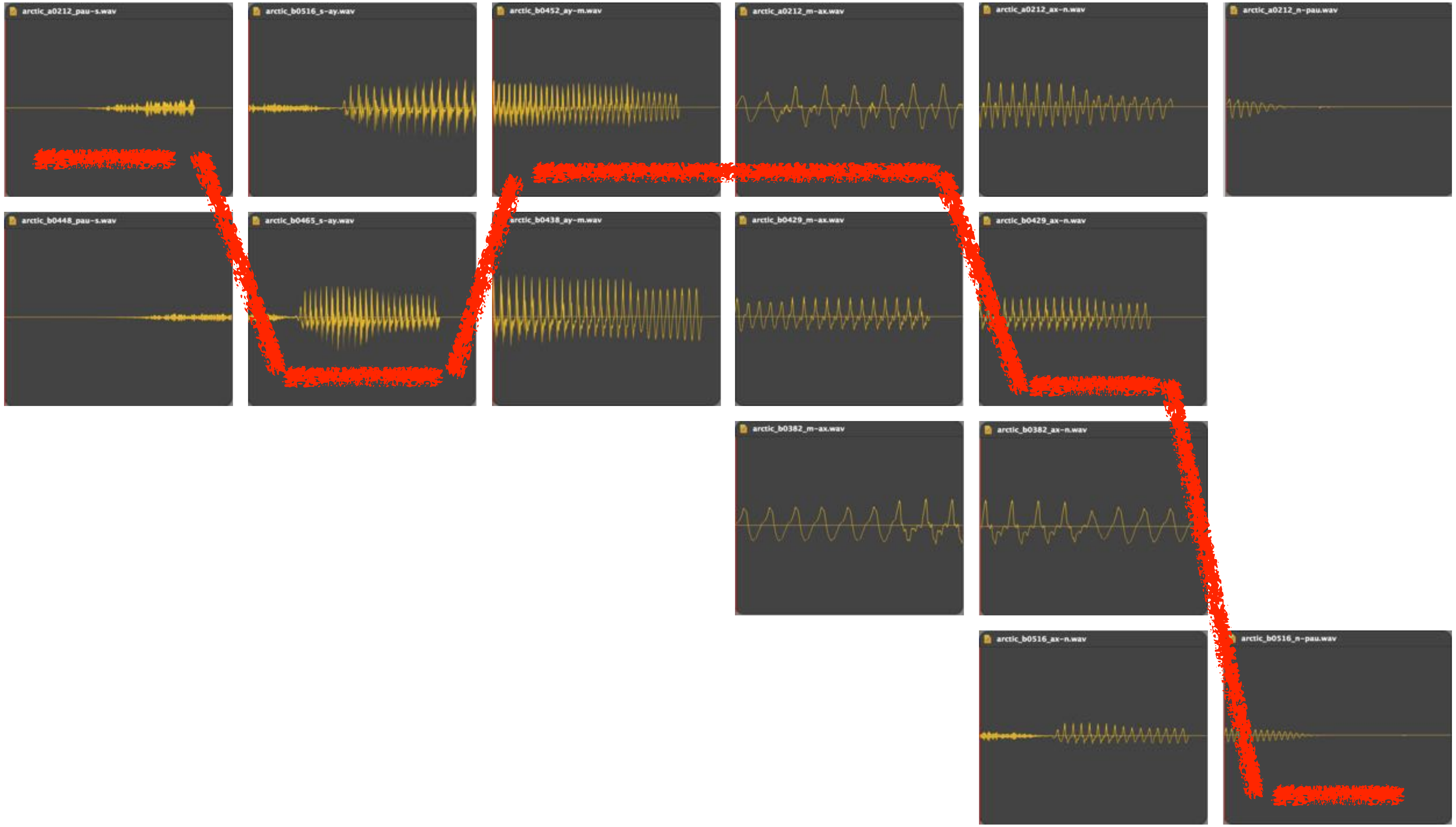
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
tic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

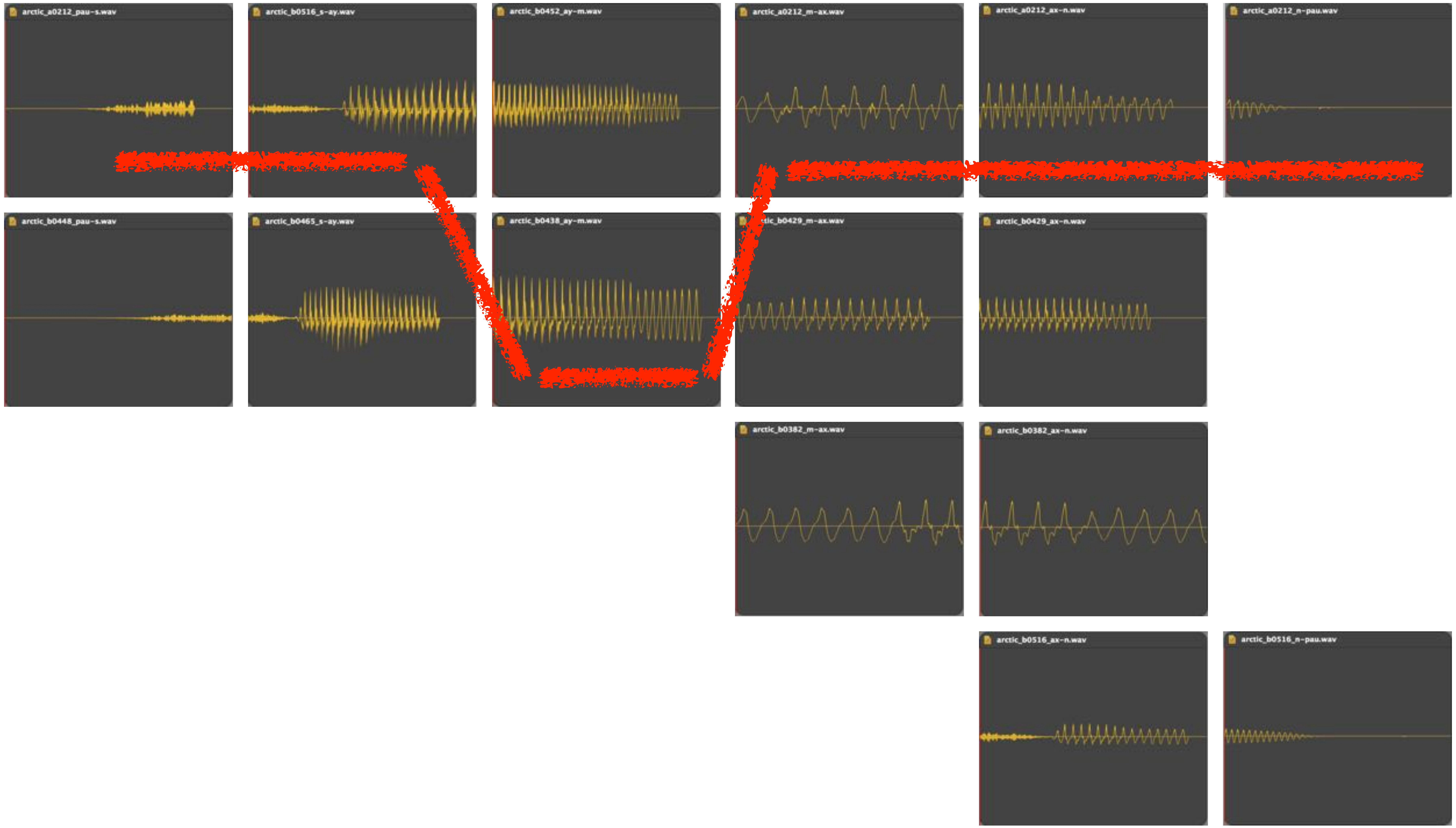
“Simon”


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

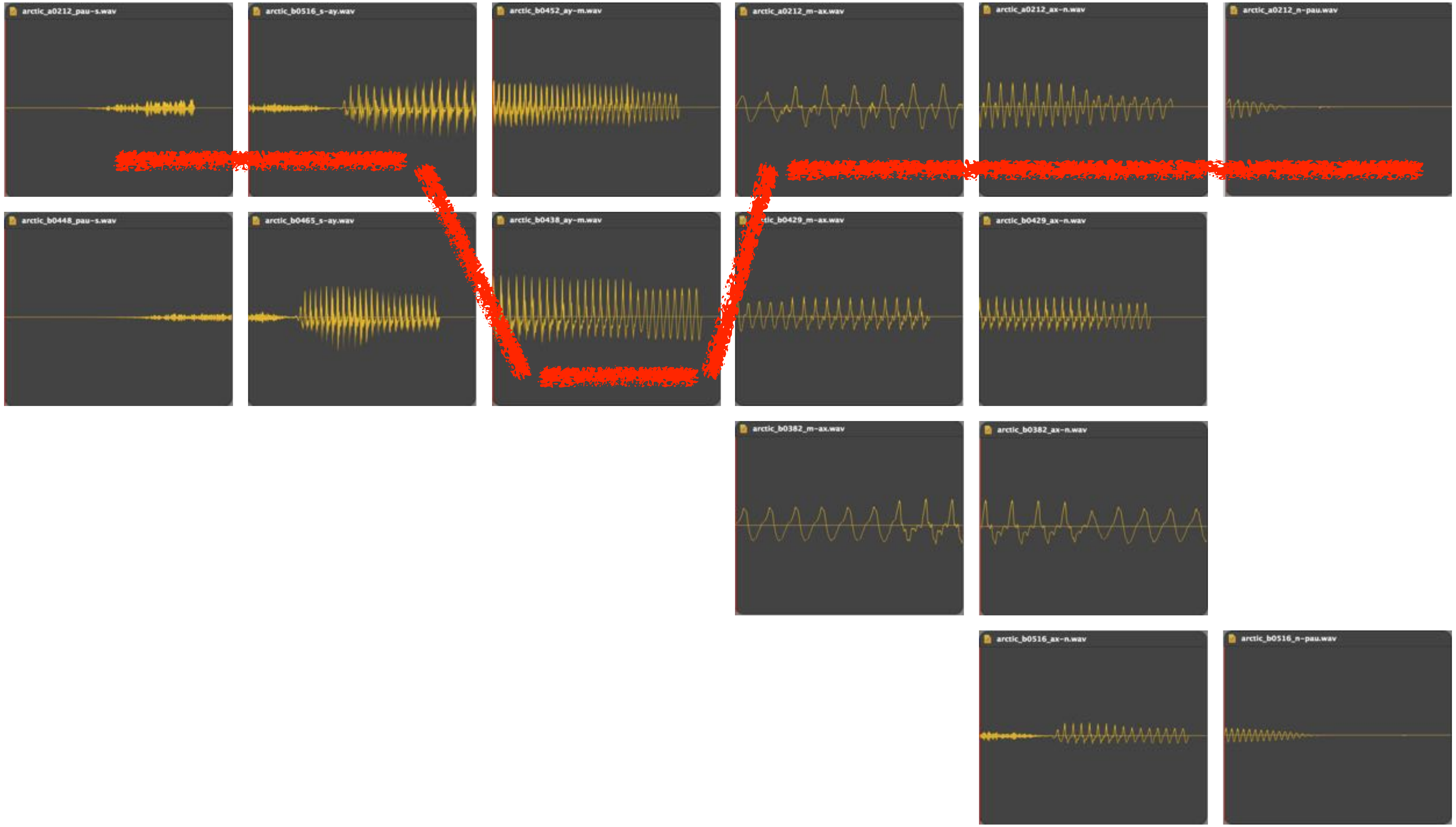
“Simon”


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

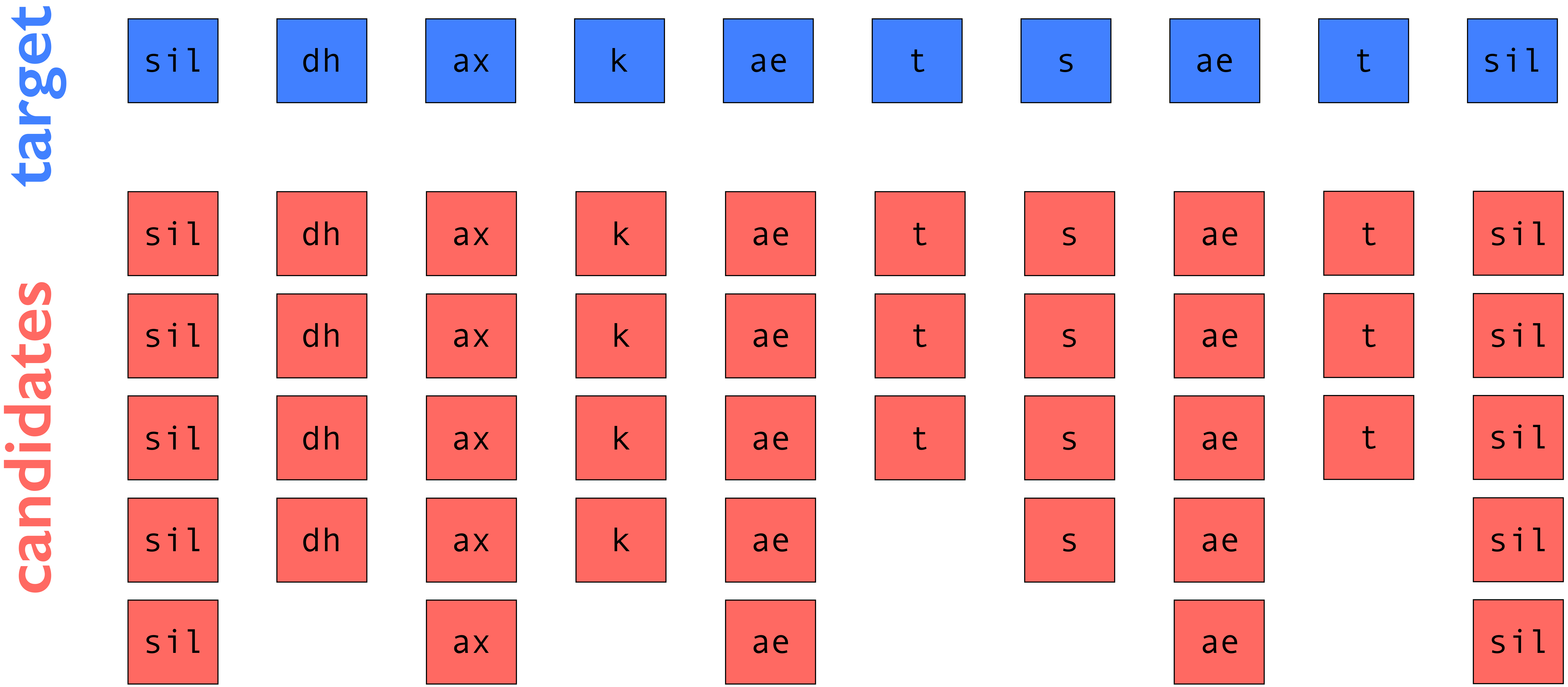
“Simon”


arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
tic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

“Simon”

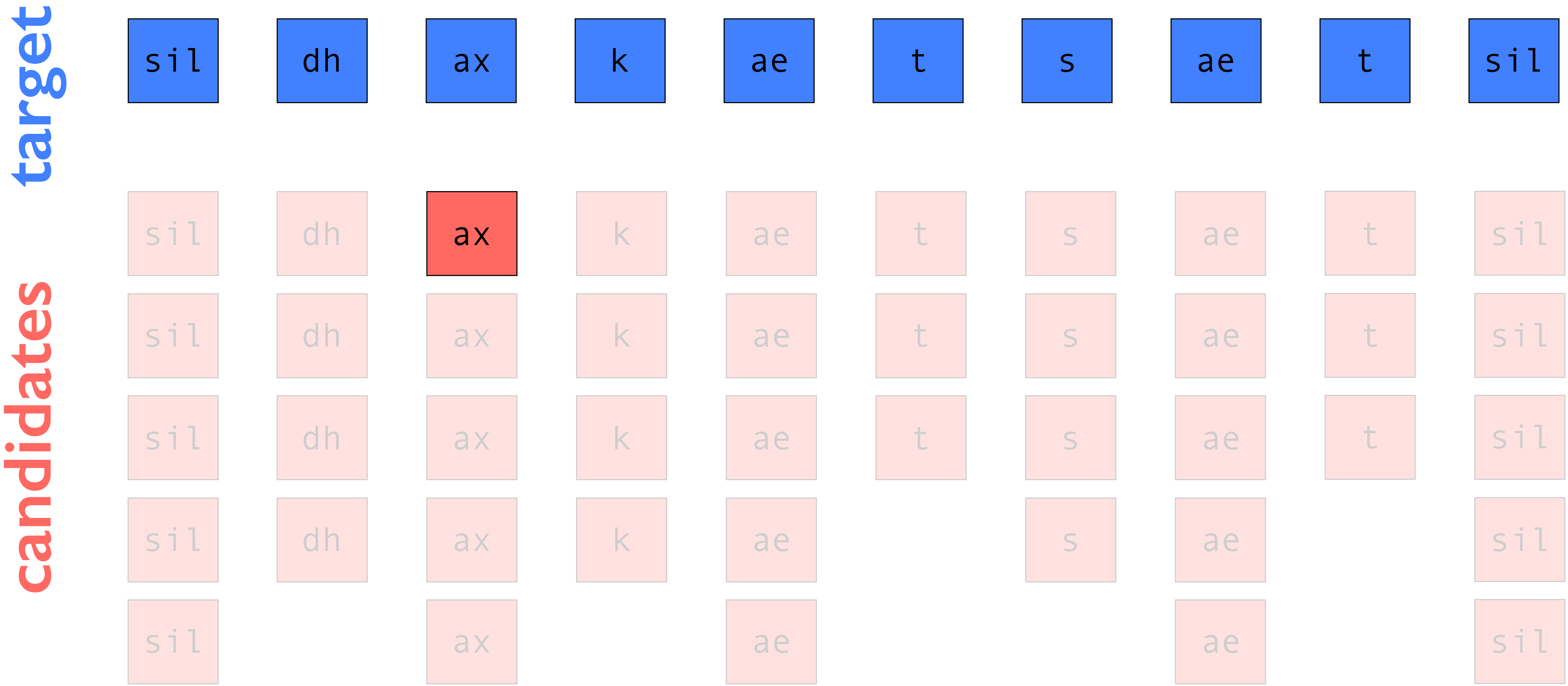

arctic_a0212_pau-s.wav
arctic_b0516_s-ay.wav
arctic_b0452_ay-m.wav
arctic_a0212_m-ax.wav
arctic_a0212_ax-n.wav
arctic_a0212_n-pau.wav
arctic_b0448_pau-s.wav
arctic_b0465_s-ay.wav
arctic_b0438_ay-m.wav
tic_b0429_m-ax.wav
arctic_b0429_ax-n.wav
arctic_b0382_m-ax.wav
arctic_b0382_ax-n.wav
arctic_b0516_ax-n.wav
arctic_b0516_n-pau.wav

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



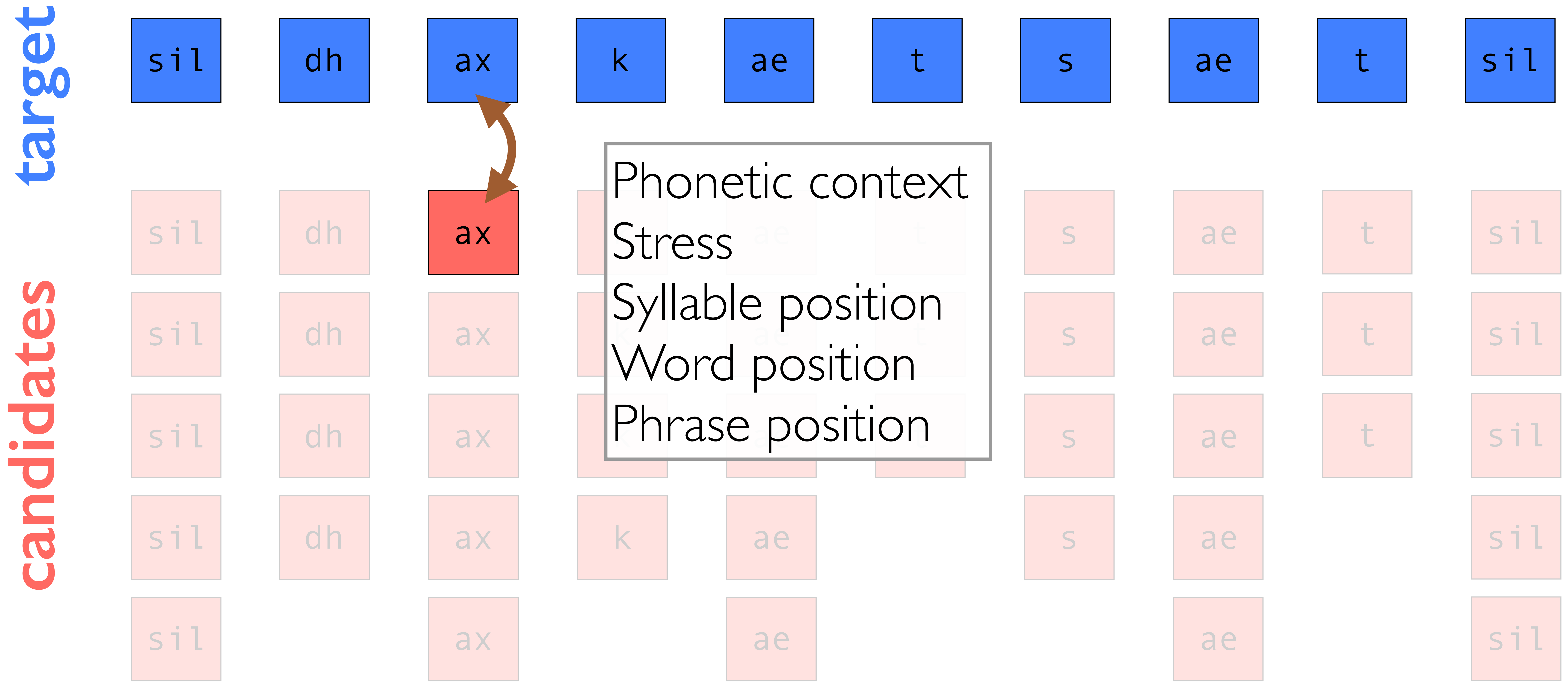
target	sil	dh	ax	k	ae	t	s	ae	t	sil
candidates	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae		s	ae		sil
	sil		ax		ae			ae		sil

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



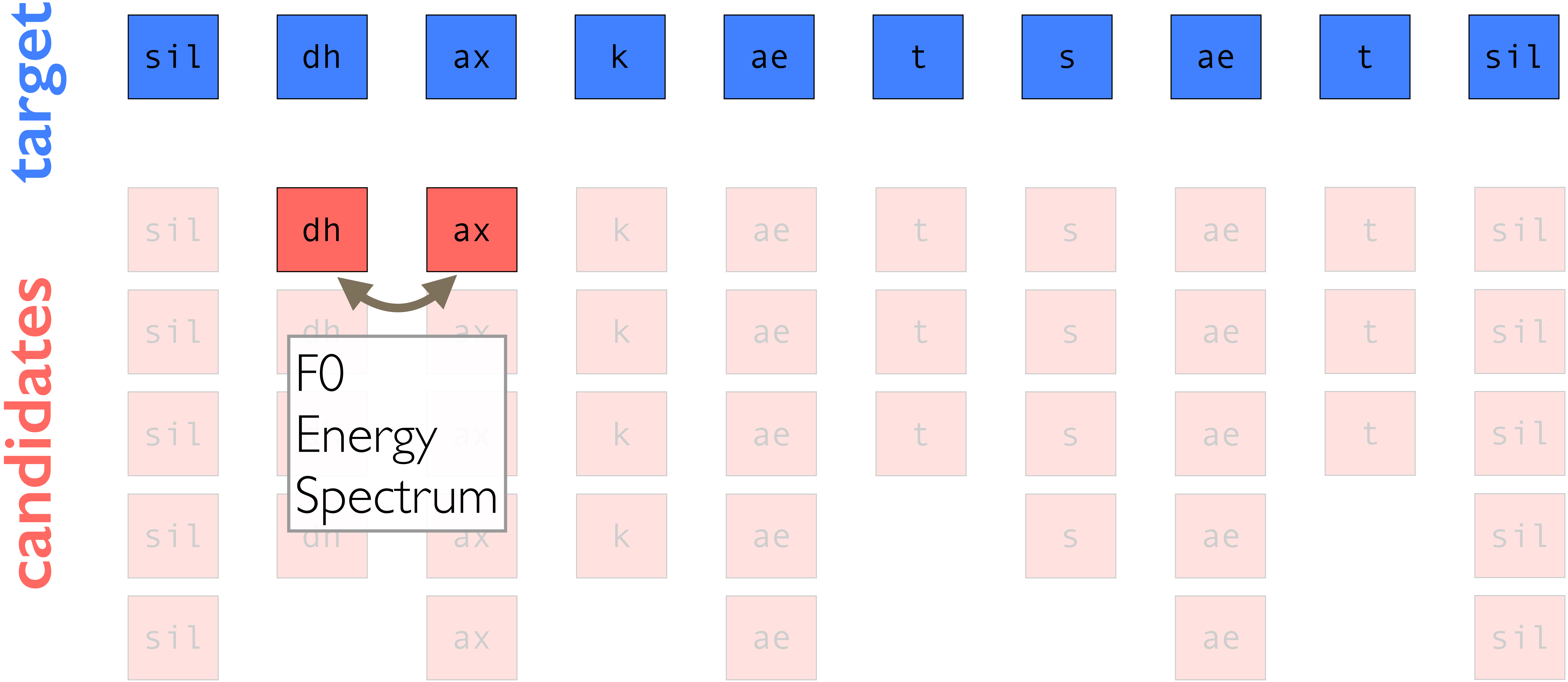
target	sil	dh	ax	k	ae	t	s	ae	t	sil
candidates	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae		s	ae		sil
	sil		ax		ae			ae		sil

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



Phonetic context
Stress
Syllable position
Word position
Phrase position

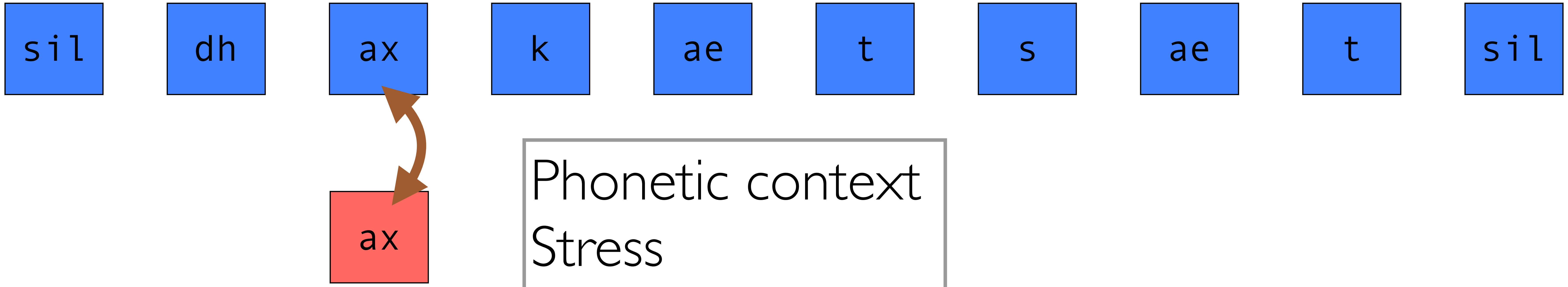
Classical unit selection (drawn here with phone units) - target and join costs



target	sil	dh	ax	k	ae	t	s	ae	t	sil
candidates	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae	t	s	ae	t	sil
	sil		ax	k	ae	t	s	ae	t	sil
	sil	dh	ax	k	ae		s	ae		sil
	sil		ax		ae			ae		sil

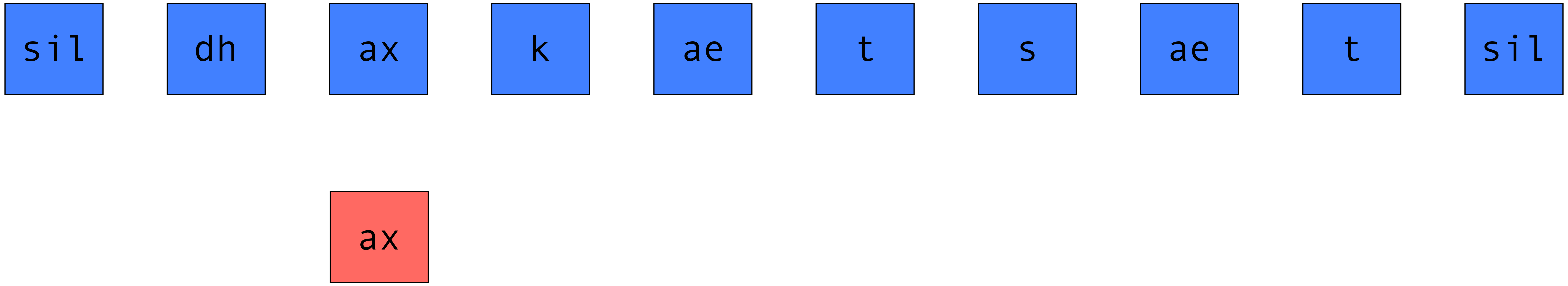
F0
Energy
Spectrum

Independent Feature Formulation (IFF) target cost



Phonetic context
Stress
Syllable position
Word position
Phrase position

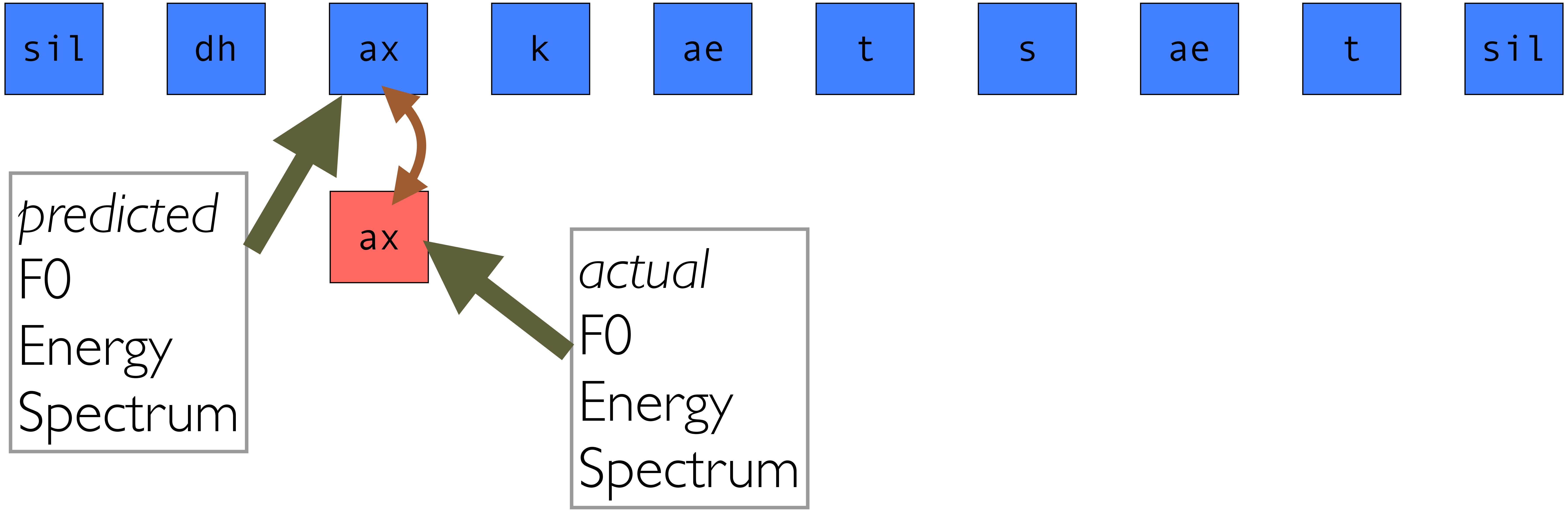
Acoustic Space Formulation (ASF) target cost



sil dh ax k ae t s ae t sil

ax

Acoustic Space Formulation (ASF) target cost

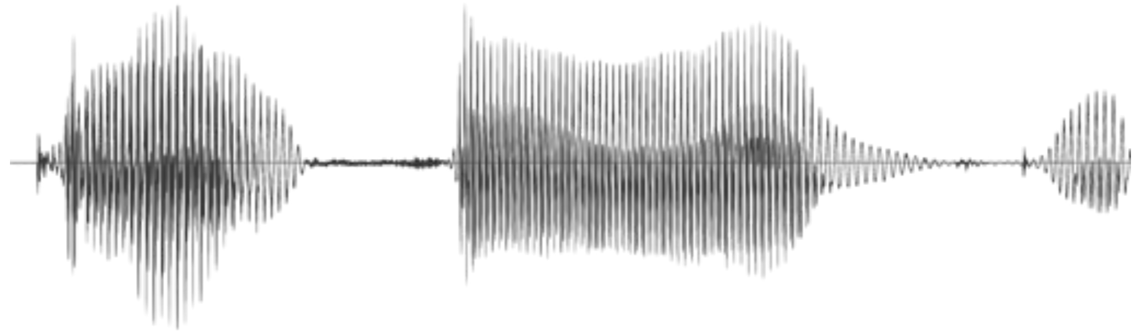


predicted
F0
Energy
Spectrum

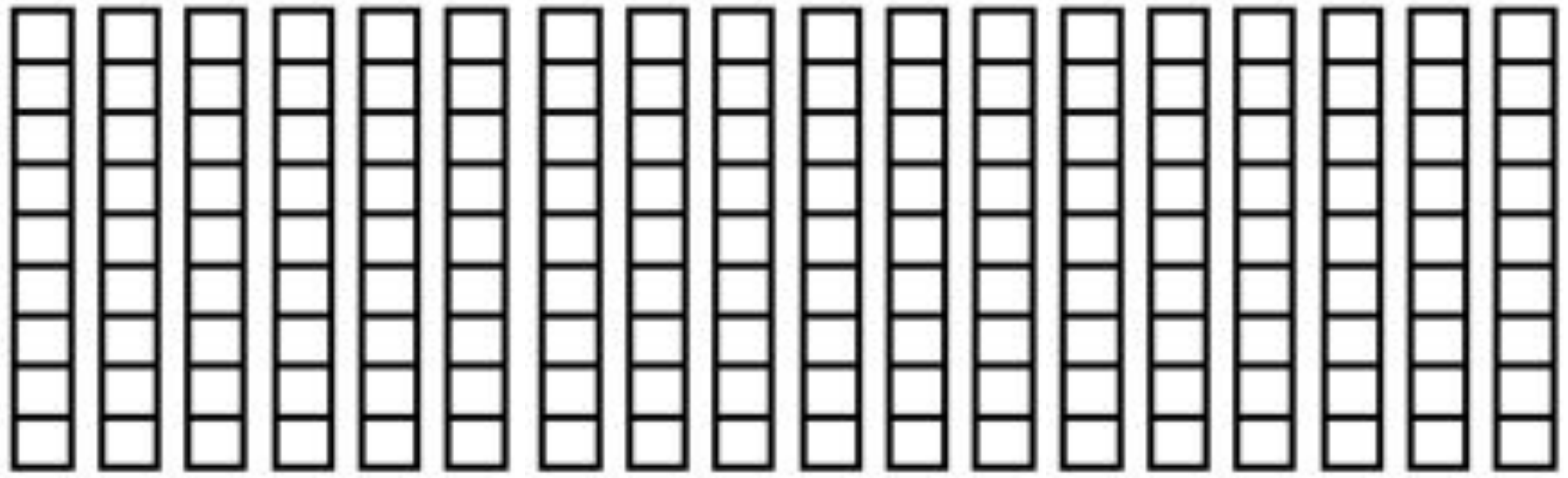
actual
F0
Energy
Spectrum

Hybrid speech synthesis is like
unit selection with an Acoustic Space Formulation target cost

waveform



acoustic features

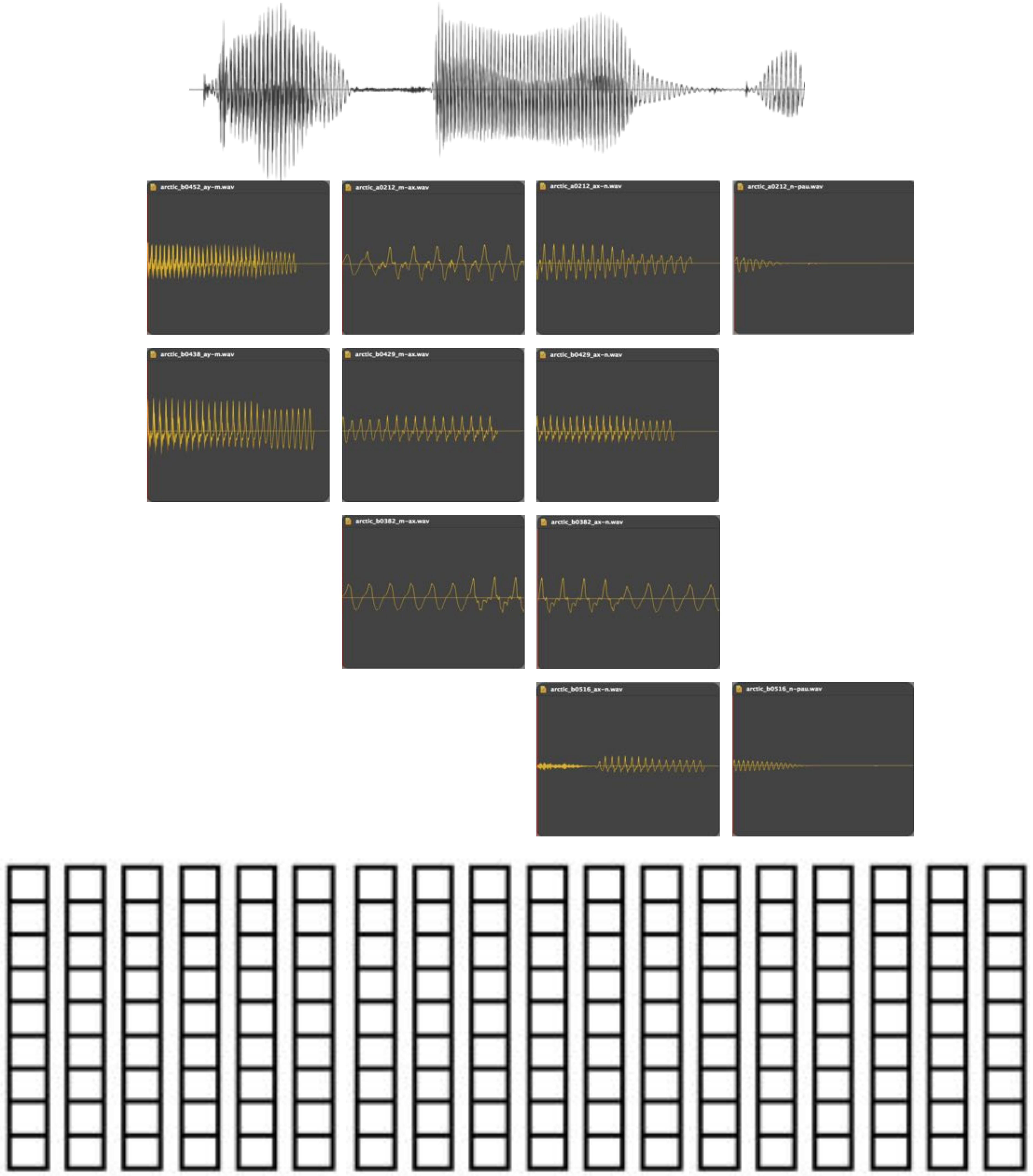


Hybrid speech synthesis is like
unit selection with an Acoustic Space Formulation target cost

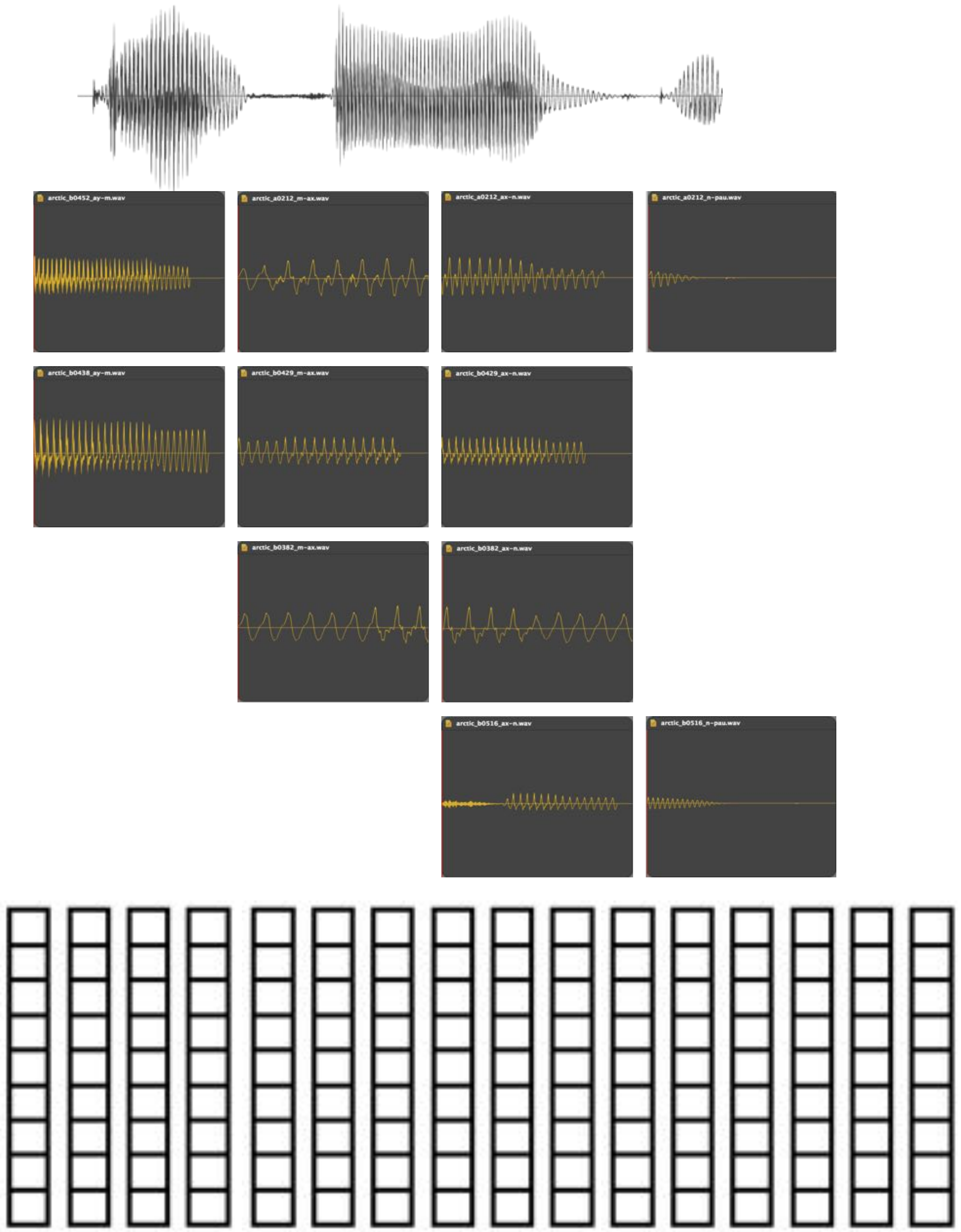
waveform

speech database

acoustic features



Hybrid speech synthesis is like
unit selection with an Acoustic Space Formulation target cost



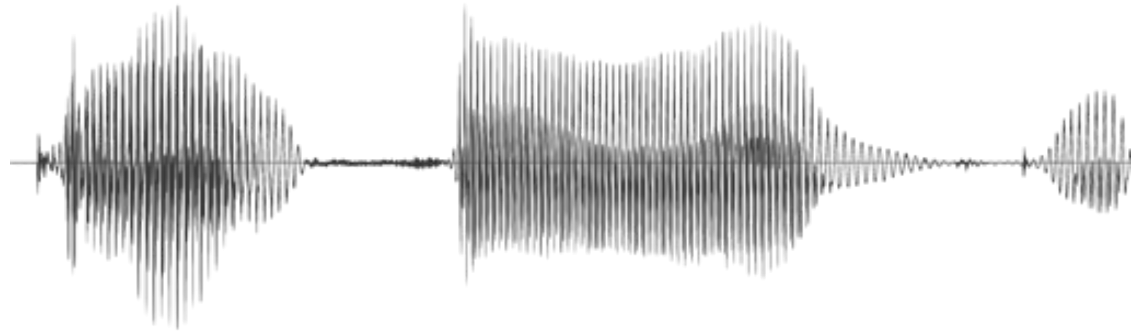
waveform

speech
database

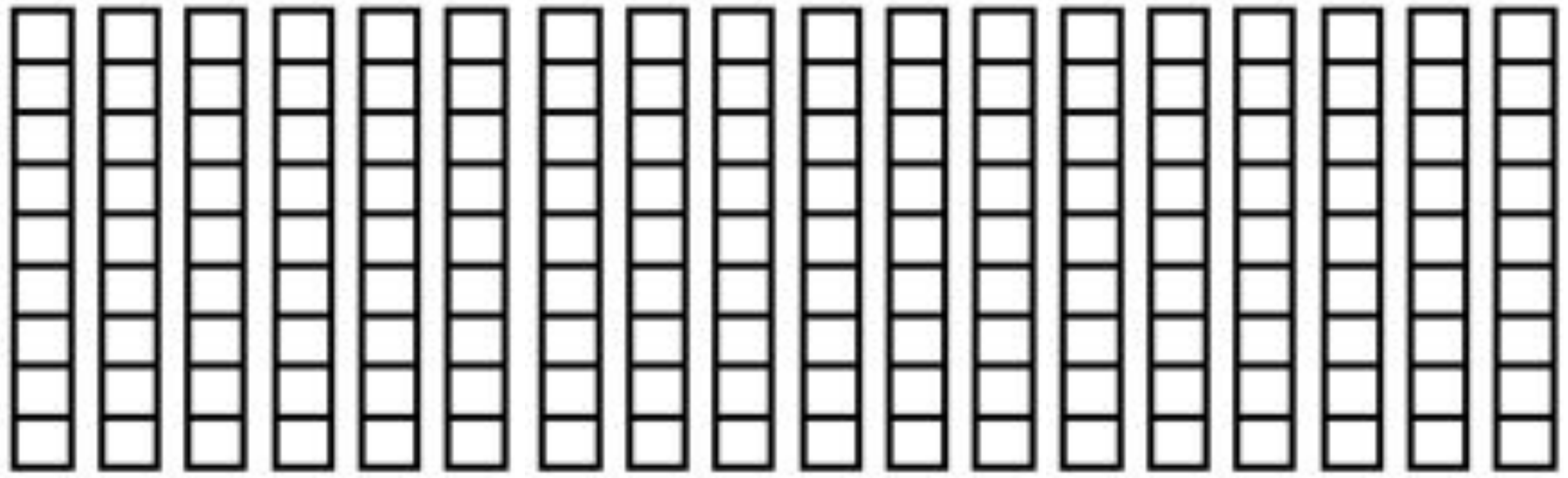
“partial
synthesis”

Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder

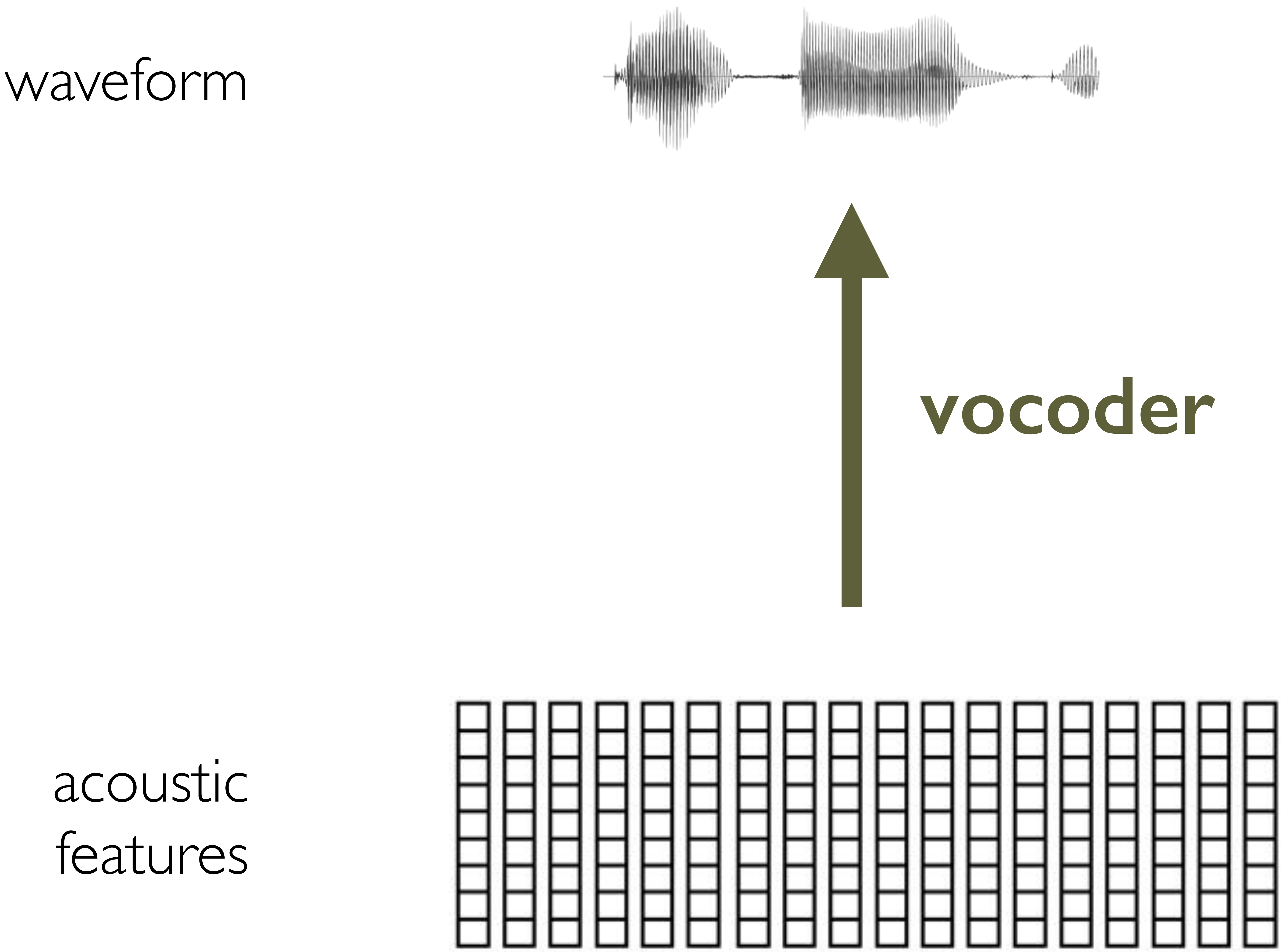
waveform



acoustic features



Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder

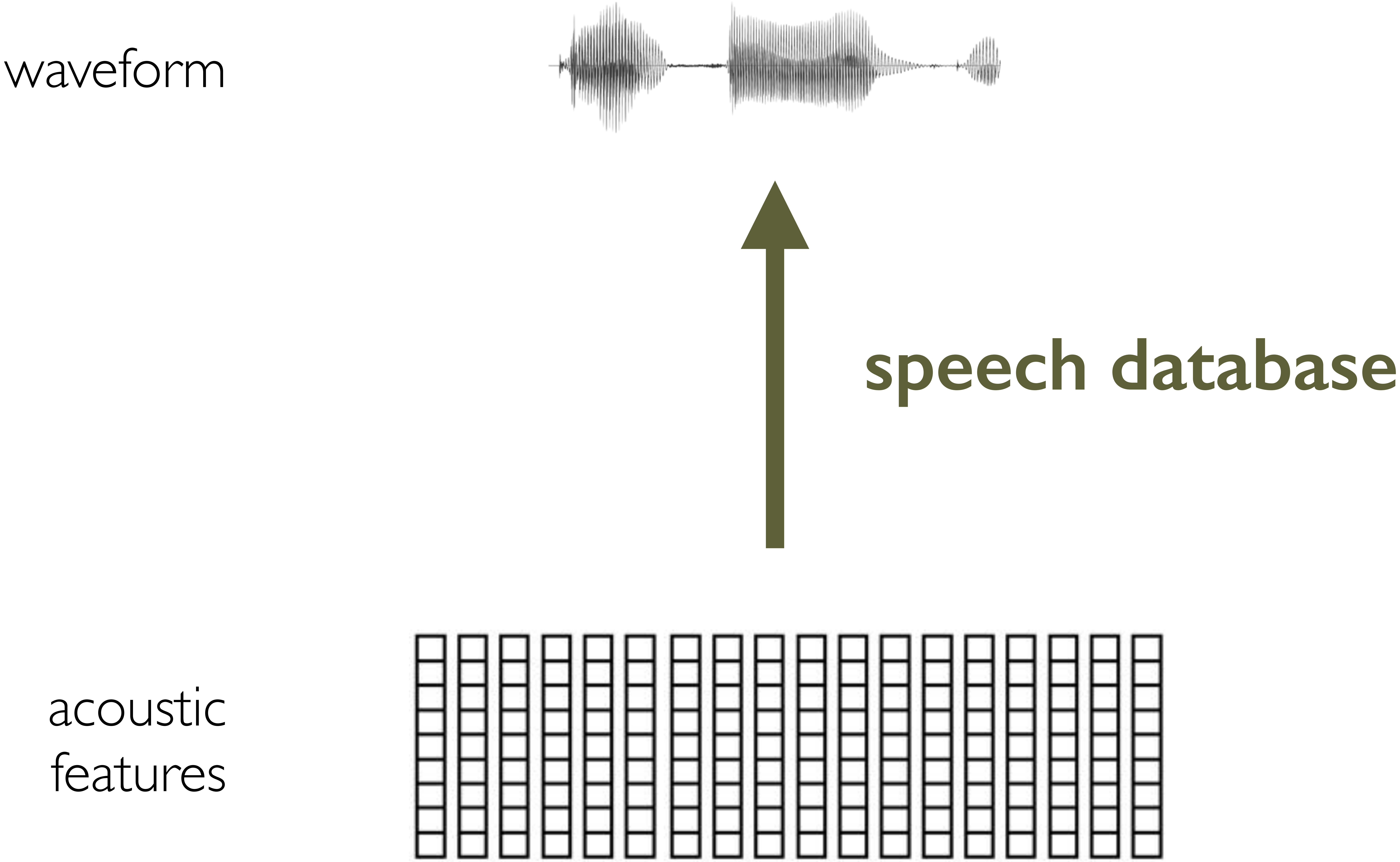


waveform

vocoder

acoustic features

Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder



waveform

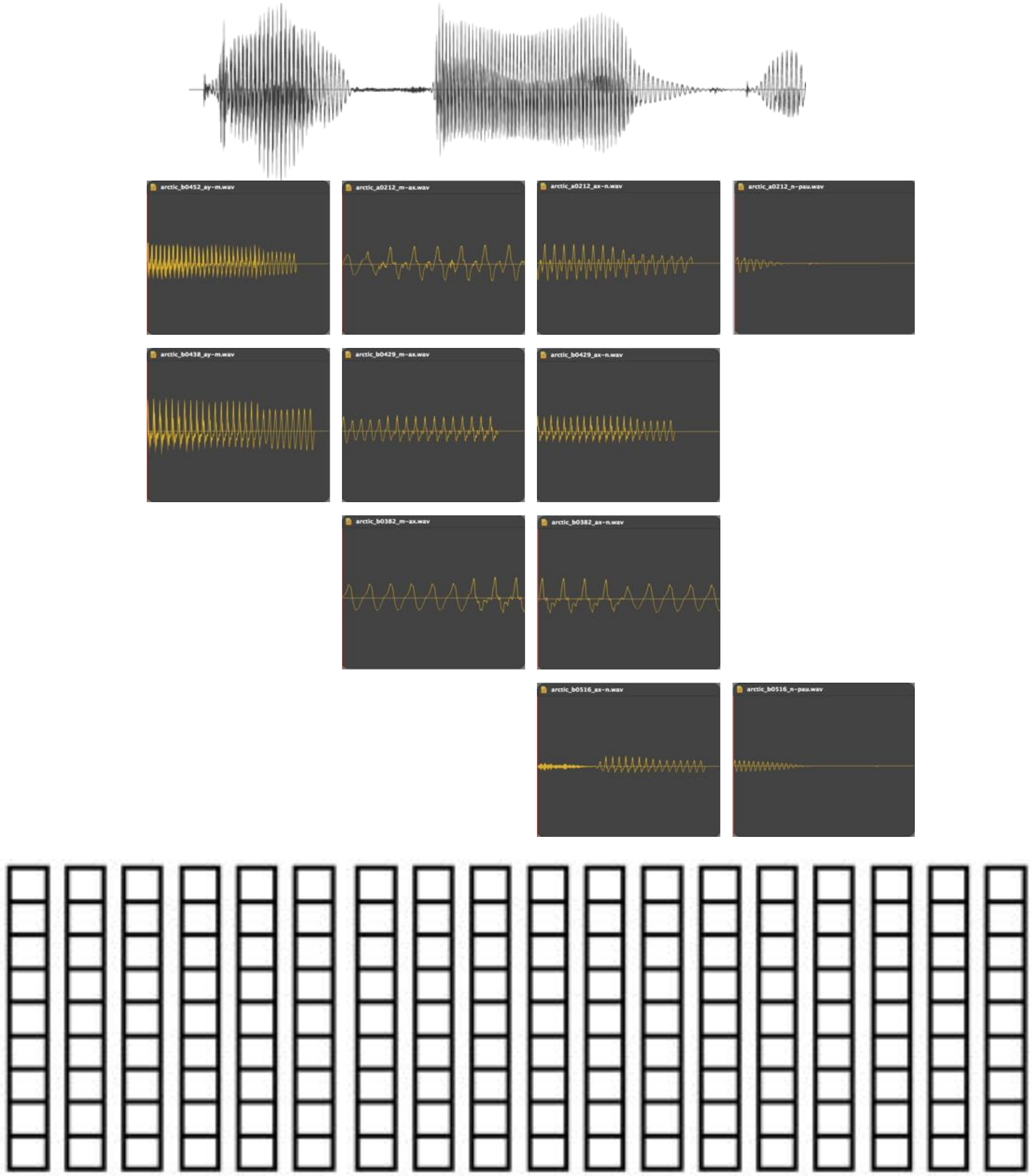
speech database

acoustic features

Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder

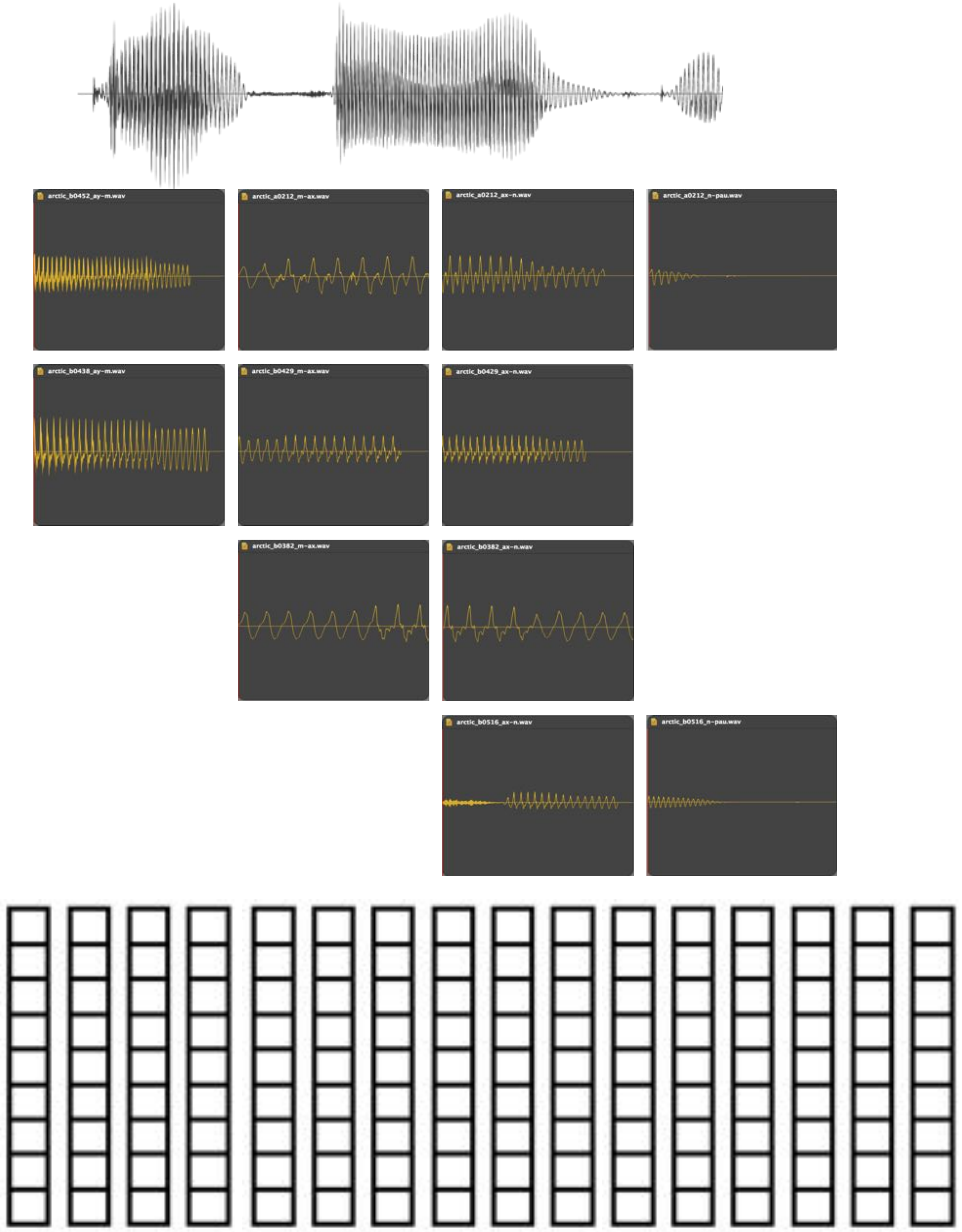
waveform

acoustic
features



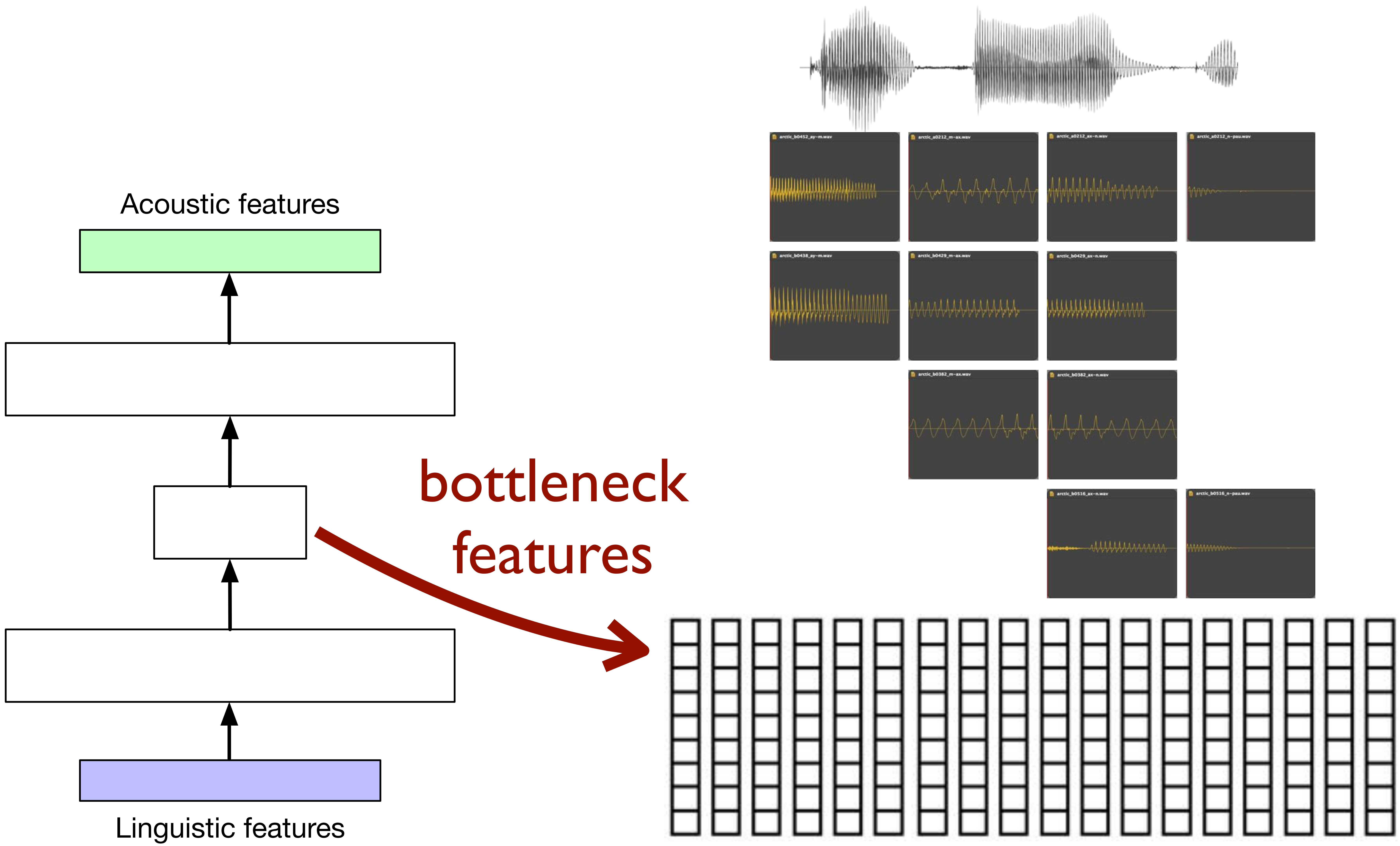
Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder

waveform



any features
you like !

Hybrid speech synthesis is like
Statistical Parametric Speech Synthesis, with a replacement for the vocoder

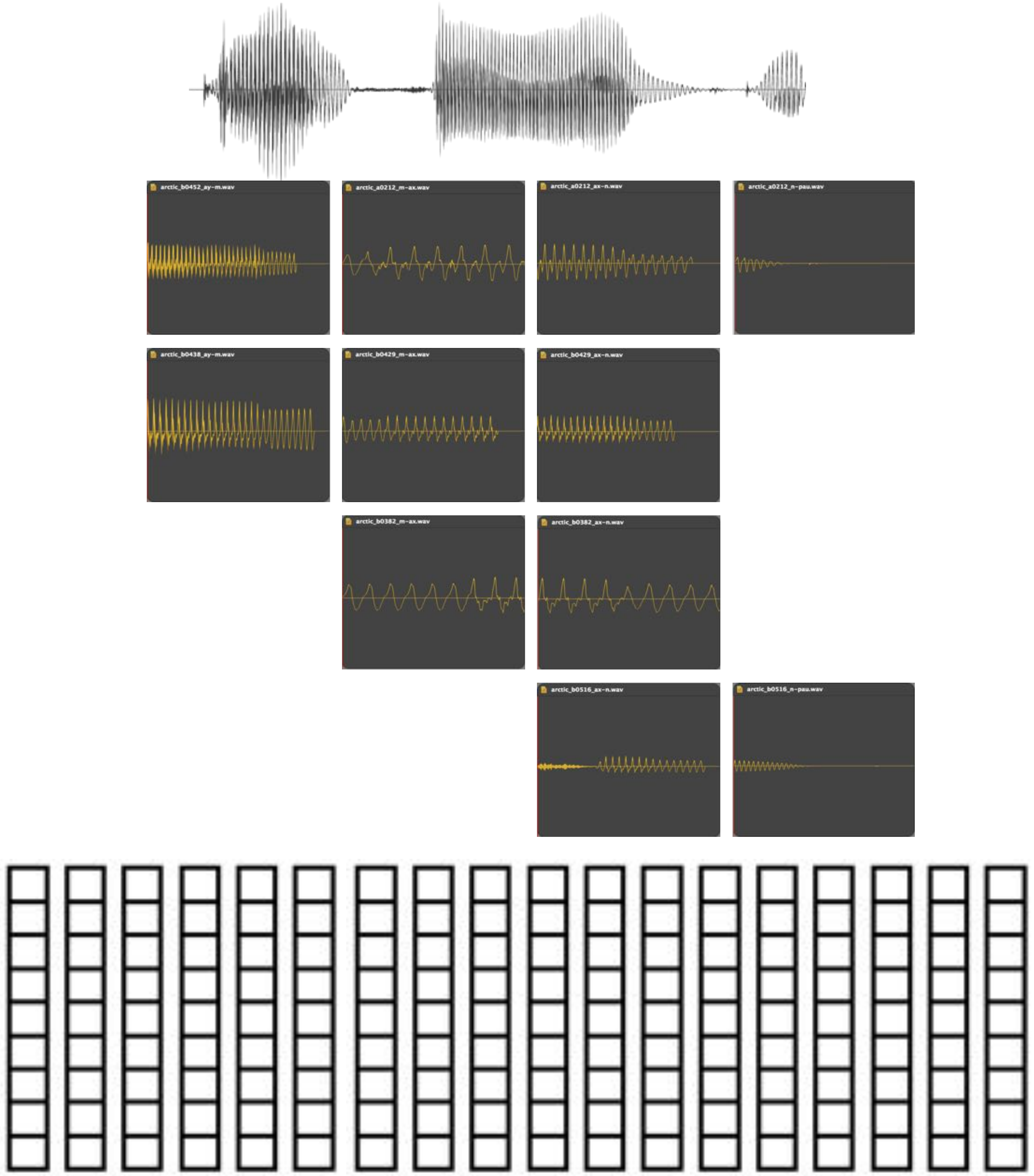


Acoustic features

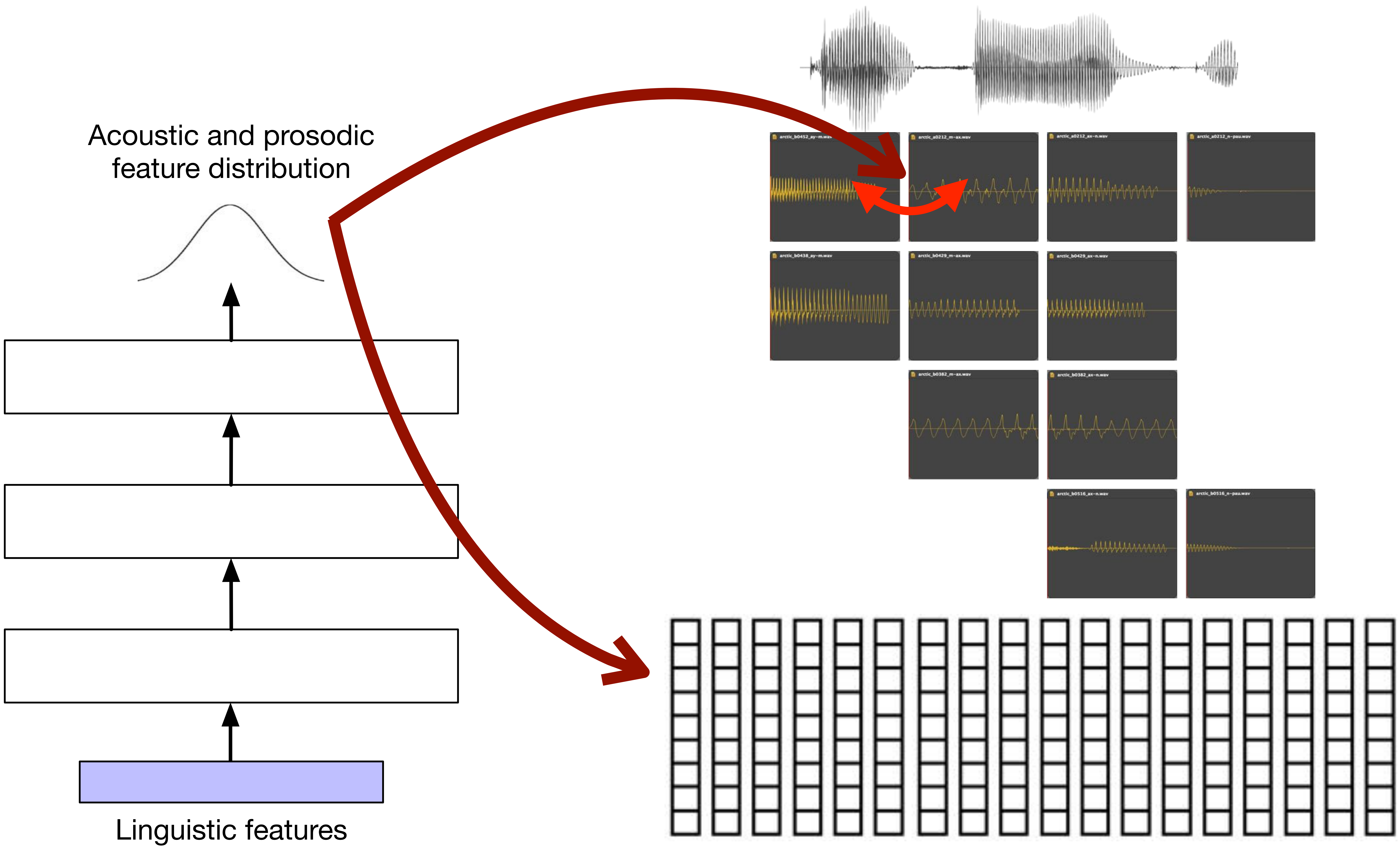
bottleneck features

Linguistic features

Hybrid speech synthesis
with a mixture density network for both target and join costs



Hybrid speech synthesis
with a mixture density network for both target and join costs



See Interspeech poster
Thu-P-9-4-12

Extensions

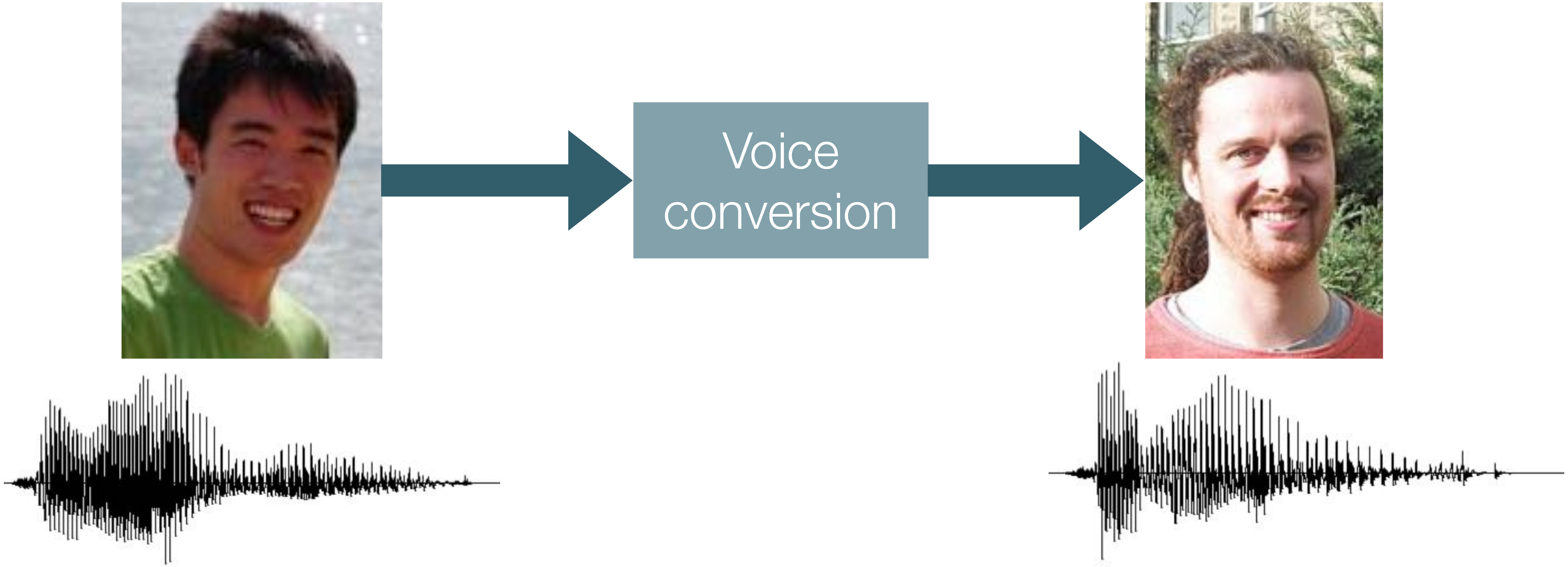
- Hybrid speech synthesis
  - make acoustic feature predictions with Merlin, then select units with Festival
- Voice conversion
  - input speech, instead of text
  - training data is aligned input and output speech (instead of phone labels and speech)
- Speaker adaptation
  - augmenting the input
  - adapting hidden layers
  - transforming the output

Voice Conversion

- Manipulate source speaker's voice to sound like target without changing language content

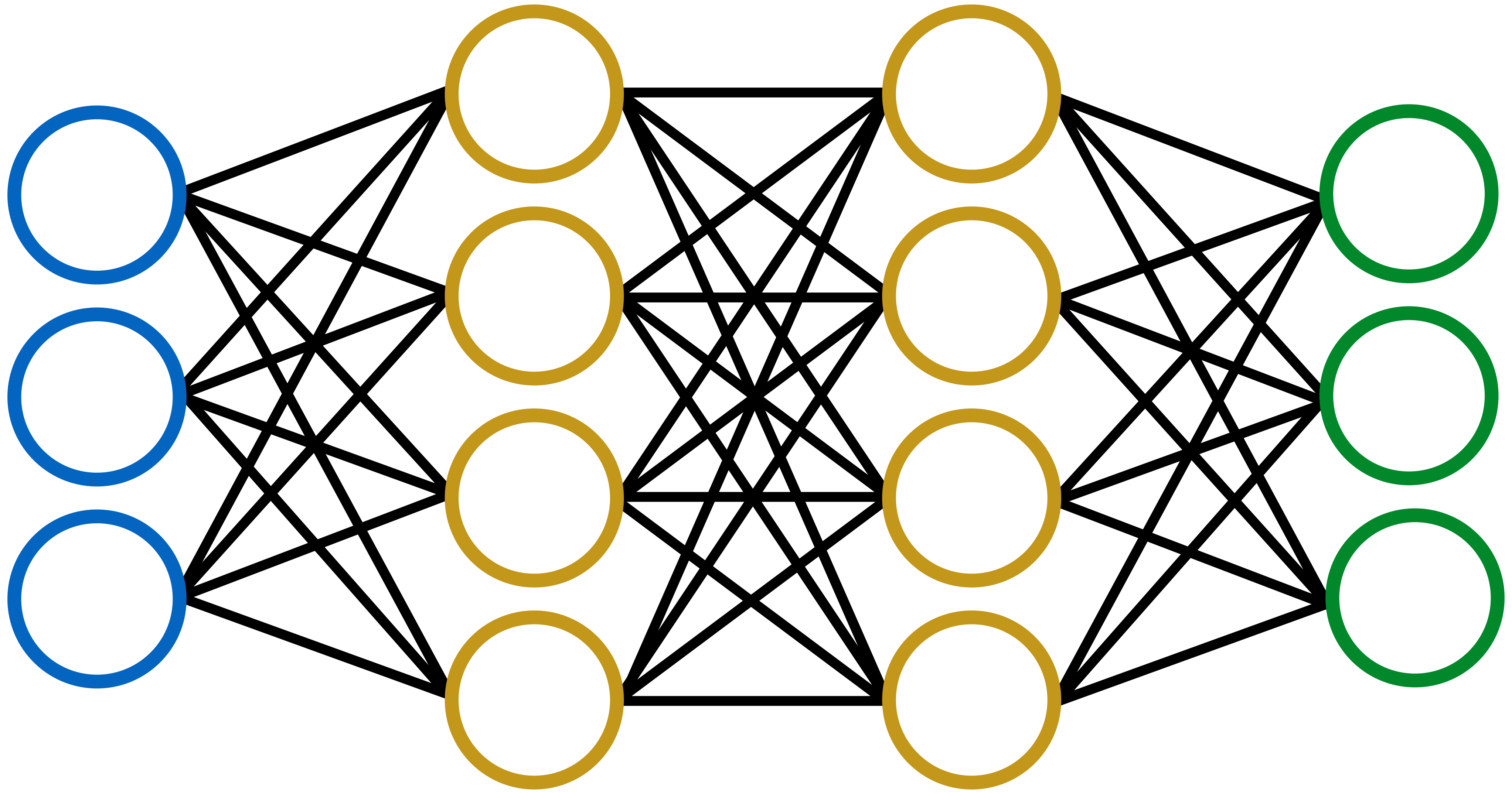
Voice Conversion

- Manipulate source speaker's voice to sound like target without changing language content

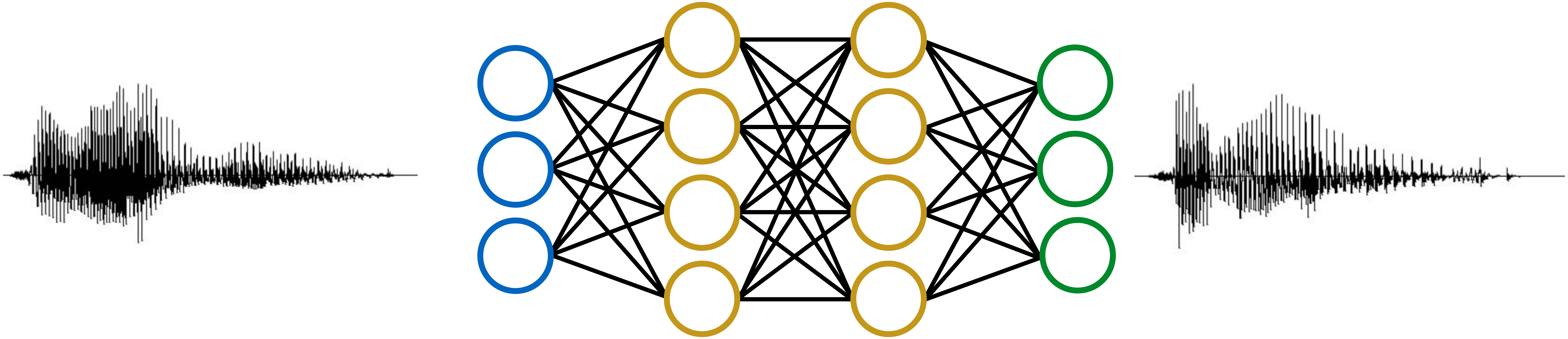


Voice conversion

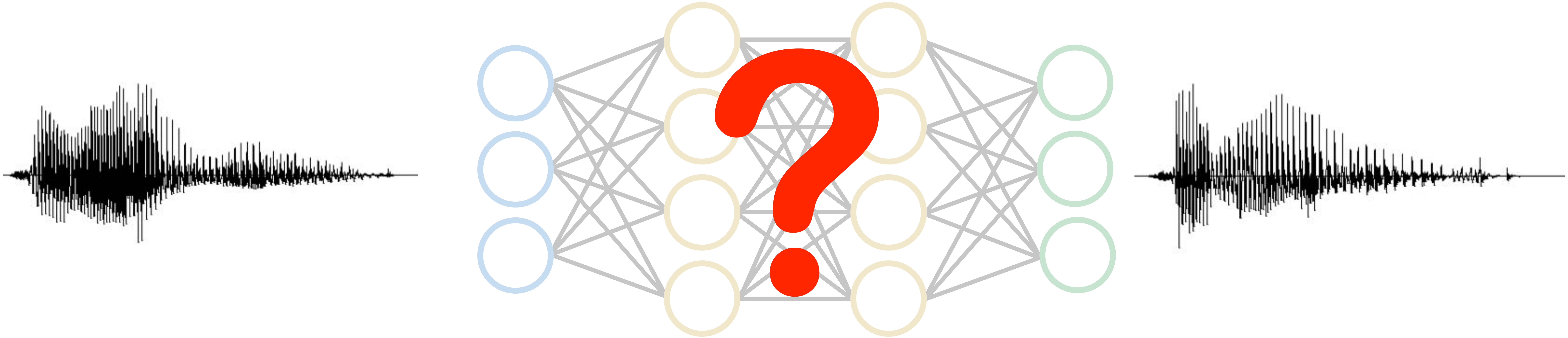
Voice Conversion using a neural network



Voice Conversion using a neural network



Voice Conversion using a neural network

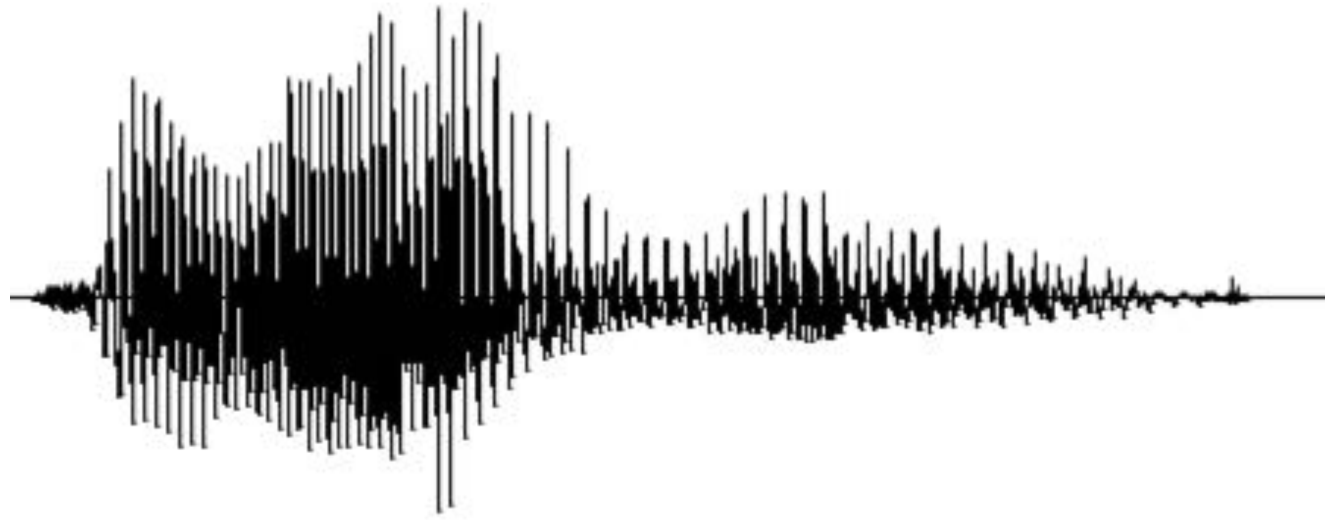


Need solutions for:

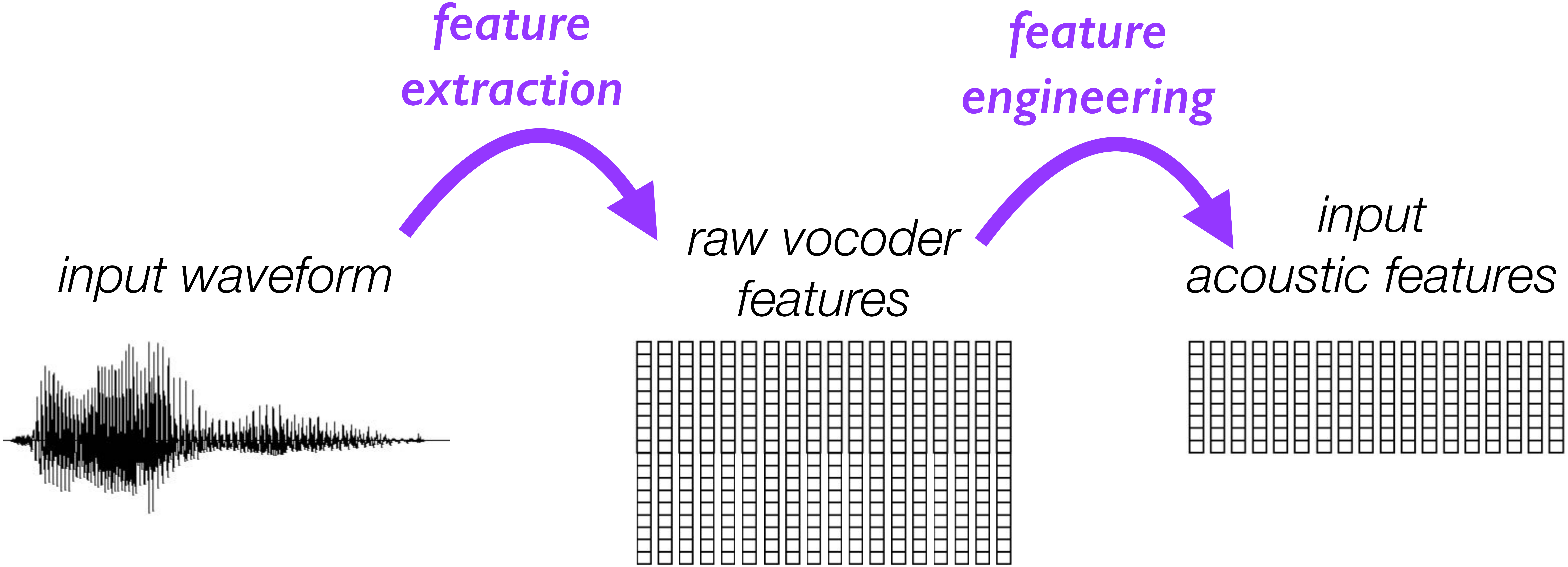
- acoustic feature extraction and engineering
- alignment between input and output

Acoustic feature extraction & engineering for both input and output

input waveform



Acoustic feature extraction & engineering for both input and output



input waveform

feature extraction

raw vocoder features

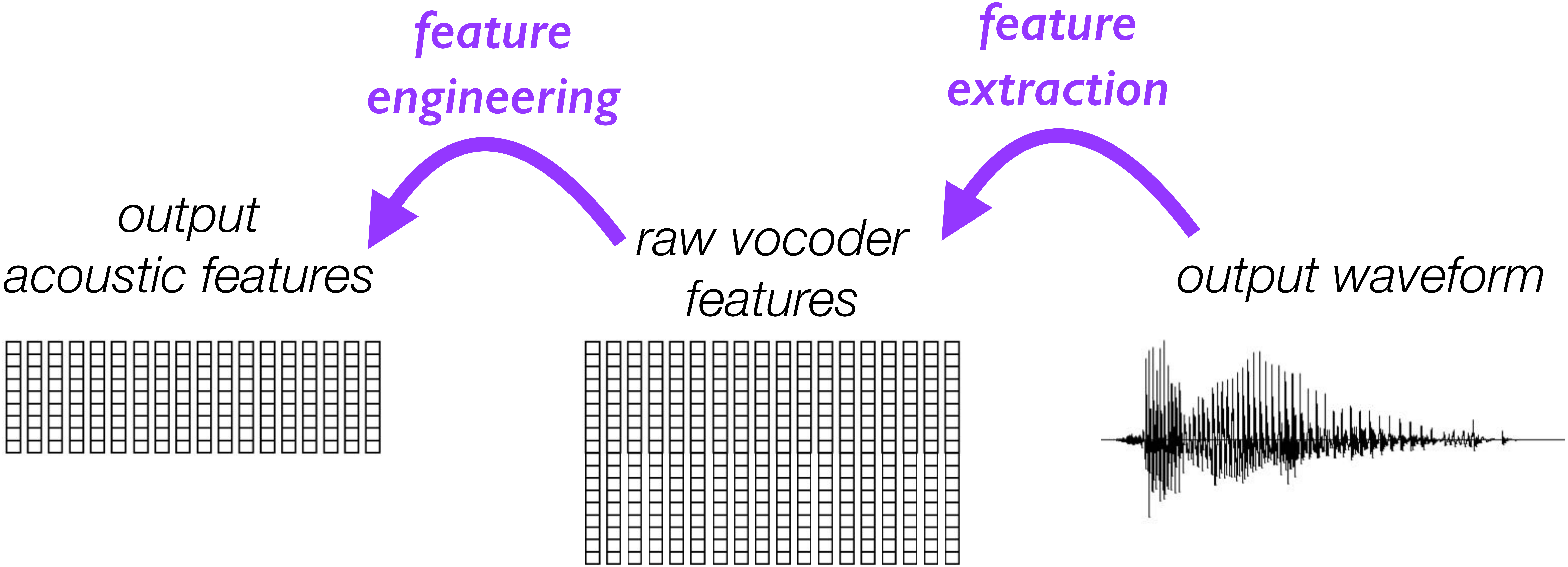
feature engineering

input acoustic features

Acoustic feature extraction & engineering for both input and output

output waveform

Acoustic feature extraction & engineering for both input and output



feature engineering

feature extraction

output acoustic features

raw vocoder features

output waveform

Alignment of input and output

- extract acoustic features from waveforms
- use Dynamic Time Warping (DTW)

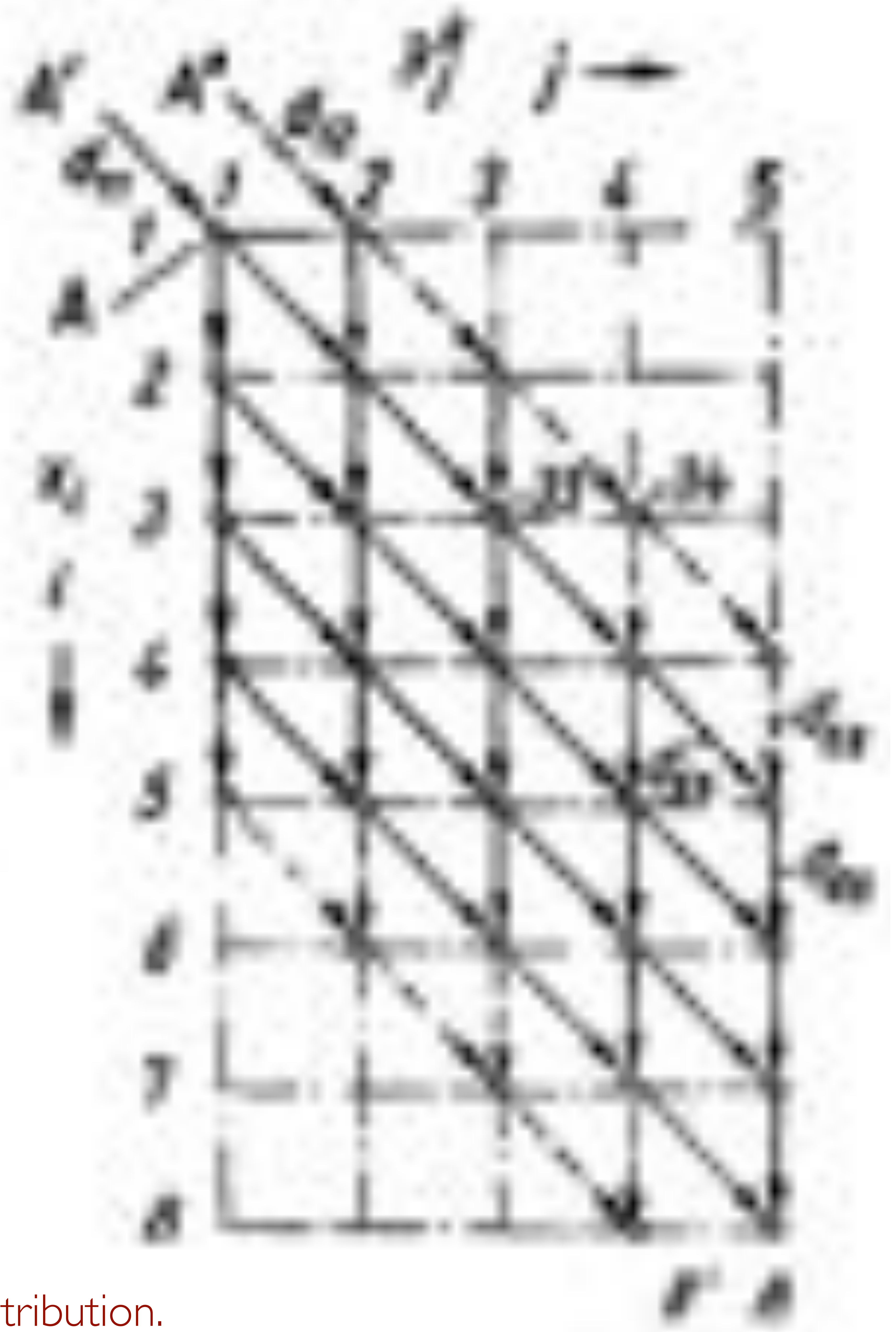
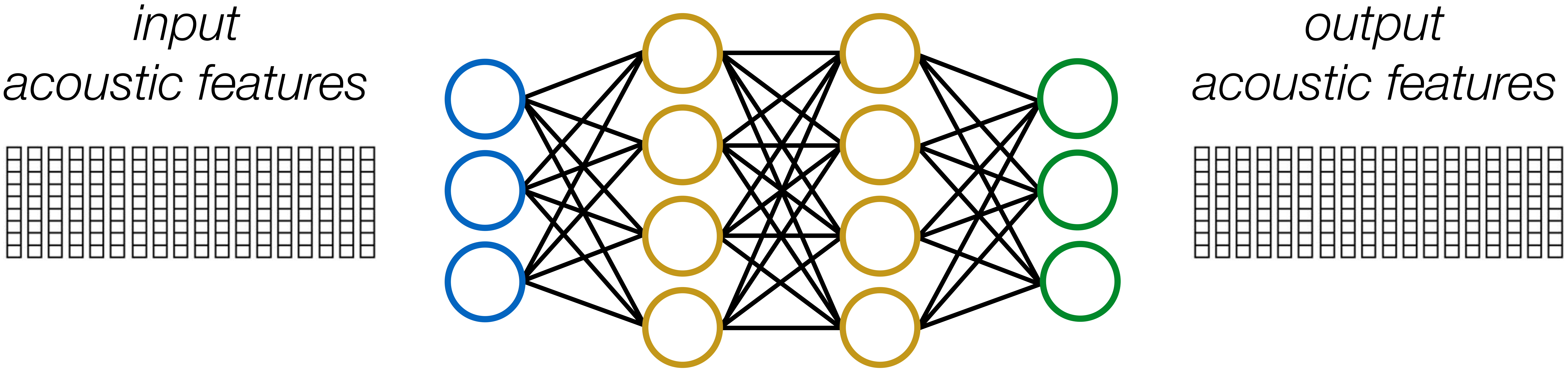


Figure from T. K. Vintsyuk "Speech discrimination by dynamic programming", Cybernetics 4(1) pp 52–57, January 1968

Simplest approach: aligned input and output features + frame-by-frame regression



input
acoustic features

output
acoustic features

Of course, we can do better than a feedforward network

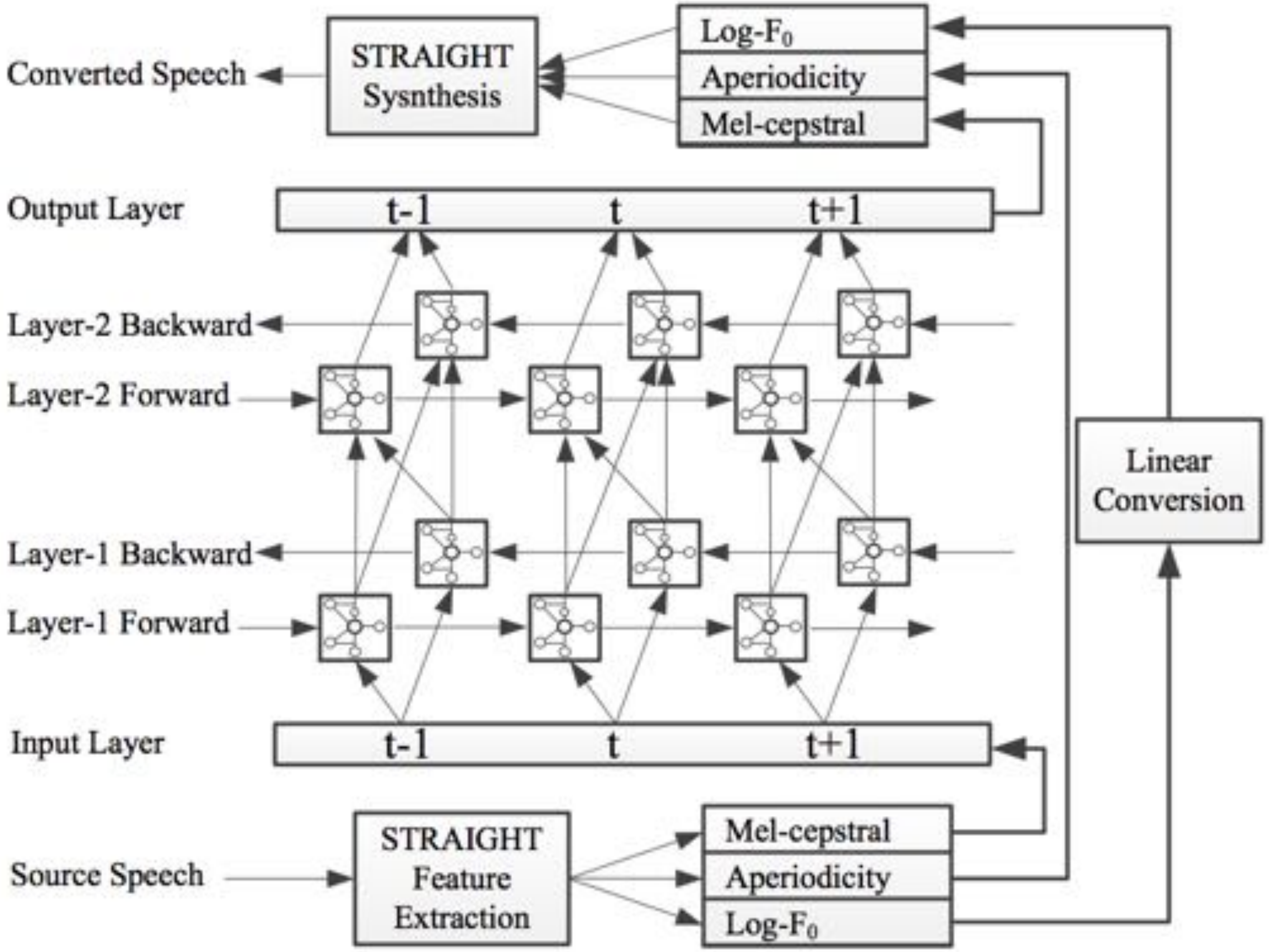
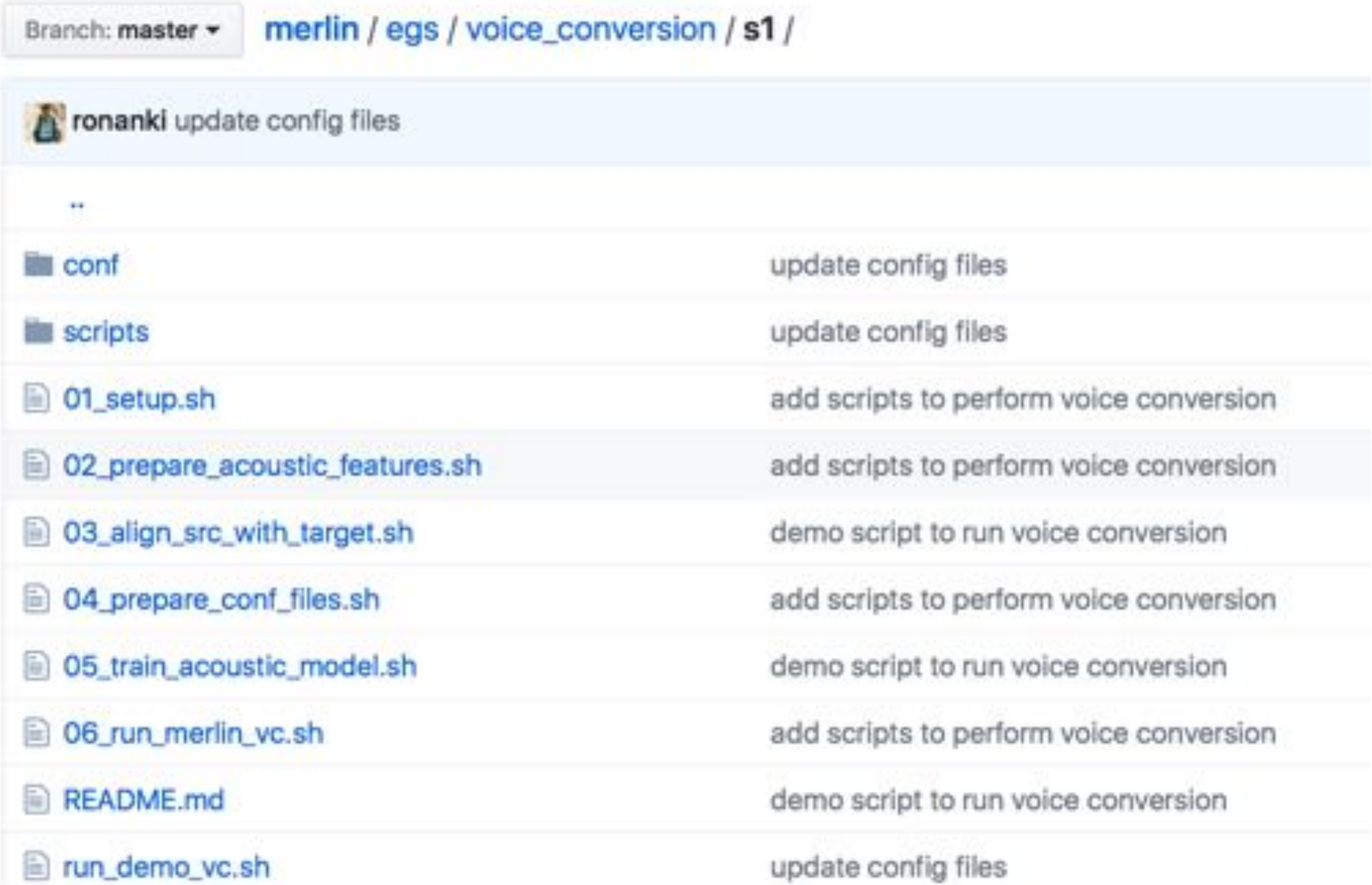


Figure from: Sun et al "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks" ICASSP 2015



merlin / egs / voice_conversion / s1 /

ronanki update config files

..	
conf	update config files
scripts	update config files
01_setup.sh	add scripts to perform voice conversion
02_prepare_acoustic_features.sh	add scripts to perform voice conversion
03_align_src_with_target.sh	demo script to run voice conversion
04_prepare_conf_files.sh	add scripts to perform voice conversion
05_train_acoustic_model.sh	demo script to run voice conversion
06_run_merlin_vc.sh	add scripts to perform voice conversion
README.md	demo script to run voice conversion
run_demo_vc.sh	update config files

03_align_src_with_target.sh

```
src_feat_dir=$1
tgt_feat_dir=$2
src_aligned_feat_dir=$3

src_mgc_dir=$src_feat_dir/mgc
tgt_mgc_dir=$tgt_feat_dir/mgc


echo "Align source acoustic features with target acoustic features..."
python ${MerlinDir}/misc/scripts/voice_conversion/dtw_aligner_festvox.py ${MerlinDir}/tools
${src_feat_dir} ${tgt_feat_dir} ${src_aligned_feat_dir} ${bap_dim}
```

phonealign

classic DTW alignment

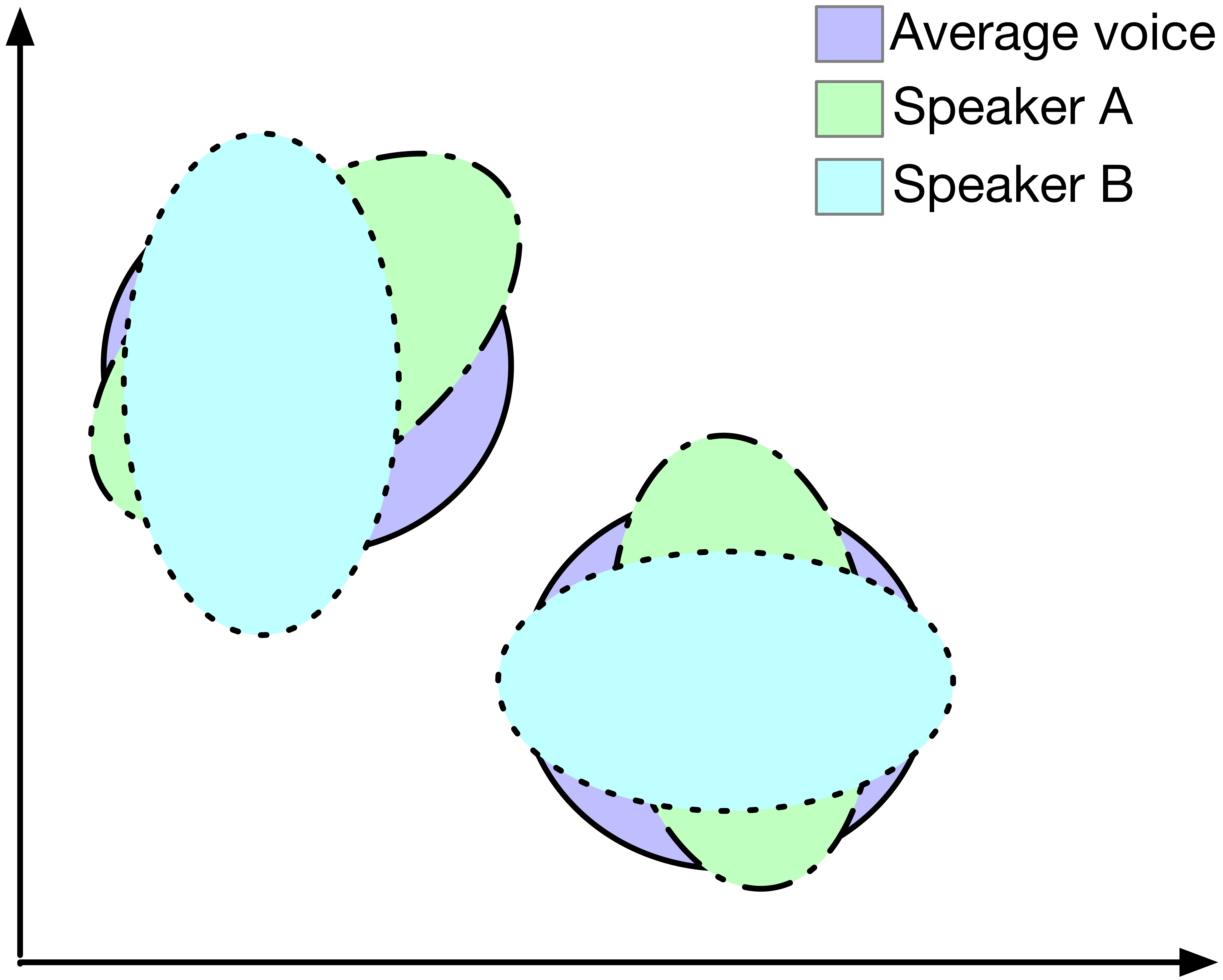
```
for (i=1; i < itrack.num_frames(); i++)
{
for (j=1; j < otrack.num_frames(); j++)
{
    dpt(i,j) = frame_distance(itrack,i,otrack,j);
    if (dpt(i-1,j) < dpt(i-1,j-1))
    {
  if (dpt(i,j-1) < dpt(i-1,j))
  {
      dpt(i,j) += dpt(i,j-1);
      dpp(i,j) = 1; // hold
  }
  else
  {   // horizontal best
      dpt(i,j) += dpt(i-1,j);
      dpp(i,j) = -1; // jump
  }
    }
    else if (dpt(i,j-1) < dpt(i-1,j-1))
    {
  dpt(i,j) += dpt(i,j-1);
  dpp(i,j) = 1; // hold
    }
    else
    {
  dpt(i,j) += dpt(i-1,j-1);
  dpp(i,j) = 0;
    }
```

Extensions

- Hybrid speech synthesis
  - make acoustic feature predictions with Merlin, then select units with Festival
- Voice conversion
  - input speech, instead of text
  - training data is aligned input and output speech (instead of phone labels and speech)
- <u>Speaker adaptation</u>
  - augmenting the input
  - adapting hidden layers
  - transforming the output

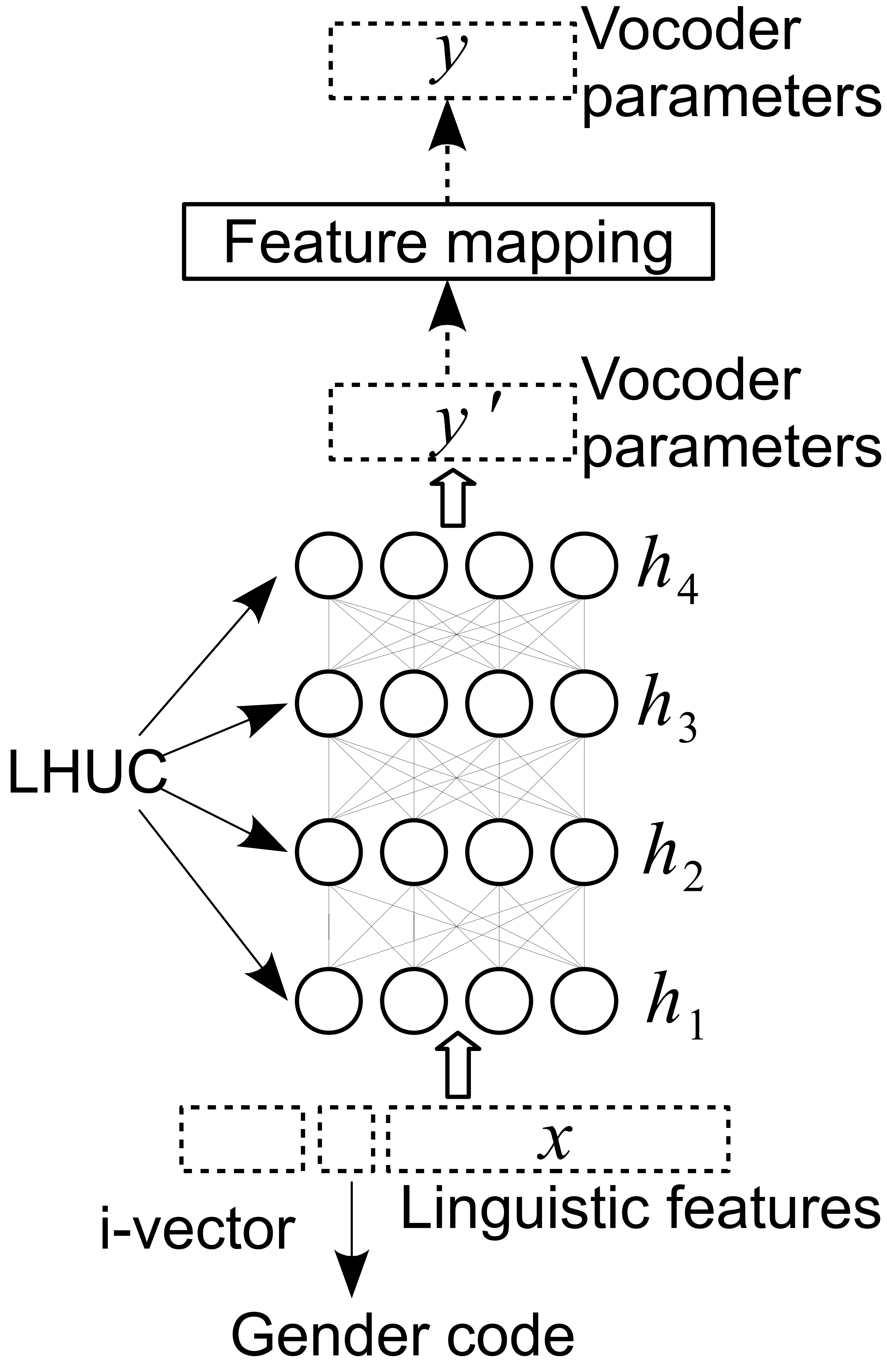
Speaker adaptation

- Create a new voice with only a short recording of speech from the target speaker



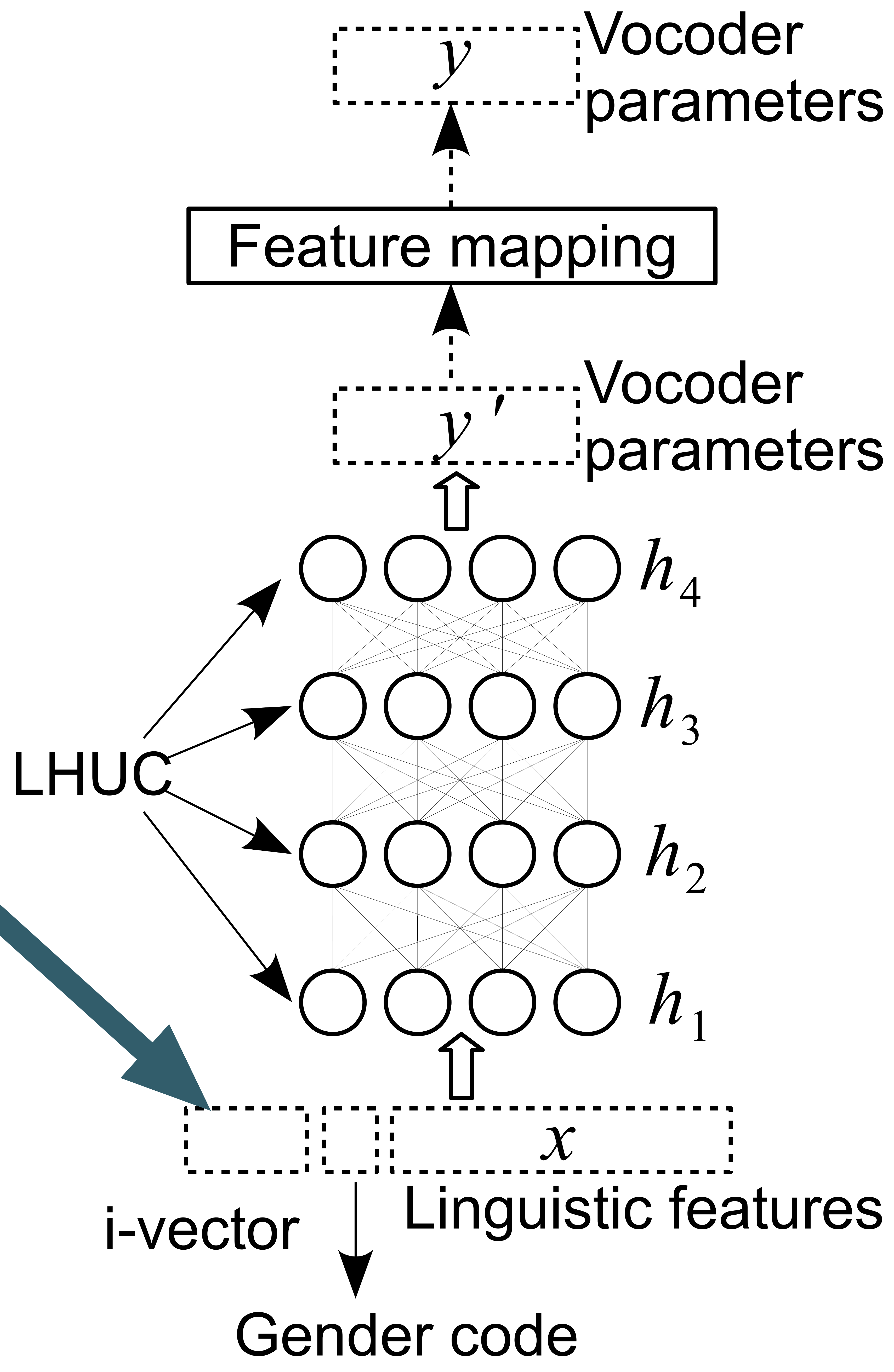
Speaker adaptation for DNNs

- additional input features
- apply transformation (voice conversion) to output features
- learn a modification of the model parameters (LHUC)
- shared layers / "hat swapping"
- retrain ('fine tune') entire model on target speaker data



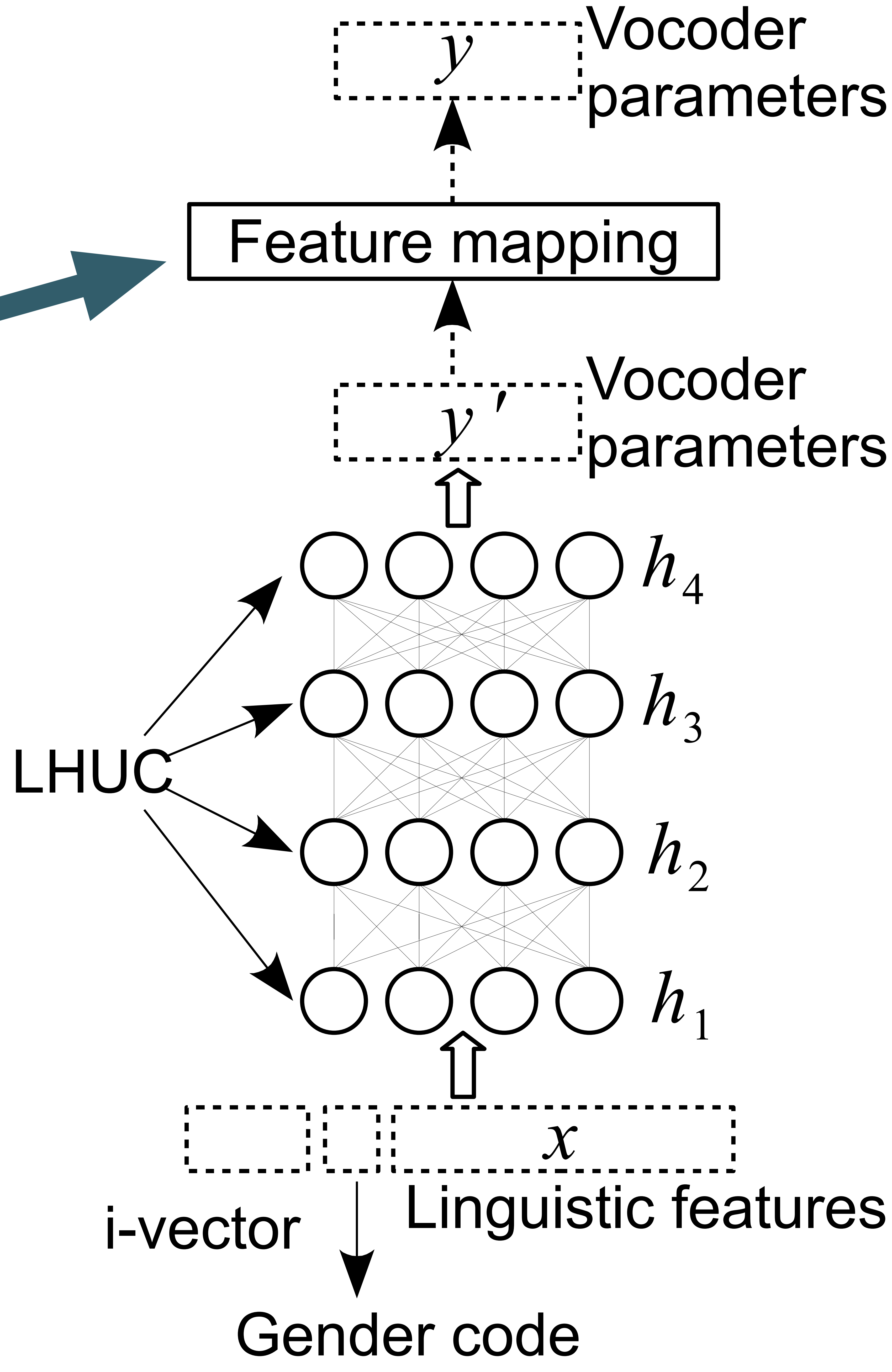
Speaker adaptation for DNNs

- additional input features
- apply transformation (voice conversion) to output features
- learn a modification of the model parameters (LHUC)
- shared layers / "hat swapping"
- retrain ('fine tune') entire model on target speaker data



Speaker adaptation for DNNs

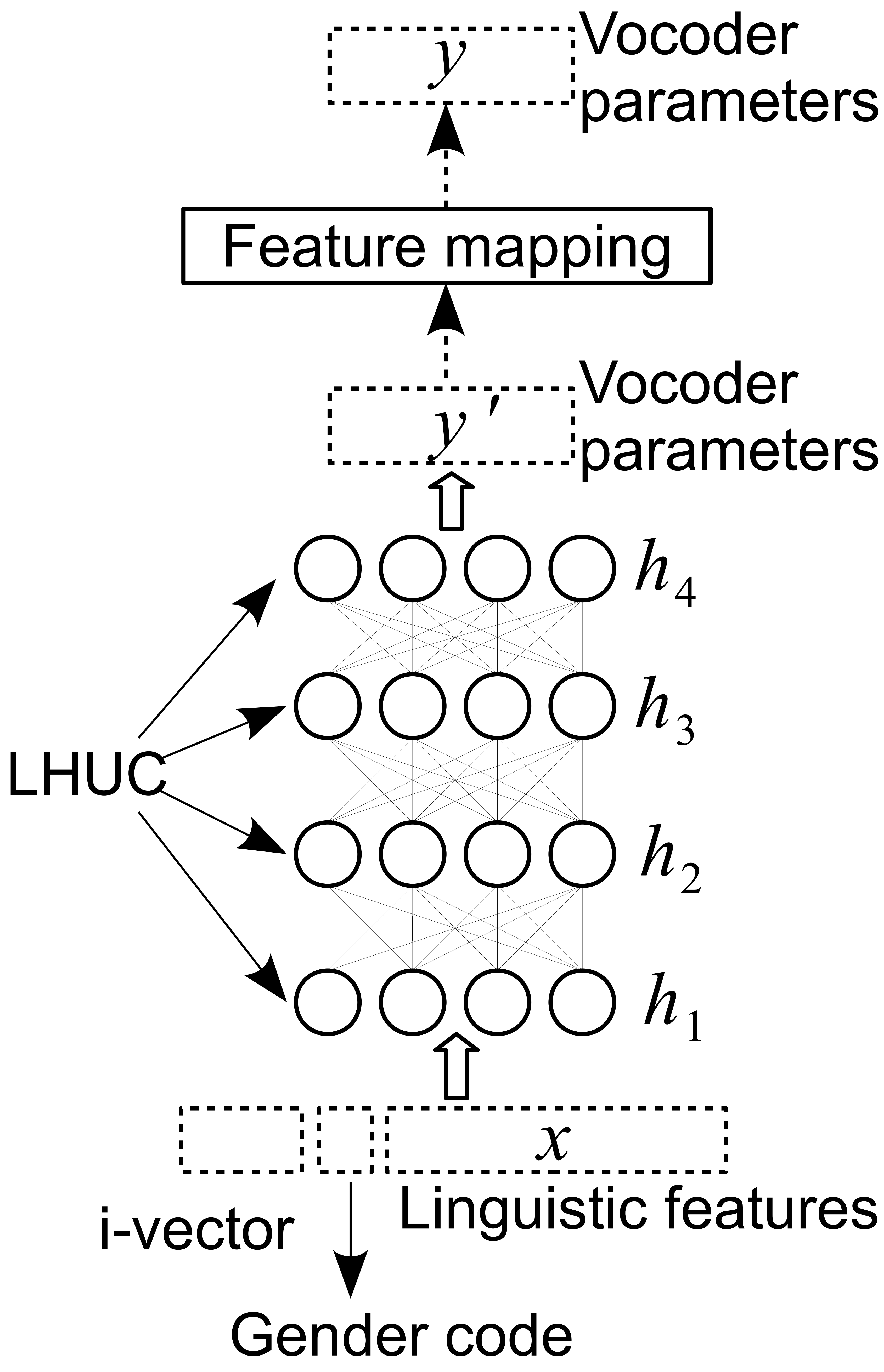
- additional input features
- apply transformation (voice conversion) to output features
- learn a modification of the model parameters (LHUC)
- shared layers / "hat swapping"
- retrain ('fine tune') entire model on target speaker data



Speaker adaptation for DNNs

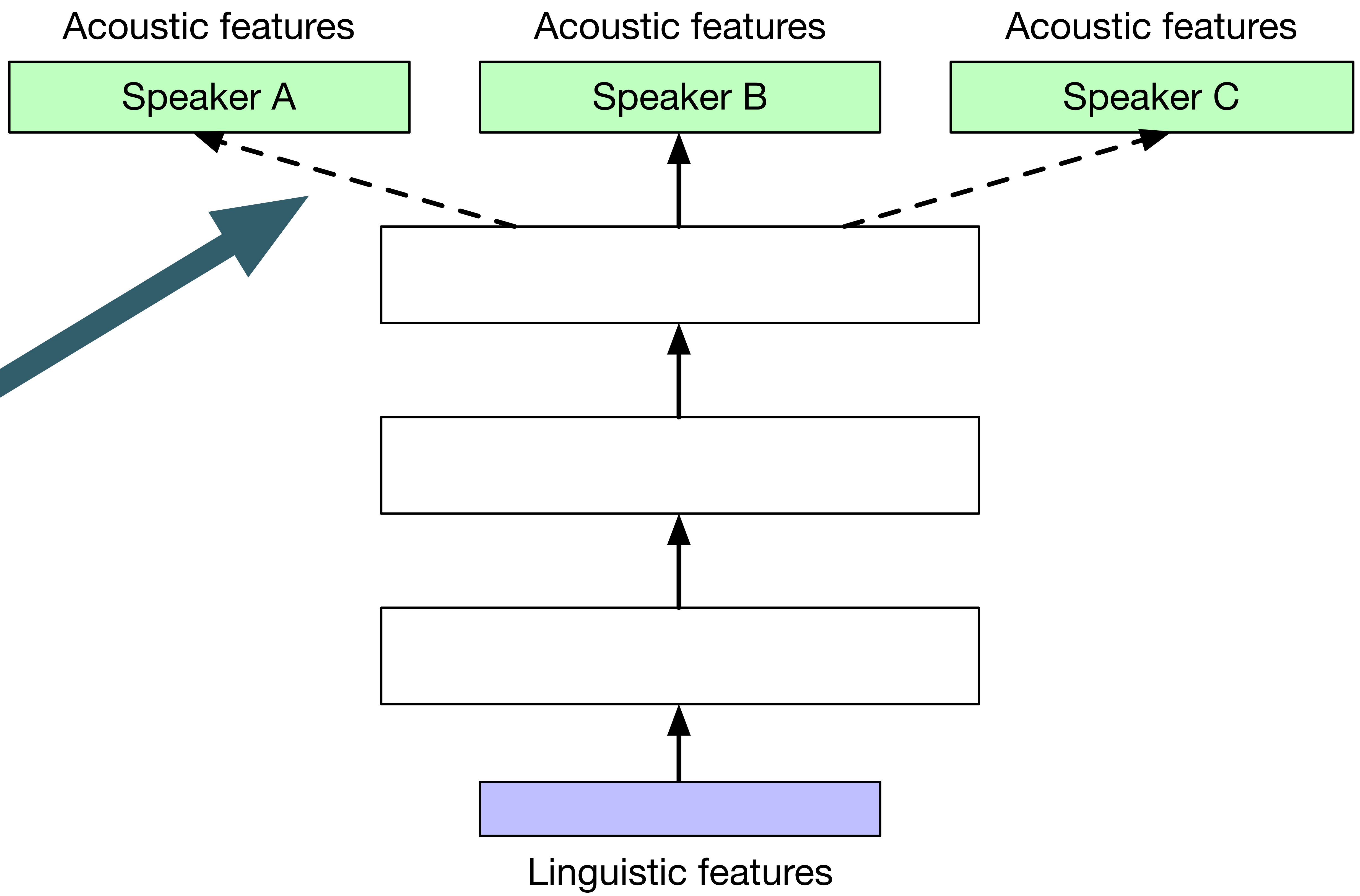
- additional input features
- apply transformation (voice conversion) to output features
- learn a modification of the model parameters (LHUC)
- shared layers / "hat swapping"
- retrain ('fine tune') entire model on target speaker data

$$h_k = g(\gamma_k) \cdot f(\mathbf{w}_k \times \mathbf{x}^T)$$



Speaker adaptation for DNNs

- additional input features
- apply transformation (voice conversion) to output features
- learn a modification of the model parameters (LHUC)
- shared layers / “hat swapping”
- retrain (‘fine tune’) entire model on target speaker data



That’s all - thank-you for attending !

Reading list

Speech synthesis in general

- Paul Taylor. "Text-to-speech synthesis." Cambridge University Press, Cambridge, 2009. ISBN 0521899273
- http://www.speech.zone/courses/speech-synthesis - includes further reading

Front end

- Conventional front end
  - Paul Taylor "Text-to-speech synthesis" - *the first half of the book is mainly about this topic*
  - P. Ebden and R. Sproat. "The Kestrel TTS Text Normalization System." Journal of Natural Language Engineering 21(3), May 2015. DOI: 10.1017/S1351324914000175
- Machine learning for text processing
  - O. Watts, S. Gangireddy, J. Yamagishi, S. King, S. Renals, A. Stan, and M. Giurgiu. "Neural net word representations for phrase-break prediction without a part of speech tagger." Proc. ICASSP, Florence, Italy, May 2014. DOI: 10.1109/ICASSP.2014.6854070
  - R. Sproat, N. Jaitly "RNN Approaches to Text Normalization: A Challenge" arXiv: 1611.00068

Signal processing / vocoding for speech synthesis

- F. Espic, C. Valentini-Botinhao and S. King. "Direct Modelling of Magnitude and Phase Spectra for Statistical Parametric Speech Synthesis" Proc. Interspeech, Stockholm, Sweden, Aug. 2017.
- M. Morise, F. Yokomori, and K. Ozawa. "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications." IEICE Trans. Information & Systems, E99-D(7), 2016.
- H. Kawahara, I. Masuda-Kasuse and A. de Cheveigne. "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds." Speech Communication 27(3-4), Apr. 1999. DOI: 10.1016/S0167-6393(98)00085-5 - *the STRAIGHT vocoder*

Speech synthesis using DNNs

- H. Zen, A. Senior and M. Schuster "Statistical parametric speech synthesis using deep neural networks." Proc. ICASSP, Vancouver, BC, Canada, May 2013. DOI: 10.1109/ICASSP.2013.6639215
- O. Watts, G. Eje Henter, T. Merritt, Z. Wu and S. King. "From HMMs to DNNs: where do the improvements come from?" Proc. ICASSP, Shanghai, China, Apr. 2016. DOI: 10.1109/ICASSP.2016.7472730

Adaptation for speech synthesis using DNNs

- Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King. "A study of speaker adaptation for DNN-based speech synthesis." Proc Interspeech, Dresden, Germany, Sep. 2015.
- N. Hojo, Y. Ijima, and H. Mizuno. "An Investigation of DNN-Based Speech Synthesis Using Speaker Codes." Proc. Interspeech, San Francisco, CA, USA, Sep. 2016.
- H.T. Luong, S. Takaki, G. Henter, J. Yamagishi. "Adapting and controlling DNN-based speech synthesis using input codes." Proc. ICASSP, New Orleans, LA, USA, Mar. 2017. DOI: 10.1109/ICASSP.2017.7953089

Hybrid speech synthesis

- Paul Taylor "Text-to-speech synthesis", 2009, Cambridge University Press, Cambridge, ISBN 0521899273 - *section 16.4 describes the "Acoustic Space Formulation" target cost, which is essentially hybrid synthesis*
- Y. Qian, F. K. Soong and Z. J. Yan "A Unified Trajectory Tiling Approach to High Quality Speech Rendering" IEEE Trans. Audio, Speech, and Language Proc. 21(2), Feb. 2013. DOI:10.1109/TASL.2012.2221460
- T. Merritt, R. A. J. Clark, Z. Wu, J. Yamagishi and S. King. "Deep neural network-guided unit selection synthesis." Proc. ICASSP, Shanghai, China, Mar. 2016. DOI: 10.1109/ICASSP.2016.7472658
- T. Capes, P. Coles, A. C., L. Golipour, A. Hadjitarkhani, Q. Hu, N. Huddleston, M. Hunt, J. Li, M. Neeracher, K. Prahallad, T. Raitio, R. Rasipuram, G. Townsend, B. Williamson, D. Winarsky, Z. Wu, H. Zhang. "Siri On-Device Deep Learning-Guided Unit Selection Text-to-Speech System." Proc. Interspeech 2017, Stockholm, Sweden, Aug. 2017.

Voice conversion

- L. Sun, S. Kang, K. Li, and H. Meng. "Voice conversion using deep bidirectional long short-term memory based recurrent neural networks." Proc. ICASSP, Brisbane, Australia, Apr. 2015. DOI: 10.1109/ICASSP.2015.7178896

Surveys, review articles, miscellaneous

- Simon King. "Measuring a decade of progress in Text-to-Speech." Loquens, 1(1), Jan. 2014. DOI: 10.3989/loquens.2014.006
- Z. Ling, S. Kang, H. Zen, A. Senior, M. Schuster, X. Qian, H. Meng and L. Deng. "Deep Learning for Acoustic Modeling in Parametric Speech Generation: A systematic review of existing techniques and future trends." IEEE Signal Processing Magazine 32(3), May 2015. DOI: 10.1109/MSP.2014.2359987
- J. Dines, J. Yamagishi and S. King. "Measuring the Gap Between HMM-Based ASR and TTS." IEEE Journal of Selected Topics in Signal Processing 4(6), Dec. 2010. DOI: 10.1109/JSTSP.2010.2079315 - *demonstrates that doing ASR with TTS features doesn't work very well - which is relevant for alignment in sequence-to-sequence models for TTS*