Speech Processing: Text-to-Speech: Phonemes & Front-End

Module 5

Catherine Lai

19 Oct 2023

Upcoming assessments

Week 5: TTS frontend

Week 6: Waveform Generation

Week 7: No new content 🎉

- Assessment: TTS multiple choice test open Mon-Wed (on Learn)
- Labs will be on as usual Wed
- No lecture on Thursday

Week 8: Intro to ASR/Pattern Matching (Module 7)

- Assessment: TTS Assignment due Monday 6 November 2023, 12 noon
- Labs on Wednesday (shell scripting)
- Lecture on Thursday
- Start on ASR assignment in module lab

Rest of semester….

Week 9: Feature Engineering

Week 10: Hidden Markov Models

Week 11: HMM training for ASR

- Assessment: ASR multiple choice test Mon-Wed (on Learn)

Reading week:

- Assessment: Assignment 2 (ASR) due Thu 7/12/23 12 noon

No end of course exam! 🎉

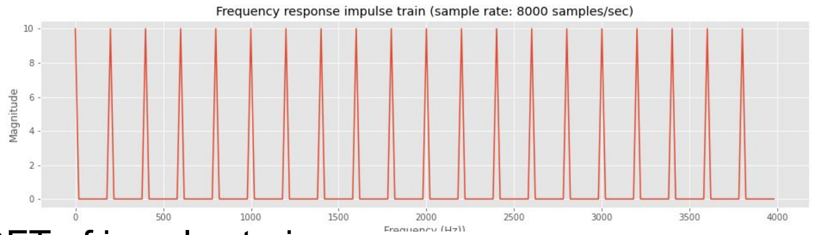
Previously: Source-Filter model

We can describe the source filter model mathematically

- Source=impulse train: frequency of impulses determines F0
- Filter=Infinite Impulse Response filter: a weighted sum of previous inputs and outputs

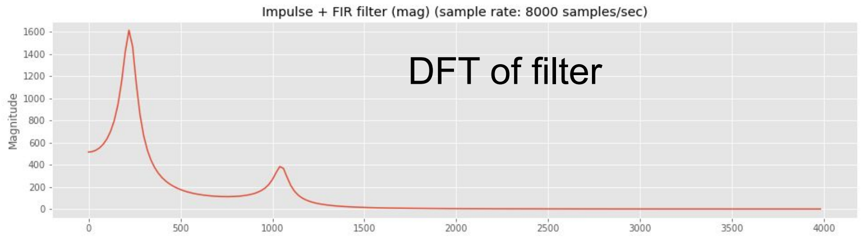
$$s[t] = e[t] + \sum_{k=1}^{K} a[k]s[t-k]$$

Effect of filters in the frequency domain



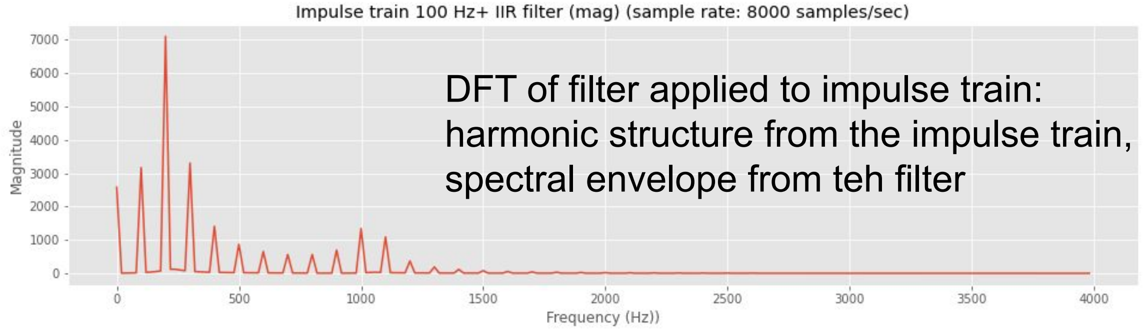
DFT of impulse train shows F0 and it's harmonics

X



DFT of filter

=



DFT of filter applied to impulse train: harmonic structure from the impulse train, spectral envelope from teh filter

Computer, say ‘Text-to-Speech’

- So, in theory, we can use impulse trains and IIR filters to generate recognisable phones
- If we put those phones together we can make words
- If we can make words, we can make sentences…

In theory!

[See Klatt Synthesizer demo]

Text-to-Speech

- Real dynamics of speech are hard to capture using source-filter based models (e.g. formant synthesizers are intelligible but not natural sounding)
- Solution: use real speech where possible → Concatenative synthesis (more details in Module 6)

Text-to-Speech

Two main components:

- **Front-end:** Analyze text, generate a **linguistic specification** of what to actually generate
- **Back-end:** Waveform generation from the linguistic specification

This week focuses on the front end.

TTS Front end

We want to generate speech that is

- Intelligible: you can clearly perceive what words are being said
- Natural: sounds like human speech
- Appropriate: conveys the right meaning in a specific context

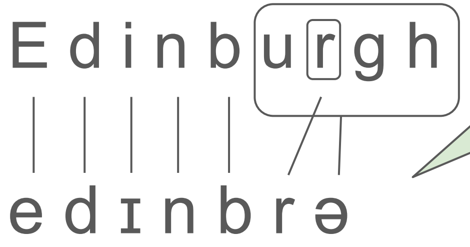
Front-end aim: derive a **linguistic specification** from text that includes the necessary information to generate speech

TTS Front-end

Ideally, we want to develop a linguistic specification that guides what we generate

- Phones
- Syllables
- Words
- Phrases
- Utterances
- Discourses

From Letters to Phones



Who has this pronunciation? Are there other pronunciations?

The mapping between letters used for writing (graphemes) and pronunciation (phones) isn't always straightforward (especially for english)

From Words to Utterances: Phrasing

I live in Edinburgh

aɪ lɪv ɪn edɪnbrə

Once we have the pronunciations of individual words we can put them together. But then which syllables get emphasized?

From Words to Utterances: Phrasing

I live in **E**dinburgh

(aɪ) (lɪv) (ɪn) (**e**)(dɪn)(brə)

Once we have the pronunciations of individual words we can put them together. But then which syllables get emphasized?

Within a word, this is determined by **lexical stress** → has to be learned about English words, but also changes in context

Communicating intent: question intonation

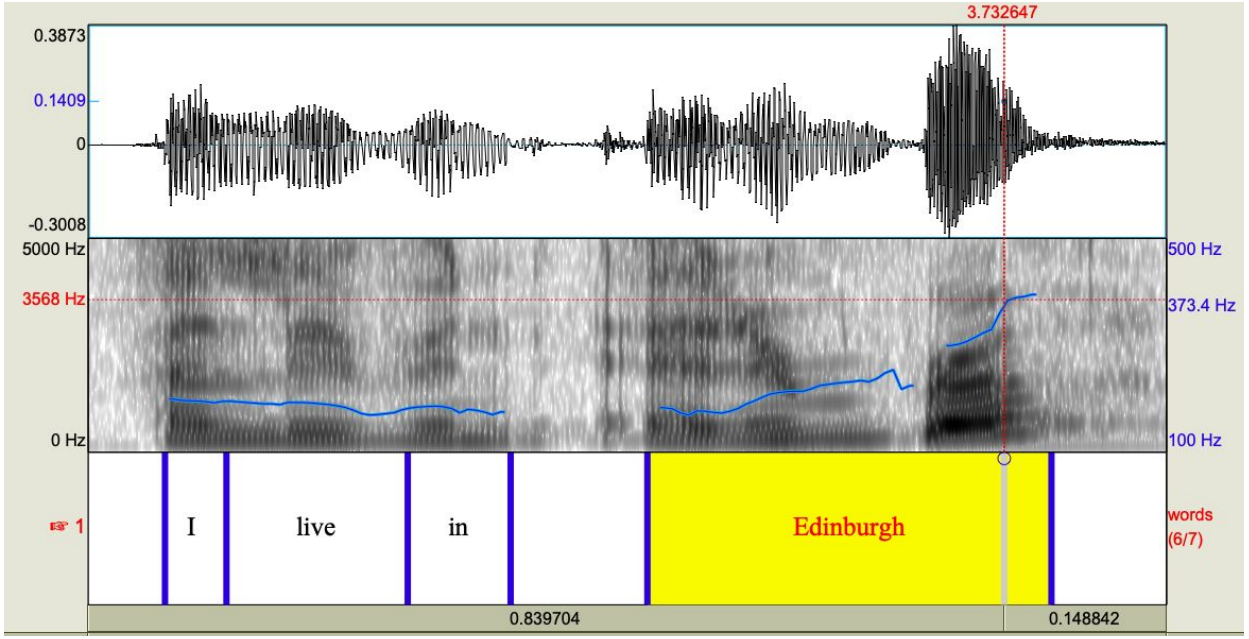
I live in Edinburgh?

aɪ lɪv ɪn edɪnbrə

Beyond the individual phones that determine which words will be perceived, we need to get the non-lexical characteristics of speech right.

This is **prosody**: very broadly the pitch, energy, and timing characteristics of speech.

Communicating intent: question intonation



In English an utterance final pitch rise on a declarative sentence (often) signals a question

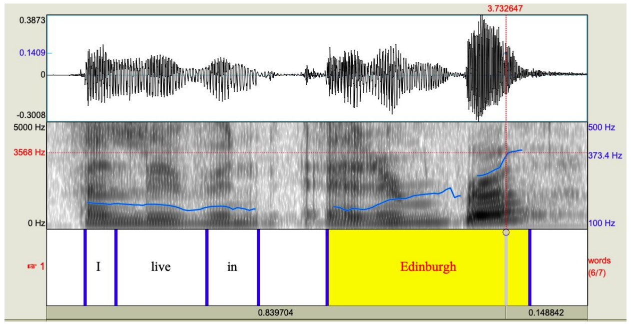
Communicating intent: question intonation

I live in **Edinburgh**?

aɪ lɪv ɪn edɪnbrə

pause emphasis Final pitch rise

“Prosodic break” “Prosodic prominence” “Boundary tone”



In English an utterance final pitch rise on a declarative sentence (often) signals a question

Communicating intent: Contrast

contrast

I don’t **live** in Edinburgh! I **work** in Edinburgh!

aɪ dəʊnt lɪv ɪn edɪnbrə aɪ wərk ɪn edɪnbrə

- Prosodic prominence (emphasis) is used to mark contrast.
- Prominence (usually) manifests on the lexically stressed syllable of a word (in English)

Text Normalization

Text Normalization: Basic steps

Transform the text into a pronounceable state

- Tokenize: split characters in text into word like units
- Detect non-standard words: rewrite as standard words
- Resolve Ambiguity: e.g., POS tagging

What sort of ambiguity is relevant?

Q: What sort of ambiguity is relevant for generating correct pronunciations?

For example, do we care about:

- Words that have different meanings but sound the same (homophones)?
- Words that have the same spelling but sound different (homographs)?

Text Normalization: Tokenization

Our first step is to split the sequence of characters into tokens (roughly words). For English, whitespace is a usually a good starting point

Which tokens here need normalization?

Example Text from BBC news:

Staff on the Caledonian Sleeper will hold two 24-hour strikes. One from 11:59 on Sunday 31 October and one on Thursday 11 November.

Unite Scotland has also said about 1,000 workers across the Stagecoach Group had backed strike action at the end of October which would affect COP26 bus travel in central Scotland.

Text Normalization: non-standard words

From the tokens, we need to detect non-standard words and convert them into pronounceable forms:

Is there a single rule we can use to map these numbers to pronounceable words? Why or why not?

Example Text from BBC news:

Staff on the Caledonian Sleeper will hold two 24-hour strikes. One from 11:59 on Sunday 31 October and one on Thursday 11 November.

Unite Scotland has also said about 1,000 workers across the Stagecoach Group had backed strike action at the end of October which would affect COP26 bus travel in central Scotland.

Text Normalization: non-standard words

We need to detect non-standard words and convert them into pronounceable forms:

Text from BBC news:

Staff on the Caledonian Sleeper will hold two twenty four hour strikes. One from eleven fifty nine on Sunday thirty first of October and one on Thursday eleventh of November.

Unite Scotland has also said about one thousand workers across the Stagecoach Group had backed strike action at the end of October which would affect COP twenty six bus travel in central Scotland.

Non-Standard Words

- Numbers: e.g. 1,520, 1.50, 2021, 2,02
- Non-letter symbols: e.g. £, $, &
- Acronyms: e.g. COP, SEC, BBC, FTSE
- Abbreviations: e.g. Dr, St, kHz,

Lots of ambiguity!

Pronunciations don't always correspond to the ordering of the symbols:
£1.50 → One **pound** fifty

Different pronunciation for different types of numbers: e.g. years versus measurements: the year 2022 v 2022 cm

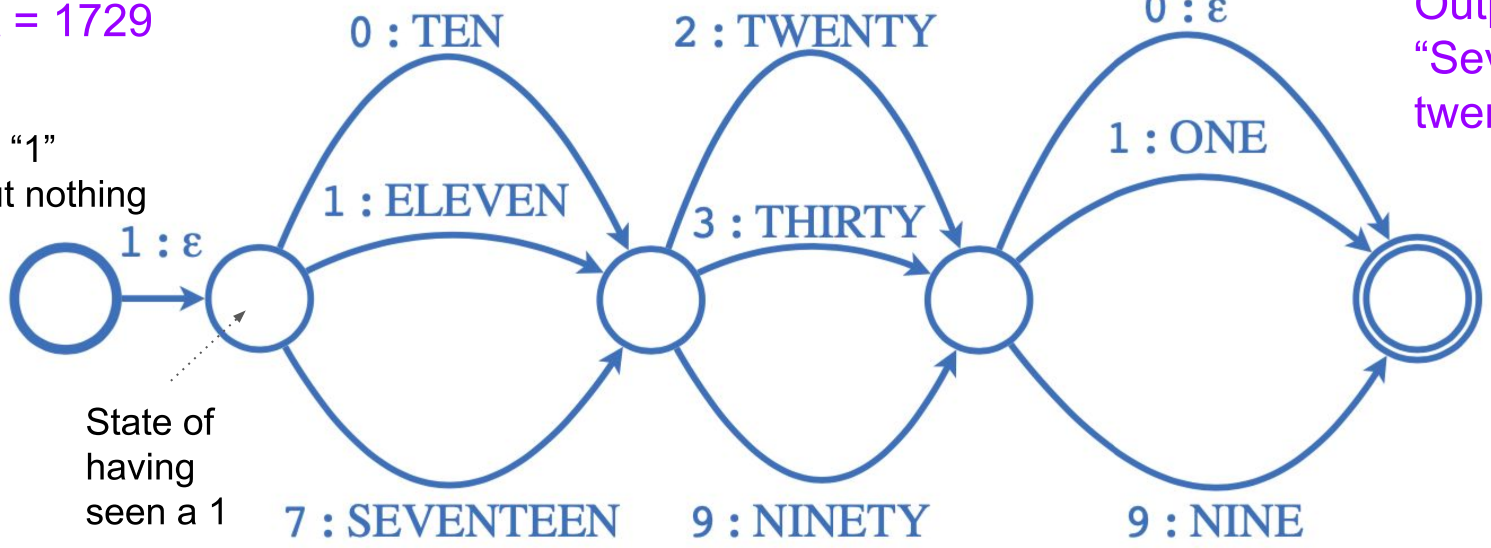
Different rules for different locales: e.g. fifteen hundred (and) twenty

Acronyms may be pronounced as separate letters or as "words"

Expressing rules computationally: Finite state transducer

Convert one string into another:

input = 1729



See a "7", output "seventeen"

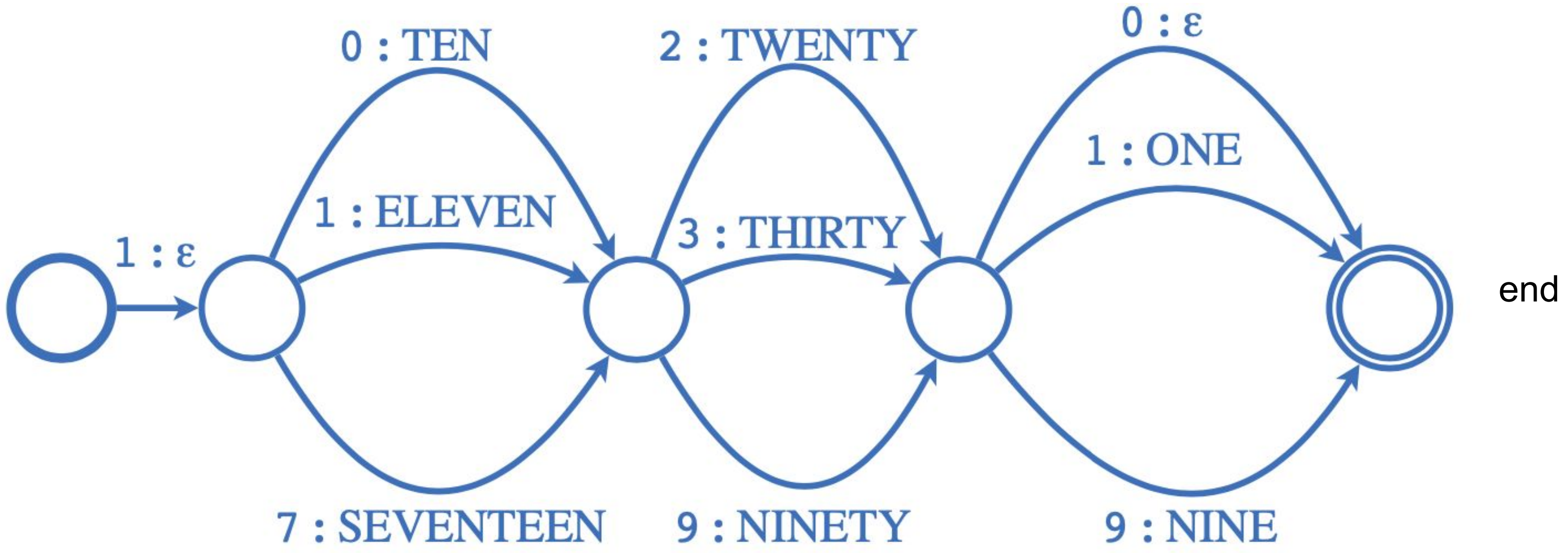
See a "9", output "nine"

Output: "Seventeen twenty nine"

Expressing rules computationally: Finite state transducer

Q: What strings does this FST generate?

Q: How can we expand this to cover all years from from 1066-1999? How about further?



Punctuation ambiguity

- TTS systems (still!) often work on a sentence by sentence basis
- So sentence segmentation is required
- But a full stop “.” doesn’t always represent the end of sentence and sentences don’t always end with full stops:

e.g. “The lecturer lives in Fife Dr. Richmond is actually a linguist by training”

[Let’s see how festival handles this…]

Word sense ambiguity: Homographs

Words that are written the same way can have different pronunciations:

- I **polish** my nails in **Polish** class

  Verb Noun

- She **records** her **records** in the office

  Verb Noun

- My **bass** guitar has a picture of a sea **bass** on it

  Noun Noun

We can disambiguate some cases by their Part Of Speech (POS)

But other cases need more contextual information for disambiguation, e.g. nearby words

How to resolve ambiguity?

Hand-written rules (can be implemented in Finite State Transducer - see video)

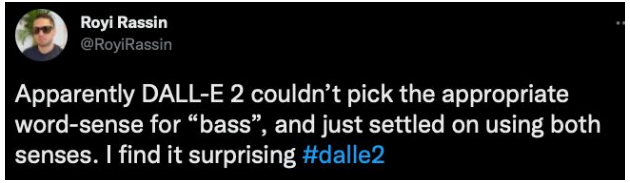
St. → [Capitalized Word] / Street / _

St. → _ / Saint / [Capitalized word]

- Alternative 1: Learn disambiguation rules from the data
  - e.g. decision trees (later today)
- Alternative 2: Learn probabilistic relationship from the data
  - e.g. Statistical Part of Speech Tagging (other courses!)

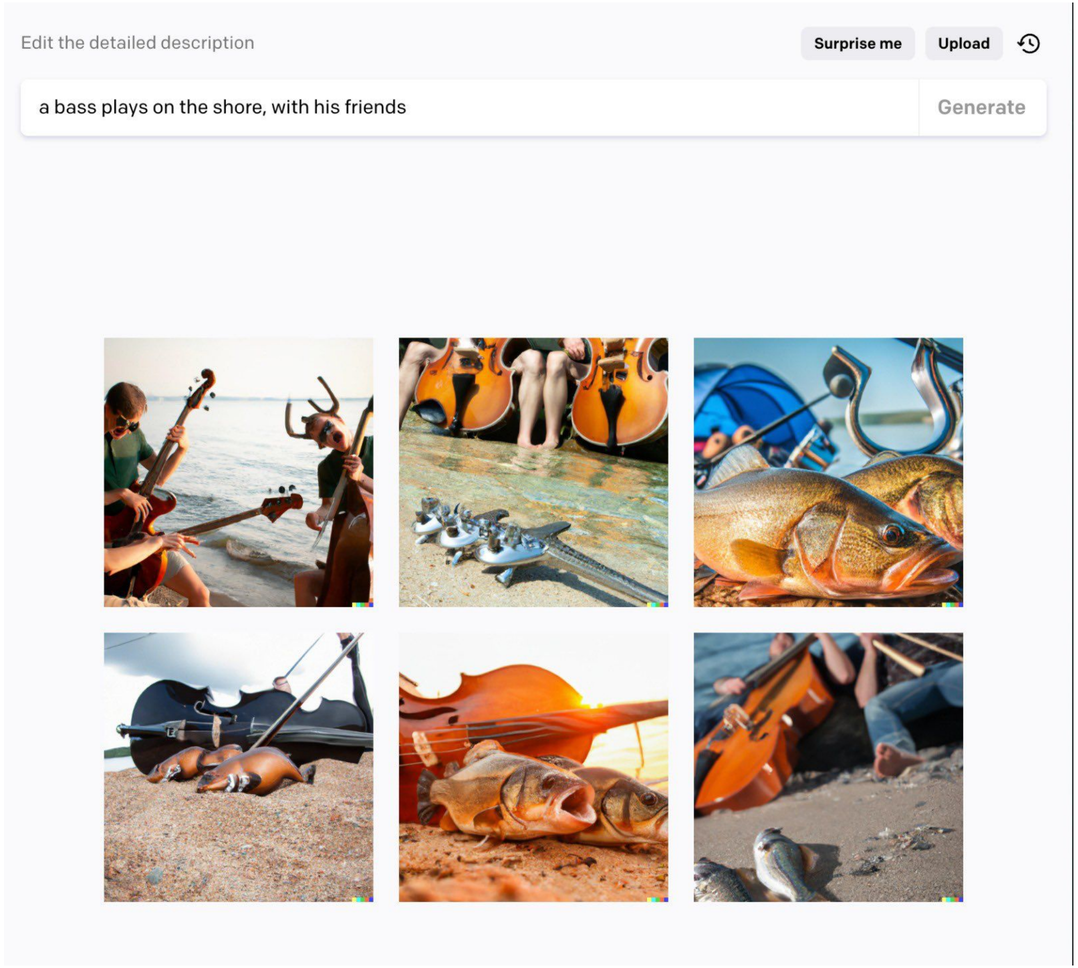
Natural language ambiguity

State of the art?



Royi Rassin
@RoyiRassin

Apparently DALL-E 2 couldn't pick the appropriate word-sense for "bass", and just settled on using both senses. I find it surprising #dalle2



a bass plays on the shore, with his friends

https://twitter.com/RoyiRassin/status/1547813162811924480?s=20&t=C2IN5OX-r-Ha8DOyRX2wVg

Deriving the linguistic specification

Now we have pronounceable words we want to derive:

Pronunciation:

- Pronunciation Dictionaries
- Letter-to-Sound (LTS) rules

Prosody:

- Prominence marking: pitch accents
- Break detection: pauses, boundary tones

Pronunciation Dictionaries

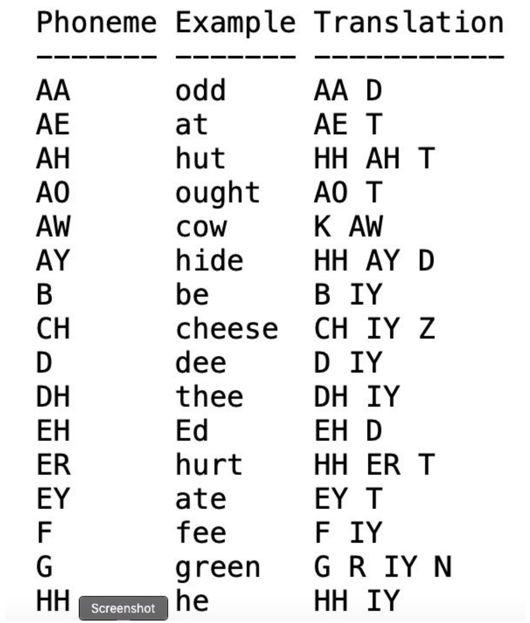
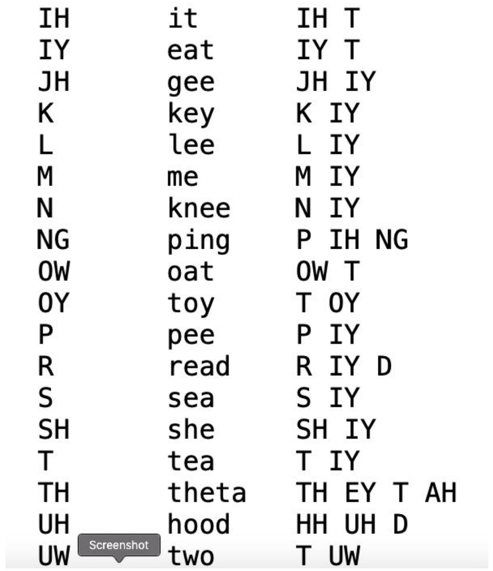
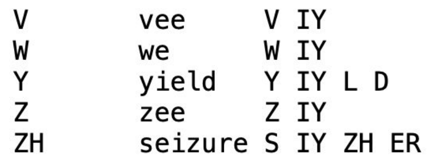
Use pre-existing pronunciation dictionaries to map words to phonetic transcriptions

But what phone set should we use?

- IPA has more symbols than necessary for many language specific tasks.
  - Not so efficient for many languages!
- Also, not as easily machine readable (until recently)
- We usually use simplified phone sets developed with speech tech in mind

CMUDict - ARPABET

The CMU Pronouncing Dictionary (US English) uses the ARPABET phoneset (39 **phones**)


Screenshot

Screenshot


Phoneme	Example	Translation
AA	odd	AA D
AE	at	AE T
AH	hut	HH AH T
AO	ought	AO T
AW	cow	K AW
AY	hide	HH AY D
B	be	B IY
CH	cheese	CH IY Z
D	dee	D IY
DH	thee	DH IY
EH	Ed	EH D
ER	hurt	HH ER T
EY	ate	EY T
F	fee	F IY
G	green	G R IY N
HH	he	HH IY
IH	it	IH T
IY	eat	IY T
JH	gee	JH IY
K	key	K IY
L	lee	L IY
M	me	M IY
N	knee	N IY
NG	ping	P IH NG
OW	oat	OW T
OY	toy	T OY
P	pee	P IY
R	read	R IY D
S	sea	S IY
SH	she	SH IY
T	tea	T IY
TH	theta	TH EY T AH
UH	hood	HH UH D
UW	two	T UW
V	vee	V IY
W	we	W IY
Y	yield	Y IY L D
Z	zee	Z IY
ZH	seizure	S IY ZH ER

http://www.speech.cs.cmu.edu/cgi-bin/cmudict (you can lookup pronunciations here for US English!)

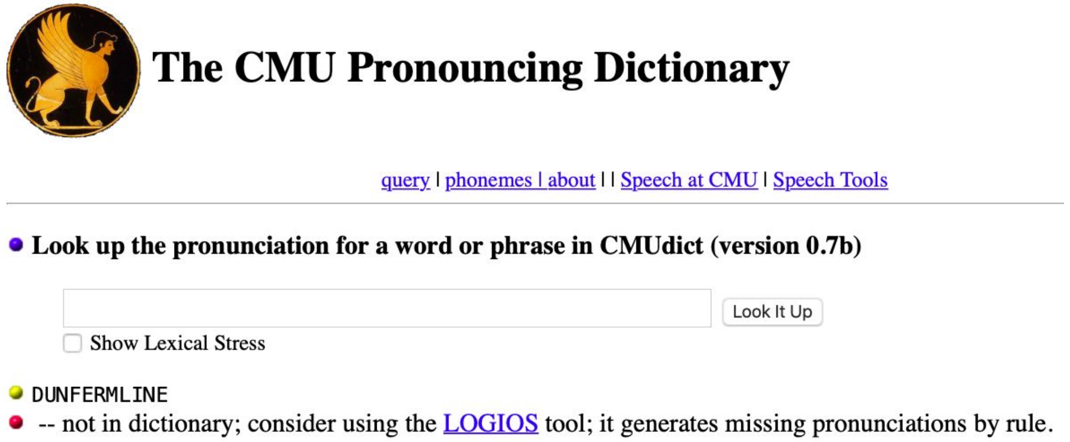
CMUDict - ARPABET

EDIFY	EH1 D AH0 F AY2
EDIFYING	EH1 D AH0 F AY2 IH0 NG
EDIGER	EH1 D IH0 G ER0
EDIN	EH1 D IH0 N
EDINA	AH0 D IY1 N AH0
EDINBORO	EH1 D AH0 N B ER0 OW0
EDINBURGH	**EH1 D AH0 N B ER0 OW0**
EDINGER	EH1 D IH0 NG ER0
EDINGTON	EH1 D IH0 NG T AH0 N
EDISON	EH1 D IH0 S AH0 N
EDISON'S	EH1 D IH0 S AH0 N Z
EDISONS	EH1 D IH0 S AH0 N Z
EDISTO	EH1 D IH0 S T OW0
EDIT	EH1 D AH0 T
EDITED	EH1 D AH0 T AH0 D
EDITED(1)	EH1 D IH0 T IH0 D

1 big text file of words and their pronunciations

CMUDict

- CMUDict is dialect specific
- What do we do if we want to pronounce a word that's not in the lexicon?
- Can we make a lexicon with greater generalizability?



The CMU Pronouncing Dictionary

query | phonemes | about | | Speech at CMU | Speech Tools

Look up the pronunciation for a word or phrase in CMUdict (version 0.7b)

DUNFERMLINE

-- not in dictionary; consider using the LOGIOS tool; it generates missing pronunciations by rule.

Unilex

Unilex is an ‘accent-independent’ lexicon based on the Unisyn database

- Classifies phones by keywords e.g. ‘Foot’ vs ‘Strut’ are keywords
    - ‘Put’ → FOOT class
    - ‘Putt’ → STRUT class
- Use this to describe phonemic variation in English dialects/accents
- A single lexicon to encode different accents: run lexicon through accent specific rules to produce accent specific lexica

Unilex - Edinburgh English (edi)

```
("edify" vb (((e) 1) ((d i) 0) ((f ae) 0)))
("edify" vbp (((e) 1) ((d i) 0) ((f ae) 0)))
("edifying" jj (((e) 1) ((d i) 0) ((f ae) 2) ((i n) 0)))
("edifying" vbg (((e) 1) ((d i) 0) ((f ae) 2) ((i n) 0)))
("edifyingly" rb (((e) 1) ((d i) 0) ((f ae) 2) ((i n) 0) ((l ii) 0)))
```

("edinburgh" nnp (((e) 1) ((d i m) 0) ((b r @) 0)))

```
("edinburgh's" nnp|pos (((e) 1) ((d i m) 0) ((b r @ z) 0)))
("edinburgh's" nnp|vbz (((e) 1) ((d i m) 0) ((b r @ z) 0)))
("edison" nnp (((e) 1) ((d i) 0) ((s n!) 0)))
("edit" nn (((e) 1) ((d i ?) 0)))
("edit" vb (((e) 1) ((d i ?) 0)))
("edit" vbp (((e) 1) ((d i ?) 0)))
("editability" nn (((e) 2) ((d i ?) 0) ((@) 0) ((b i l) 1) ((@) 0) ((? ii) 0)))
("editable" jj (((e) 1) ((d i ?) 0) ((@) 0) ((b l!) 0)))
("edited" vbd (((e) 1) ((d i ?) 0) ((i d) 0)))
```

Unilex - General American English (gam)

```
("edify" vb (((e) 1) ((t^ @) 0) ((f ai) 0)))
("edify" vbp (((e) 1) ((t^ @) 0) ((f ai) 0)))
("edifying" jj (((e) 1) ((t^ @) 0) ((f ai) 2) ((i ng) 0)))
("edifying" vbg (((e) 1) ((t^ @) 0) ((f ai) 2) ((i ng) 0)))
("edifyingly" rb (((e) 1) ((t^ @) 0) ((f ai) 2) ((i ng) 0) ((lw ii) 0)))
("edinburgh" nnp (((e) 1) ((t^ i m) 0) ((b @@r) 2) ((r ou) 0)))
("edinburgh's" nnp|pos (((e) 1) ((t^ i m) 0) ((b @@r) 2) ((r ou z) 0)))
("edinburgh's" nnp|vbz (((e) 1) ((t^ i m) 0) ((b @@r) 2) ((r ou z) 0)))
("edison" nnp (((e) 1) ((t^ i) 0) ((s n!) 0)))
("edit" nn (((e) 1) ((t^ @ t) 0)))
("edit" vb (((e) 1) ((t^ @ t) 0)))
("edit" vbp (((e) 1) ((t^ @ t) 0)))
("editability" nn (((e) 2) ((t^ @ t^) 0) ((@) 0) ((b i lw) 1) ((@) 0) ((t^ ii) 0)))
("editable" jj (((e) 1) ((t^ @ t^) 0) ((@) 0) ((b l!) 0)))
("edited" vbd (((e) 1) ((t^ @ t^) 0) ((@ d) 0)))
("edited" vbn (((e) 1) ((t^ @ t^) 0) ((@ d) 0)))
```

Dialect variation

How many different vowels in your own English dialect?

“Mary, marry me! Make me Merry!”

- Mary = marry?
- Marry = Merry?
- Mary = Merry?

Different phoneset for different tasks

- Edinburgh
- IPA: edɪmbrə
- ARPABET: EH1 D AH0 N B R AX0
- Unilex-edi: e d i m b r @
- Unilex-edi full: (((e)1) ((d i m)0) ((b r @)0))

Having a compact lexicon and rules to generate variants seems like a good idea - but how to we figure out the rules?

We need some phonemic analysis...

Phonemes

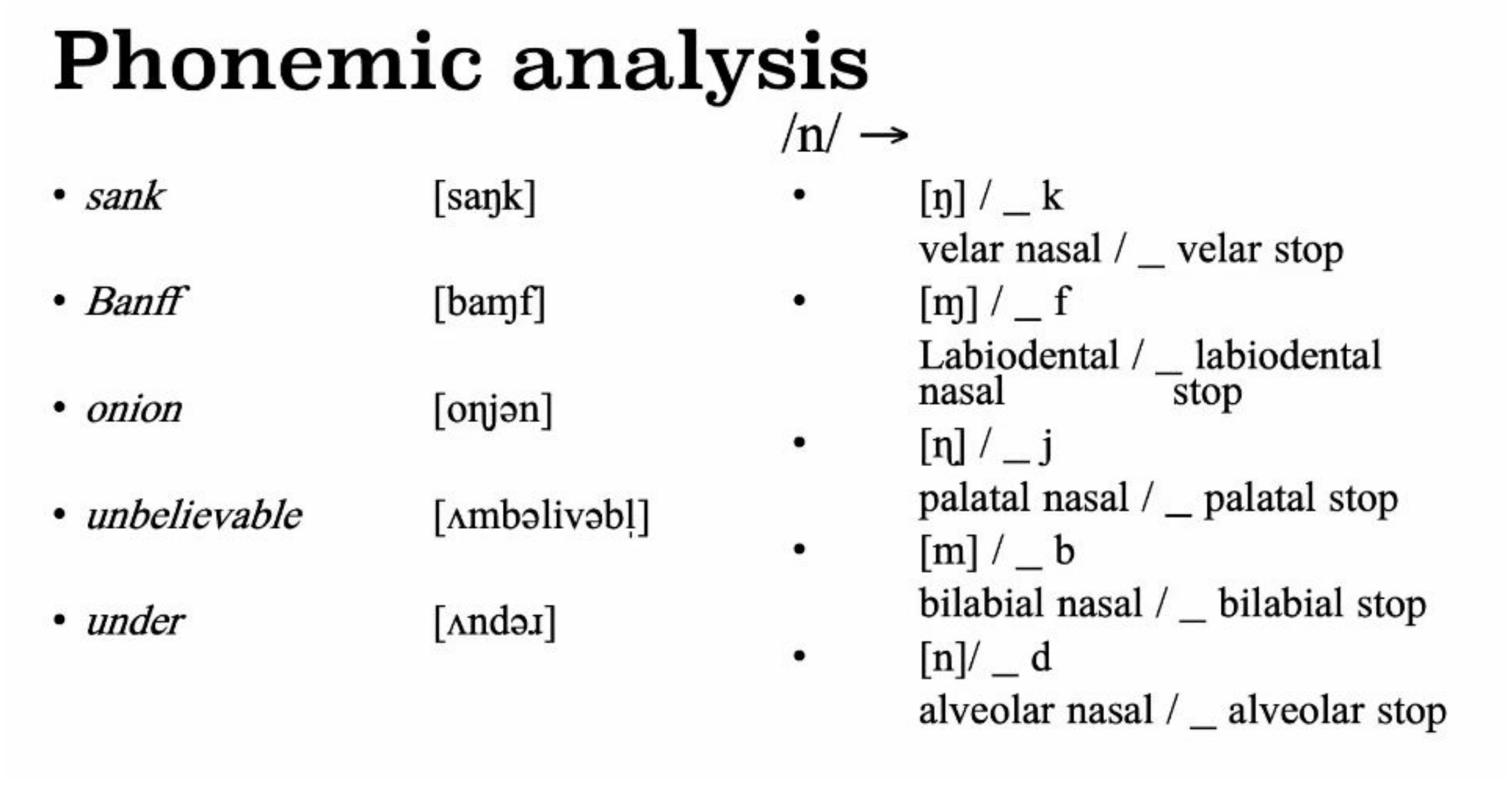
Phonemes are abstract categories over actual speech sounds

- Contrastive units
- Distinguished by minimal pairs of words
  - e.g. unaspirated/aspirated stops aren’t distinguished in English: [pot] vs [p^{h}ot]
  - e.g. unvoice/voice strops are: [pot] vs [bot]

Allophones

Allophones: the realization of a phoneme as an actual speech sound

- Different allophones are perceived as the same phoneme
- Predictable phonetic variation determined by context



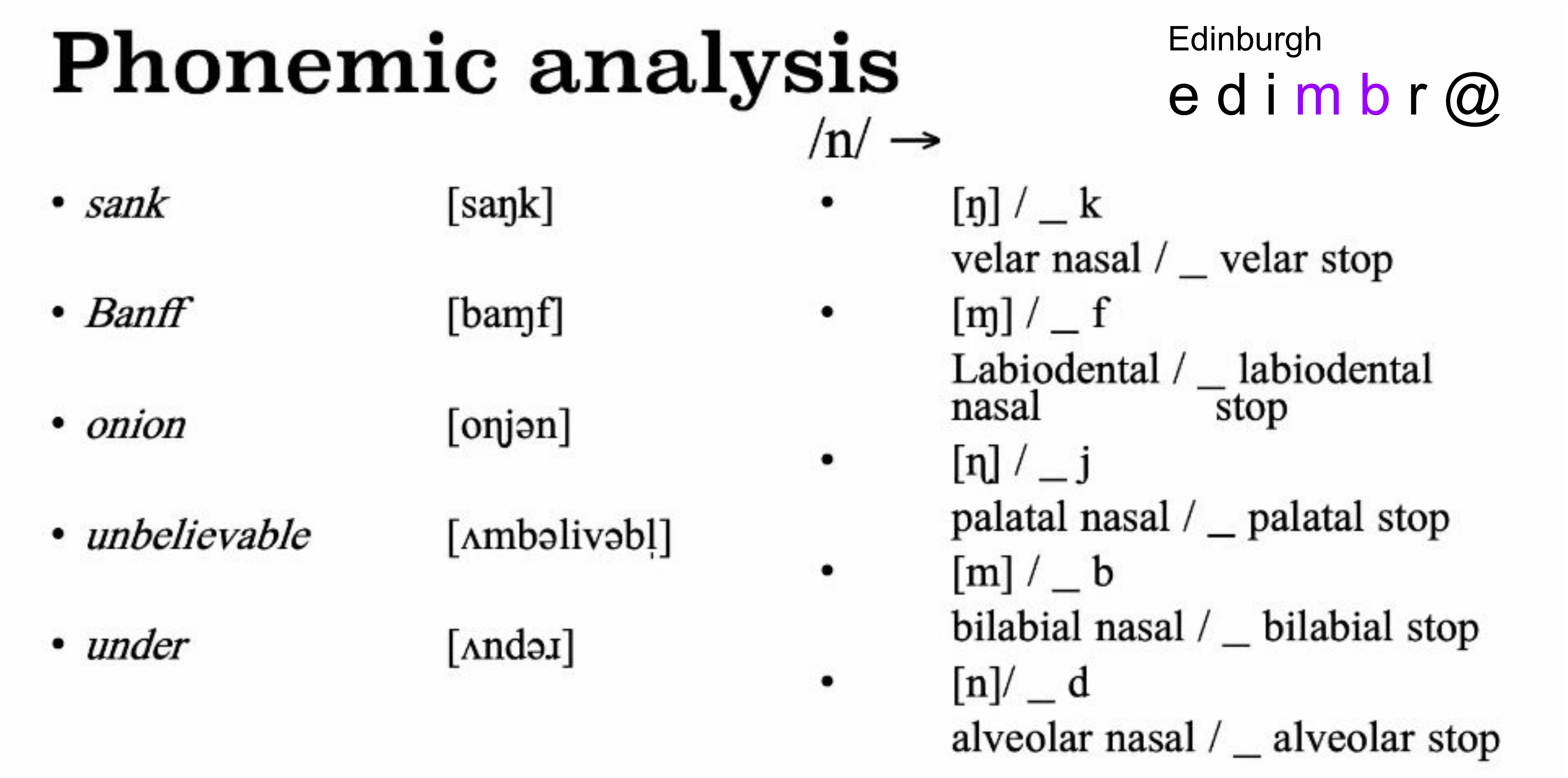
Phonemic analysis

• *sank*	[saŋk]
• *Banff*	[baɱf]
• *onion*	[oɲjən]
• *unbelievable*	[ʌmbəlivəbl̩]
• *under*	[ʌndəɹ]

/n/ →

- [ŋ] / _ k
  velar nasal / _ velar stop
- [ɱ] / _ f
  Labiodental nasal / _ labiodental stop
- [ɲ] / _ j
  palatal nasal / _ palatal stop
- [m] / _ b
  bilabial nasal / _ bilabial stop
- [n]/ _ d
  alveolar nasal / _ alveolar stop

From module 5 video: Phonemes and Allophones (R. Puderbaugh)



Phonemic analysis

Edinburgh
edimbr@

/n/ →

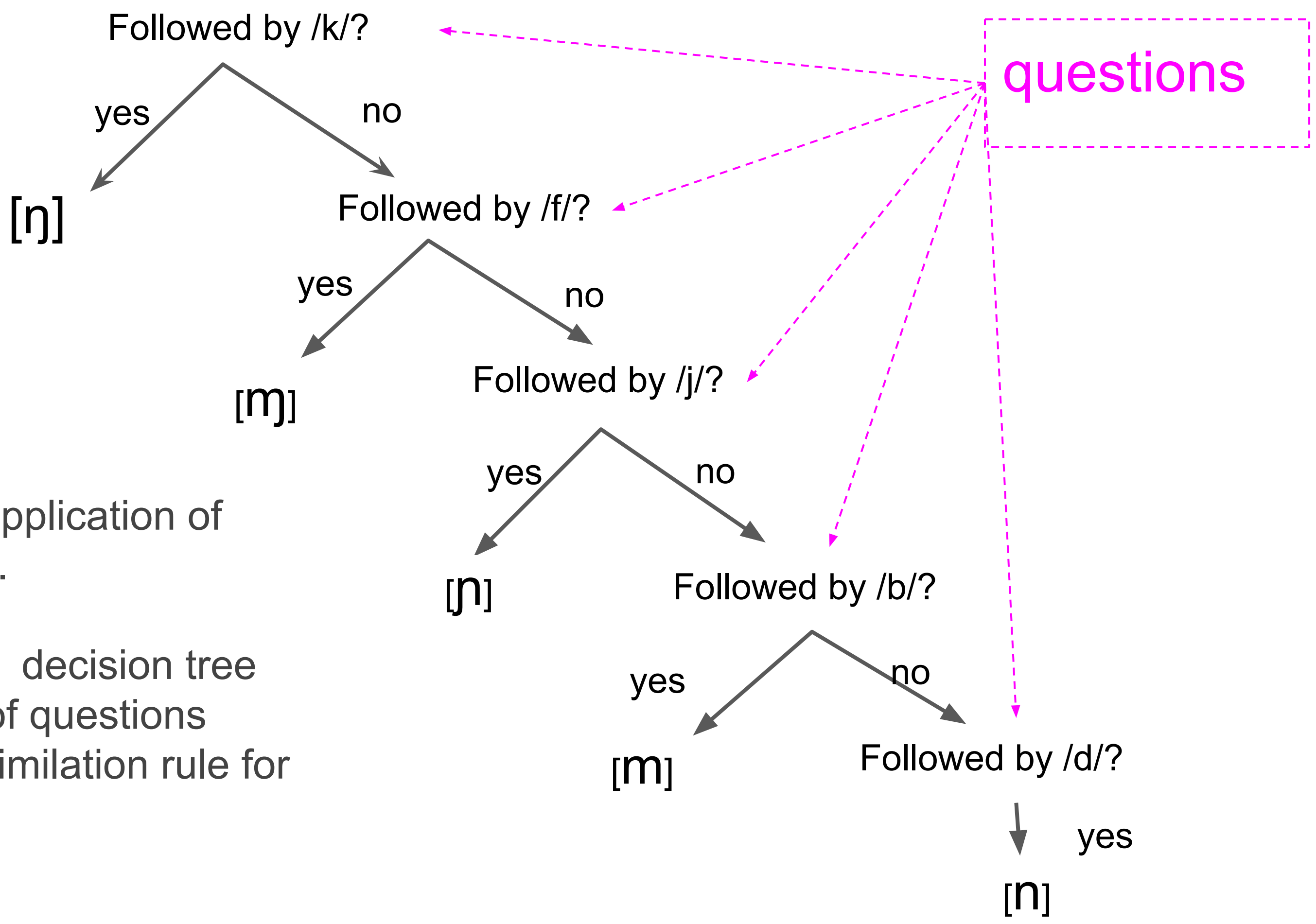
• *sank*	[saŋk]
• *Banff*	[baɱf]
• *onion*	[oɲjən]
• *unbelievable*	[ʌmbəlivəbl̩]
• *under*	[ʌndəɹ]

- [ŋ] / _ k
  velar nasal / _ velar stop
- [ɱ] / _ f
  Labiodental nasal / _ labiodental stop
- [ɲ] / _ j
  palatal nasal / _ palatal stop
- [m] / _ b
  bilabial nasal / _ bilabial stop
- [n]/ _ d
  alveolar nasal / _ alveolar stop

/n/ is realized with the **place of articulation** of the following consonant

Rules and Decision Trees

Decision Trees



We can represent the application of rules as a decision tree.

Here's a a hand-written decision tree with arbitrary ordering of questions based on the place assimilation rule for pronunciation of /n/

Learning letter to sound rules

(Morpho-)phonological analysis is hard! Can we learn letter to sound rules directly from the data?

Yes! Using machine learning methods

Learning decision trees

- Derive the questions from observed data
- Use counts over observed data to decide which questions to ask first

Counts of /n/ pronunciation in different contexts

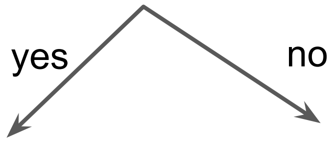
Following Context	pronunciation	count
b ("Edinburgh")	[m]	3
b (not "Edinburgh")	[n]	477
d	[n]	6138
f	[n]	1273

Following Context	pronunciation	count
io	[ny]	197
i (not o)	[n]	4537
(no boundary) k	[ng]	1338
(boundary) k	[n]	56

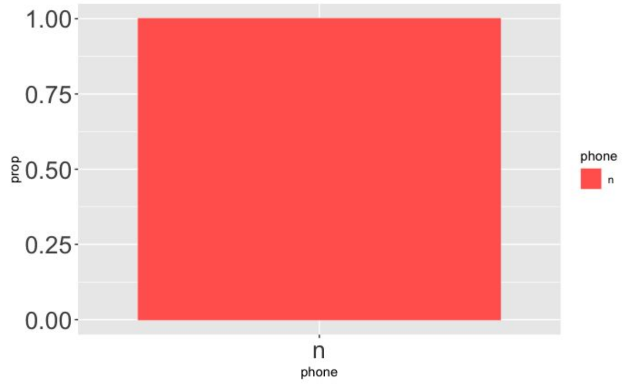
Based, but not exactly, on unilex-edi

Decision Trees

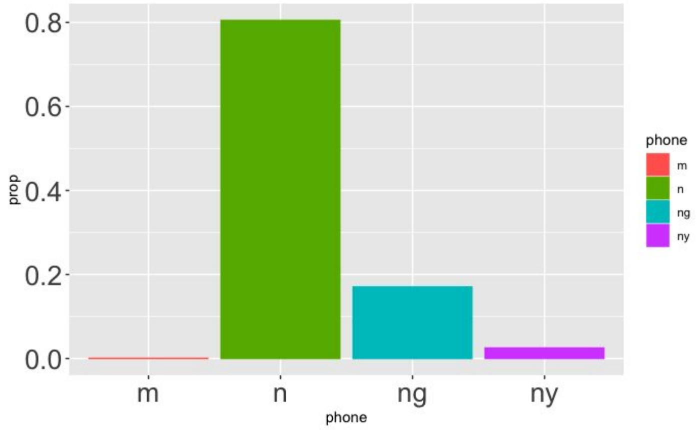
Followed by “d”?



Separate the data depending on the answer

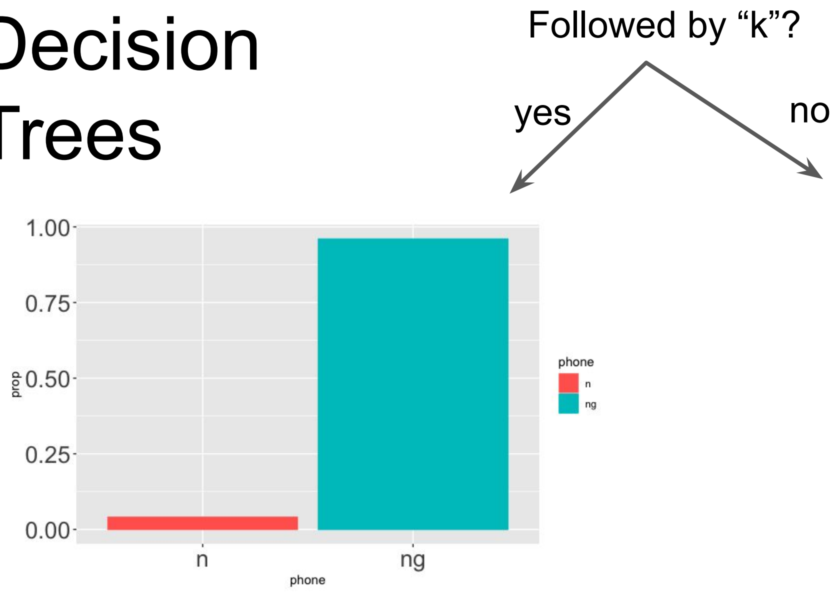


before “d”, “n” is pronounced [n]

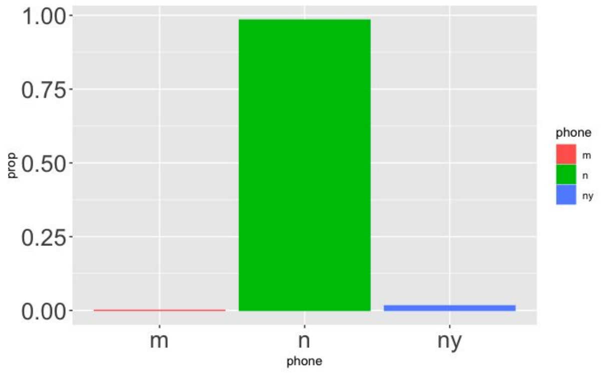


In other contexts, there’s still a range of potential pronunciations

Decision Trees



Before "k", "n" is almost always pronounced ng=[ŋ]

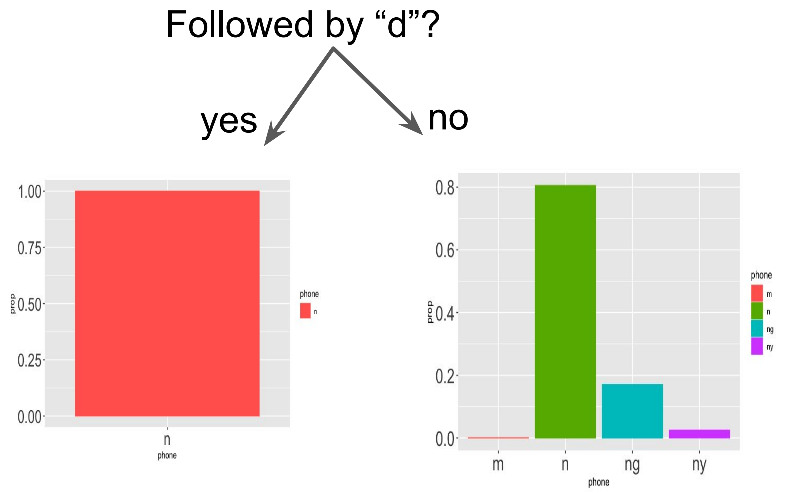


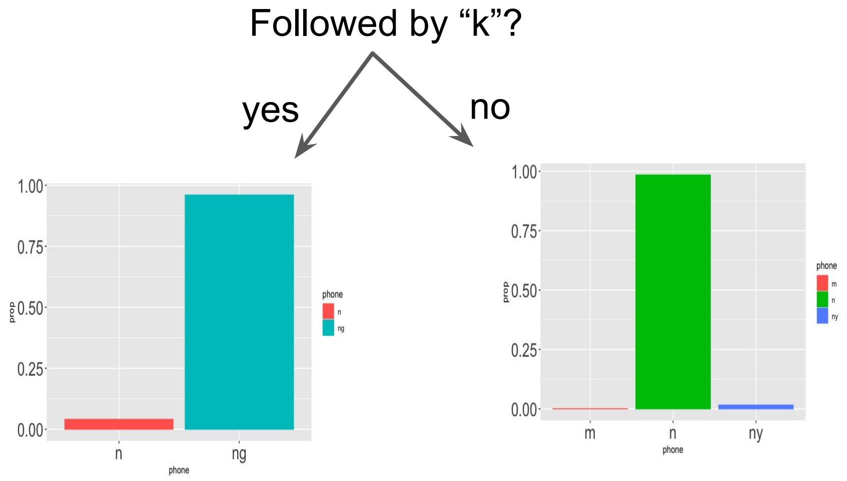
In other contexts, "n" is almost always pronounced n=[n]

Question: Does asking this question first reduce our uncertainty about the pronunciation more than the "d" question?

Fairly certain what to do if the answer is "yes" in both cases….

Reduction of uncertainty

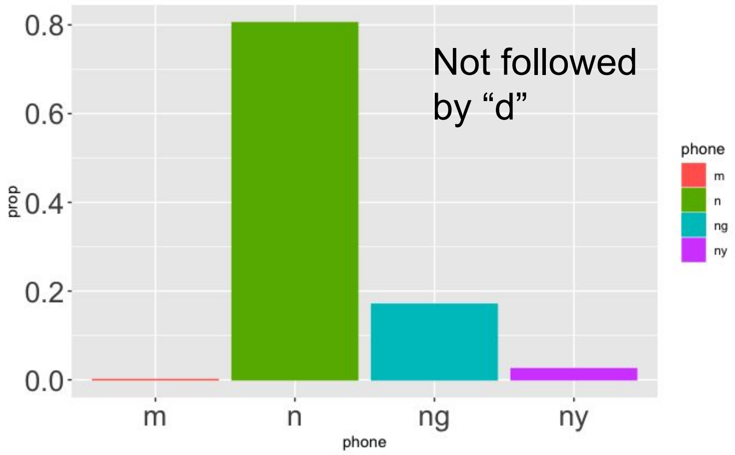




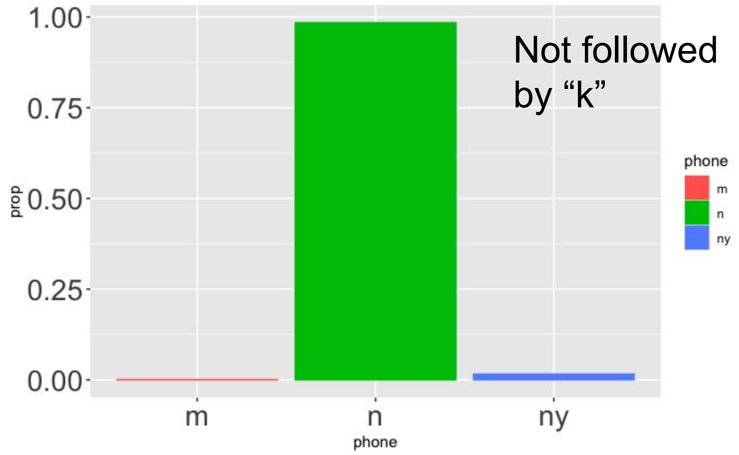
Question: Does asking th "k" question reduce our uncertainty about the pronunciation more than the "d" question?

Let's focus on the "no" answers for a minute…

Reduction of uncertainty



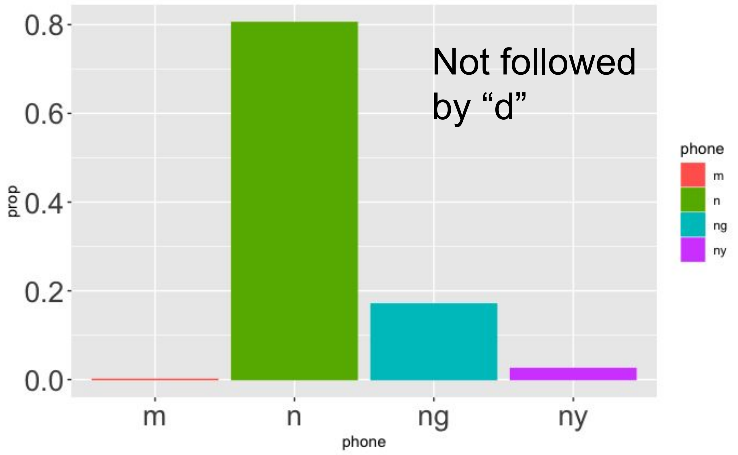
Less concentrated on [n]



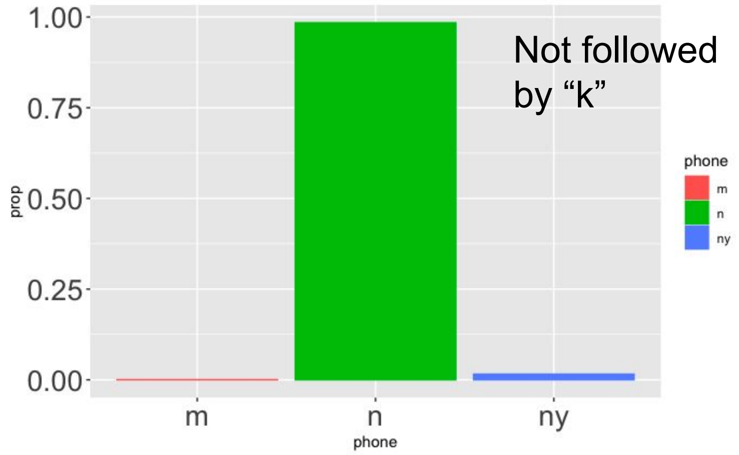
More concentrated on [n]

A distribution where the probability mass is highly concentrated on a single option indicates less uncertainty than one where many options are likely

Entropy: a measure of uncertainty



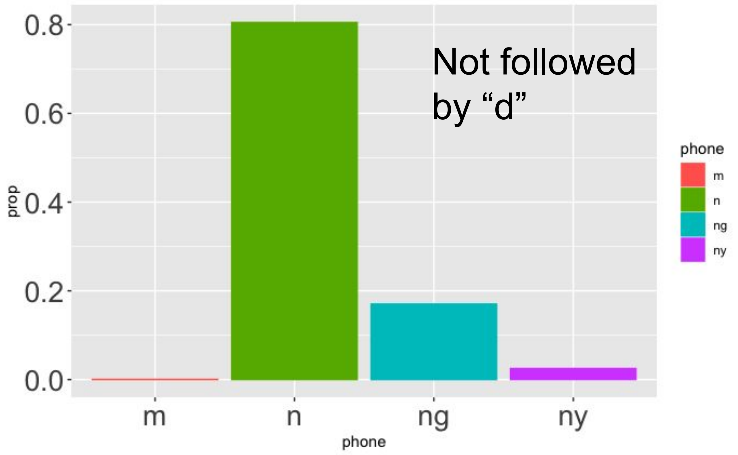
Less concentrated on [n]



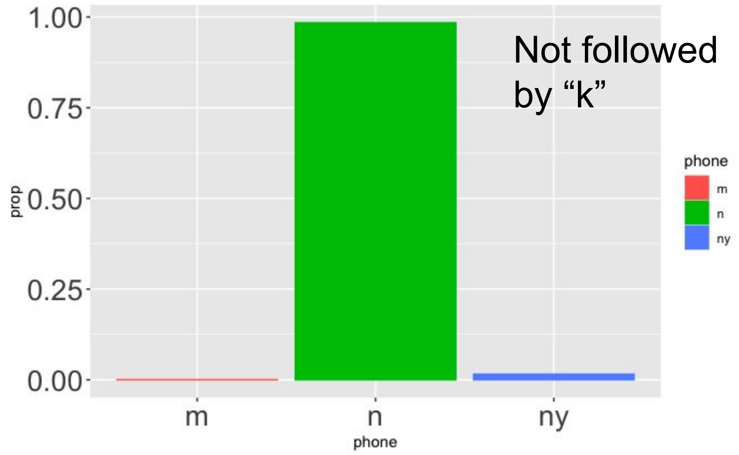
More concentrated on [n]

Entropy is highest when the probabilities over the possible events are equal, i.e. when every option is equally surprising

Entropy as a measure of uncertainty



Less concentrated on [n]

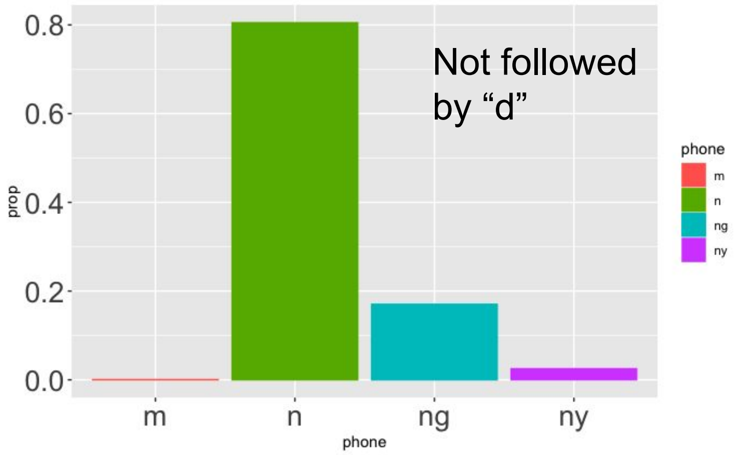


More concentrated on [n]

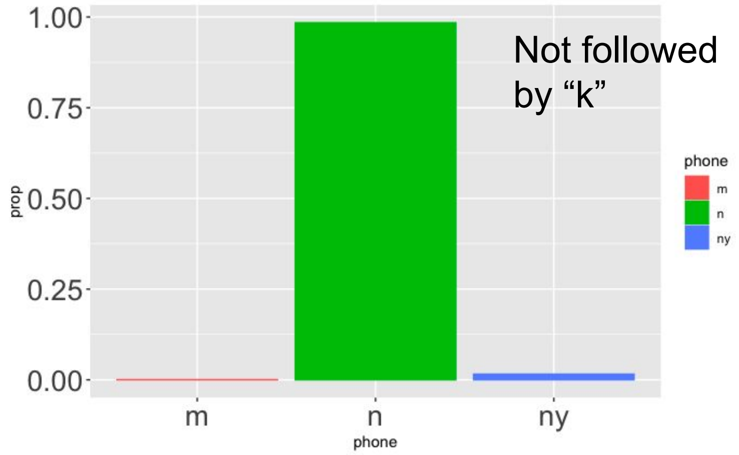
Conversely, entropy is lowest when probability is concentrated on just one event → if there's only 1 choice, there's no surprise when we see it

See also: extra lab notebook `tts/tts-1-1-entropy`

Entropy as a measure of uncertainty



Less concentrated on [n]



More concentrated on [n]

Calculate Entropy using probabilities over the set of potential options:

$$H(X) = \sum_{i=1}^{n} P(x_i) \log P(x_i)$$

Entropy as expected surprisal

Number of potential events

“Surprisal” or “information content” of the event
→ low probability events are more surprising

$$H(X) = -\boxed{\sum_{i=1}^{n}} P(x_i) \boxed{\log P(x_i)}$$

Entropy: expected suprisal of the probability distribution

Sum over possible events

Probability of event $\mathbf{x}_i$

Also more videos on this on speech.zone!

https://speech.zone/entropy-understanding-the-equation/

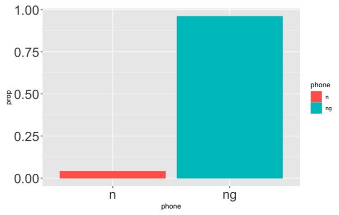
Decision Trees

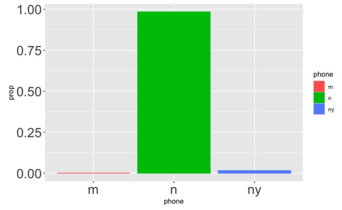
Choose the question that reduces uncertainty → Choose the split that produces the greatest information gain

Followed by “k”?

yes

no





Preceded by syl boundary?

yes

no

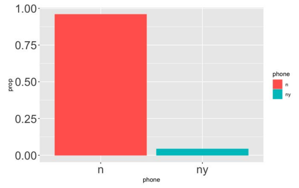
[n]

[ng]

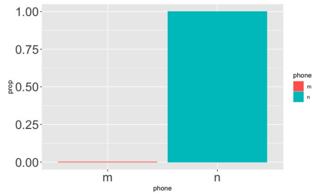
Followed by “i”?

yes

no



Mostly [n], a few [ny]’s

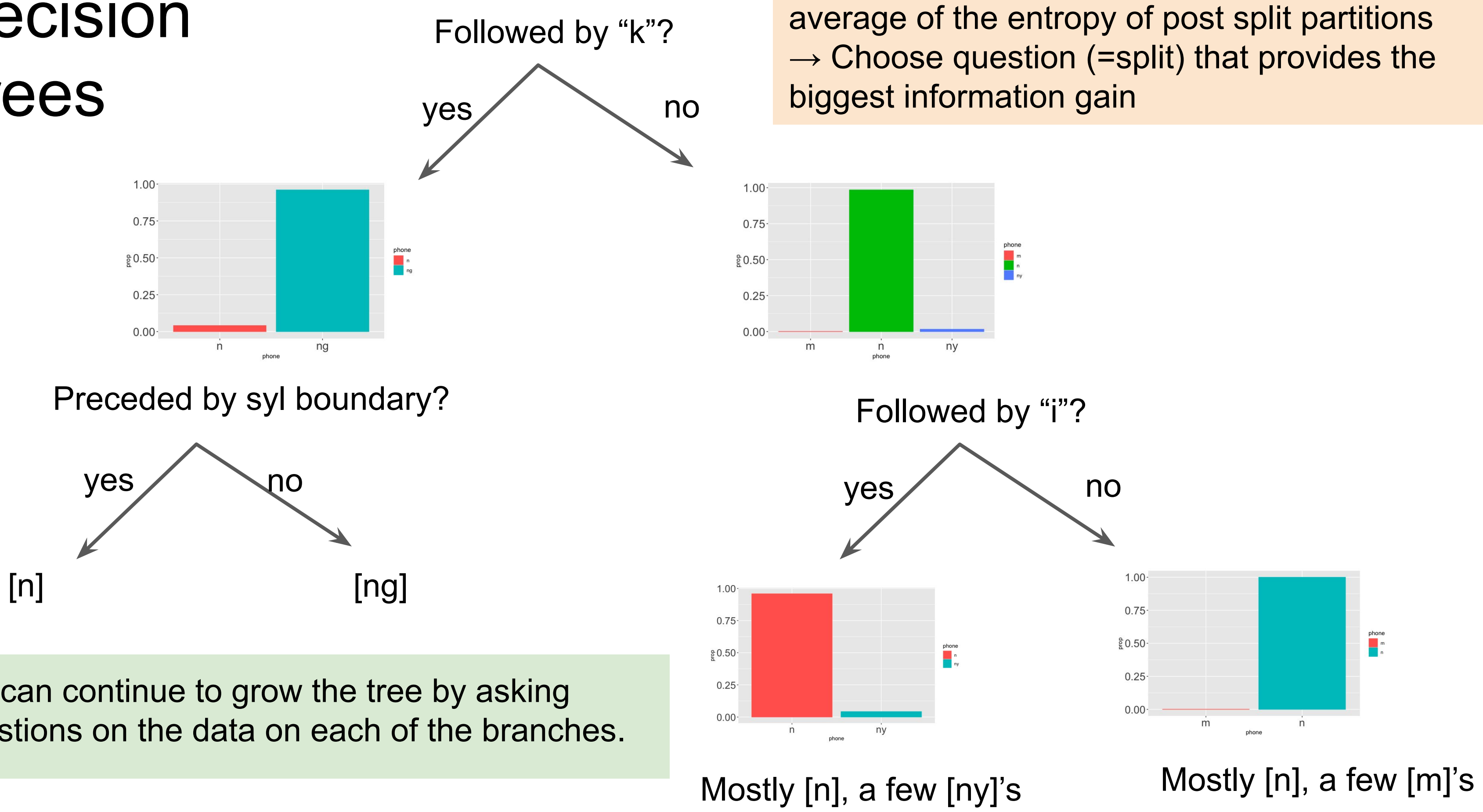


Mostly [n], a few [m]’s

We can continue to grow the tree by asking questions on the data on each of the branches.

Decision Trees

Information gain = entropy before split - weighted average of the entropy of post split partitions
→ Choose question (=split) that provides the biggest information gain



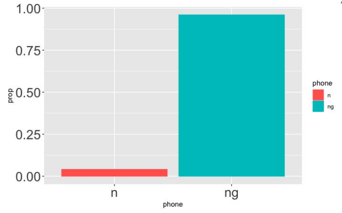
We can continue to grow the tree by asking questions on the data on each of the branches.

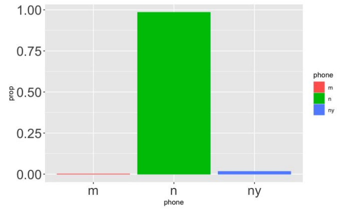
Decision Trees

Followed by “k”?

yes

no





Preceded by syl boundary?

yes

no

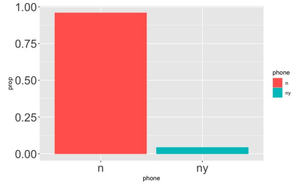
[n]

[ng]

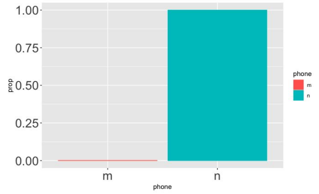
Followed by “i”?

yes

no



Mostly [n], a few [ny]’s



Mostly [n], a few [m]’s

Continue until there’s not enough data at each “leaf” to estimate probabilities or entropy is below some threshold (or other stopping rules).

TTS Pronunciation

Options for mapping written words to pronunciations (phones):

- Lookup phonetic transcript in a pronunciation dictionary (i.e., lexicon)
- Use hand-written letter-to-sound rules - requires expert knowledge
- Learn rules from the data - requires machine learning methods, e.g. decision trees (state of the art: neural network models)

Deriving the linguistic specification

Pronunciation:

- Pronunciation Dictionaries
- Letter-to-Sound (LTS) rules

Prosody:

- Prominence marking: pitch accents
- Break detection: pauses, boundary tones

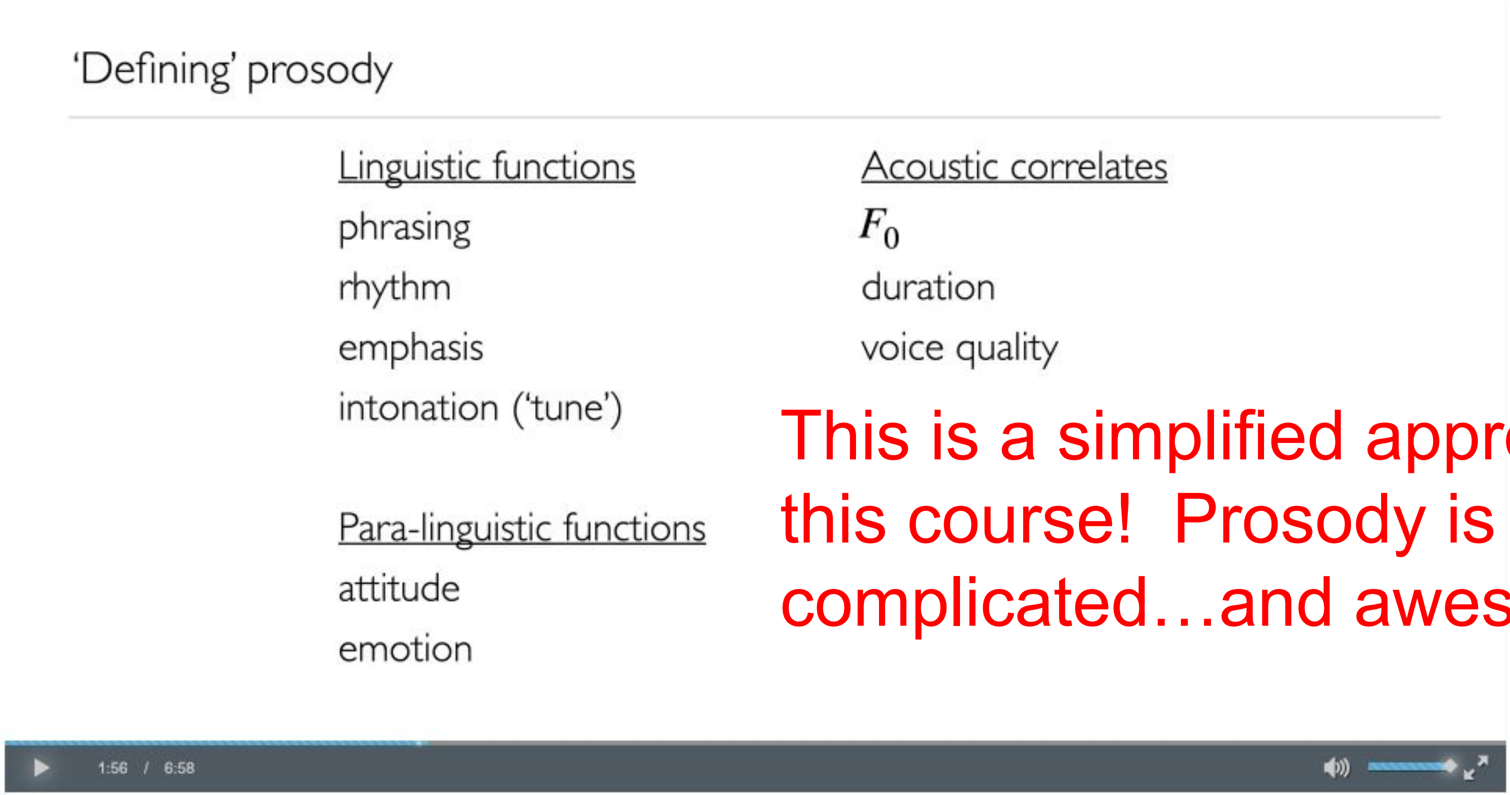
After determining the correct phone sequence, we also need to get the non-lexical characteristics of speech right, i.e. prosody.

Prosody!

Aspects of Prosody

- Pitch : Fundamental frequency (F0)
  - Is it high or low?
- Loudness: Derived from wave amplitude, e.g. Intensity
  - Loud or soft? How much energy?
- Timing: segment durations, pauses
  - Short of long?
- Voice quality: creak, breath
  - What’s happening with your glottis?

Prosody



‘Defining’ prosody

Linguistic functions	Acoustic correlates
phrasing	F_0
rhythm	duration
emphasis	voice quality
intonation (‘tune’)	

Para-linguistic functions
attitude
emotion

1:56 / 6:58

This is a simplified approach for this course! Prosody is complicated…and awesome!

Varying Prosodic Prominence

Same sentence, different prosody:

1. Emily **DID** bring a meringue
2. Emily **DID** bring a **MERINGUE**
3. Emily **DID** bring a meringue

Different emphasis falls on different words with emphasized words more prosodically prominent than others: relatively higher pitch excursions, energy and duration.

Varying Prosodic Prominence

Same sentence, different prosody:

1. Emily **DID** bring a meringue.
2. Emily **DID** bring a ***MERINGUE...***
3. Emily **DID** bring a meringue?

But prominences can sound different:

Different pitch accents and boundary tones

Boundaries

I said to him when you left do you remember I told you
I said to him don’t forget Dave if you ever get in
trouble give us a call you never know your luck

Boundaries - Prosodic breaks

```
I said to him when you left do you remember I told you
I said to him don't forget Dave if you ever get in
trouble give us a call you never know your luck
```

pause

Lowered pitch register

I said to him when you left, (do you remember? I told you) I said to him: "Don't forget Dave if you ever get in trouble give us a call. You never know your luck."

Lowered pitch register

pause

Speech is "punctuated" by prosodic breaks: most obvious perceived as pauses, but also resets in pitch or changes in pitch register.

(A bit of) Intonational Phonology

Strict Layer Hypothesis

[Selkirk 1984]

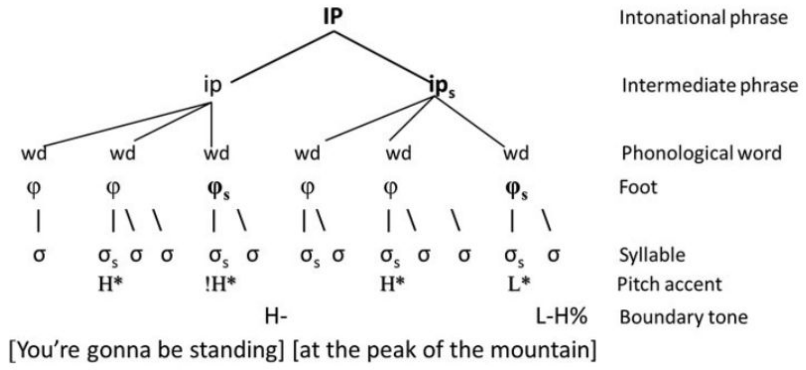
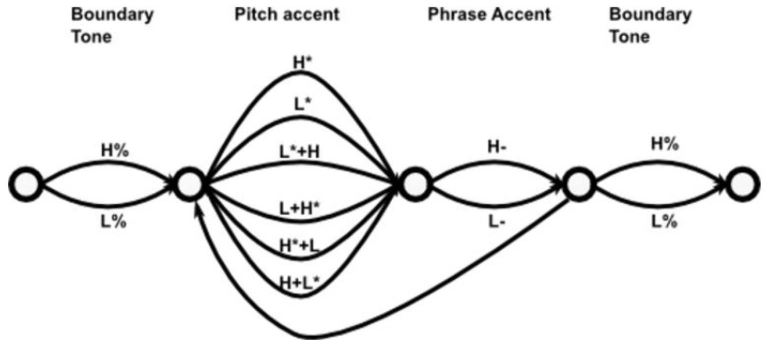


Figure from [Cole 2015]

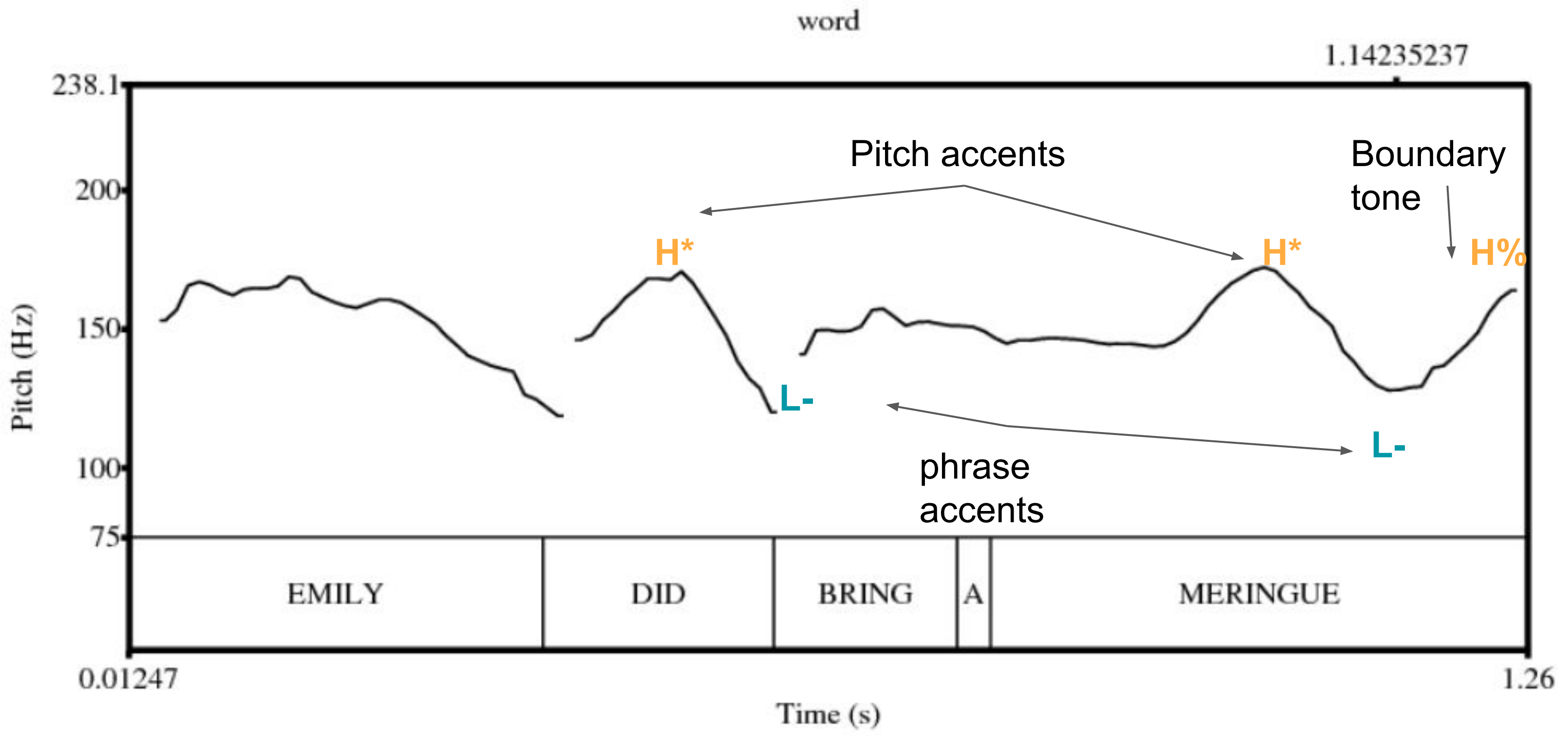
Autosegmental-Metrical Theory of English Phonology

[Pierrehumbert 1980, Beckman & Pierrehumbert 1988]



TTS systems usually assume a simplified version of intonational phonology: just labelling of prominences (pitch accents) and boundaries: no higher structure (yet!)

Prosodic Labelling: ToBI



Prosody prediction

Predict prominences and boundaries from the text: attach pitch accent and boundary tones, and break labels (NB=no break, BB=break) to words

Tones				L*+H H-H%
Breaks	NB	NB	NB	BB
Words	I	live	in	Edinburgh
Phones+Syllables	((ai)1)	((liv)1)	((in)1)	((e)1)((dim)0)((br@)0)

“I live in Edinburgh?”

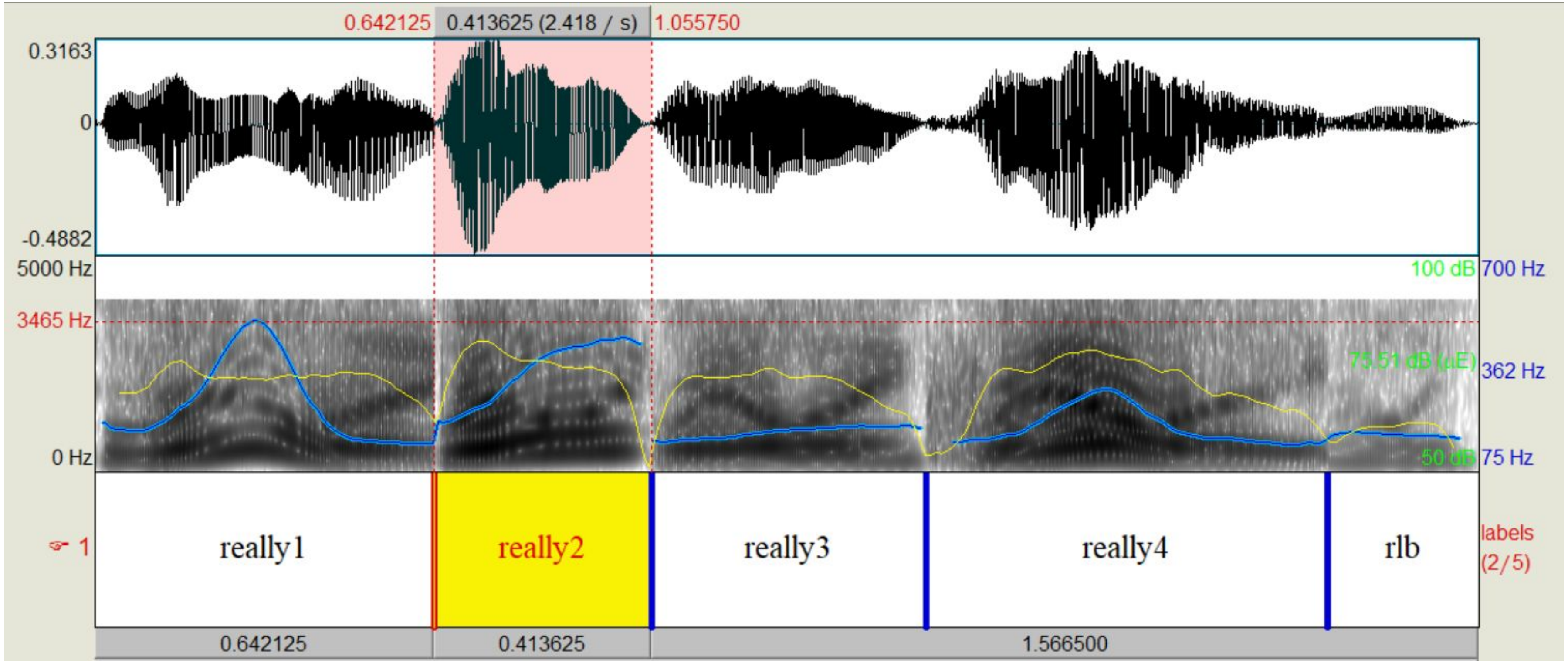
Prosody prediction

Predict prominences and boundaries from the text: attach pitch accent and break labels to words

- Simple rule based pitch accent and break labelling:
  - e.g. prominence on content words
  - e.g. breaks (i.e. pauses) at punctuation
- Learned rules:
  - use linguistic features to predict prosodic labels e.g. with a decision tree
- Probabilistic/machine learning:
  - learn probabilities of prosodic features text + speech, from a lot of data

Prosody: Really?

You can't predict all prosody just from text!



Examples from the Switchboard Corpus [Godfrey et al. 1996]

Front-end summary

Convert raw text to **pronounceable** words

- Tokenization
- Non-standard words, e.g. numbers, acronyms
- Word sense disambiguation, e.g. POS tagging

Map words to phonetic transcriptions, i.e. **phones**

- Pronunciation dictionary
- Letter to Sound rules

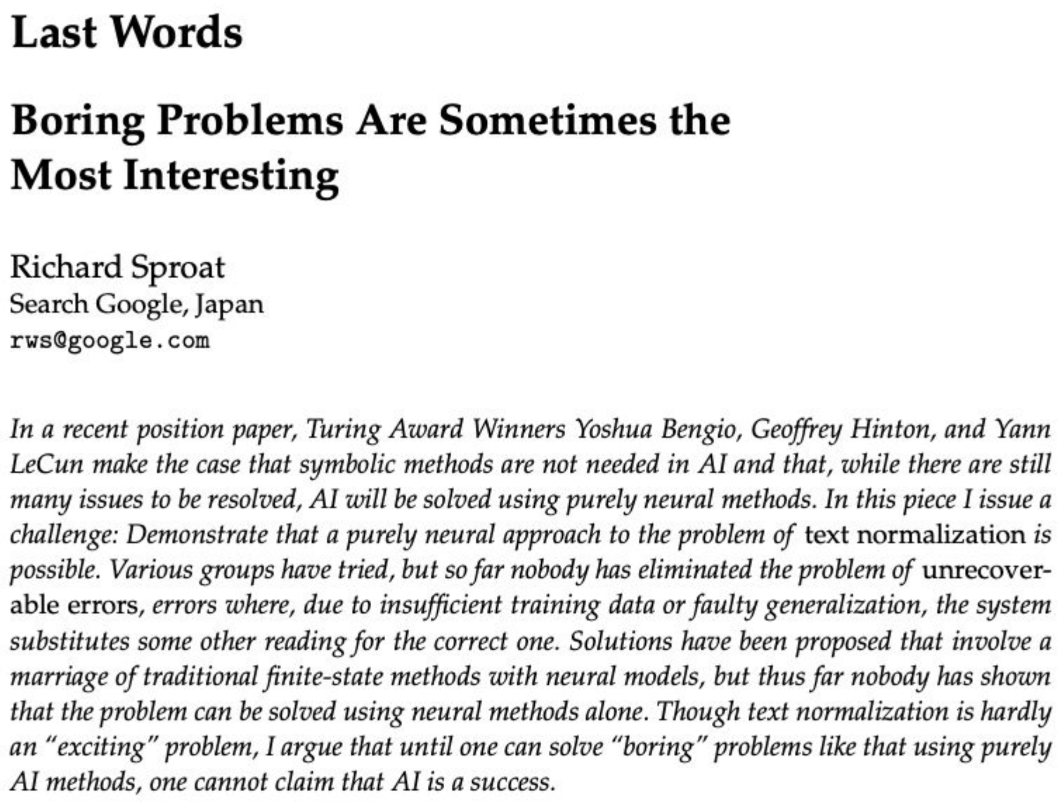
Add **prosodic** information

- Prominences: Pitch accents
- Boundaries: Breaks, boundary tones

To generate speech from text we must first analyze the text.

We derive a **linguistic specification** from which we can generate a waveform

Text normalization isn't solved!!



Last Words

Boring Problems Are Sometimes the Most Interesting

Richard Sproat
Search Google, Japan
rws@google.com

In a recent position paper, Turing Award Winners Yoshua Bengio, Geoffrey Hinton, and Yann LeCun make the case that symbolic methods are not needed in AI and that, while there are still many issues to be resolved, AI will be solved using purely neural methods. In this piece I issue a challenge: Demonstrate that a purely neural approach to the problem of text normalization *is possible. Various groups have tried, but so far nobody has eliminated the problem of* unrecoverable errors, *errors where, due to insufficient training data or faulty generalization, the system substitutes some other reading for the correct one. Solutions have been proposed that involve a marriage of traditional finite-state methods with neural models, but thus far nobody has shown that the problem can be solved using neural methods alone. Though text normalization is hardly an "exciting" problem, I argue that until one can solve "boring" problems like that using purely AI methods, one cannot claim that AI is a success.*

Sproat (2022) in Computational Linguistics, https://doi.org/10.1162/coli_a_00439

Morphology still helps in neural TTS (state of the art)

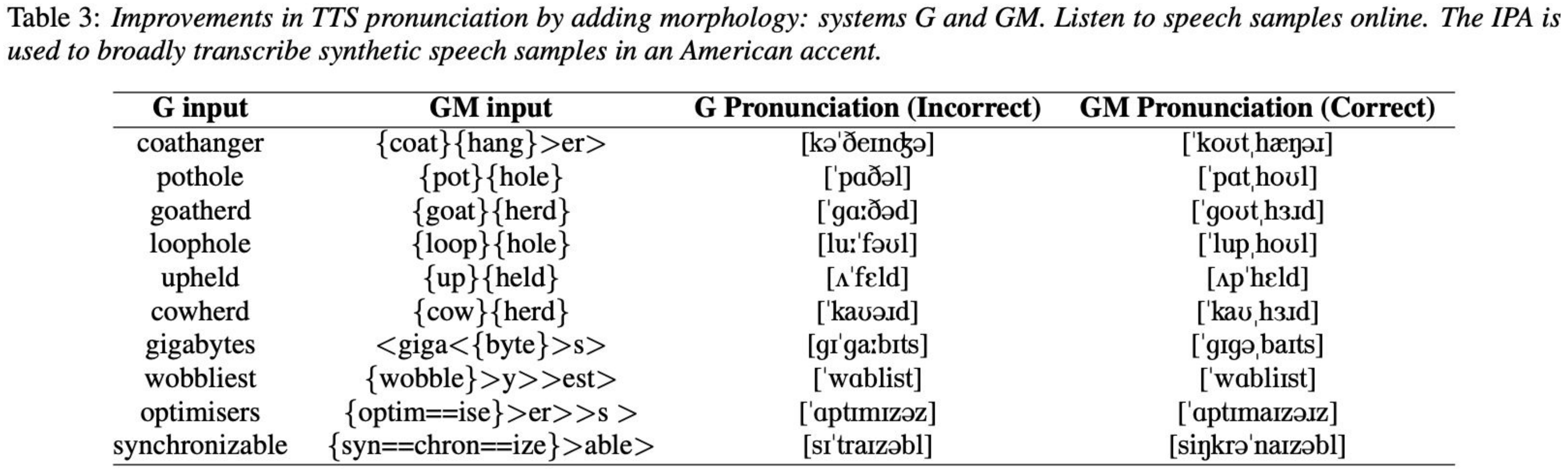


Table 3: *Improvements in TTS pronunciation by adding morphology: systems G and GM. Listen to speech samples online. The IPA is used to broadly transcribe synthetic speech samples in an American accent.*

G input	GM input	G Pronunciation (Incorrect)	GM Pronunciation (Correct)
coathanger	{coat}{hang}>er>	[kəˈðeɪnʤə]	[ˈkoʊtˌhæŋəɹ]
pothole	{pot}{hole}	[ˈpɑðəl]	[ˈpɑtˌhoʊl]
goatherd	{goat}{herd}	[ˈgɑːðəd]	[ˈgoʊtˌhɜɹd]
loophole	{loop}{hole}	[luːˈfəʊl]	[ˈlupˌhoʊl]
upheld	{up}{held}	[ʌˈfɛld]	[ʌpˈhɛld]
cowherd	{cow}{herd}	[ˈkaʊəɹd]	[ˈkaʊˌhɜɹd]
gigabytes	<giga<{byte}>s>	[gɪˈgaːbɪts]	[ˈgɪgəˌbaɪts]
wobbliest	{wobble}>y>>est>	[ˈwɑblist]	[ˈwɑbliɪst]
optimisers	{optim==ise}>er>>s >	[ˈɑptɪmɪzəz]	[ˈɑptɪmaɪzəɹz]
synchronizable	{syn==chron==ize}>able>	[sɪˈtraɪzəbl]	[siŋkrəˈnaɪzəbl]

Taylor, J., Richmond, K. (2020) Enhancing Sequence-to-Sequence Text-to-Speech with Morphology. Proc. Interspeech 2020, 1738-1742, doi: 10.21437/Interspeech.2020-1547

Assignment Guidance

Assignment 1: Analyze a TTS voice

- Festival TTS system: Concatenative TTS (Unit selection)
- Pre-recorded voice database (cstr_edi_awb_arctic_multisyn)
- Units = diphones

Choice of units to concatenate depends on:

- Target cost: how well the unit matches the linguistic specification
- Join cost: how well edges of the units match

Your task

Find out how synthetic speech is generated in this specific Festival pipeline

- What is the purpose of each module?
- What does Festival actually do?
- How does the overall process differ from human speech generation? Where would we expect to find errors?

Find errors in the TTS voice

- Describe the error
- Determine where in the pipeline it came from

Reflect on the pipeline as a whole

- What should you focus on in order to improve the voice?

Types of errors

- Text normalisation
- POS tagging/homographs
- Pronunciation (dictionary or letter-to-sound)
- Phrase break prediction
- Waveform generation (module 6)

Report - 1500 words

The report should have the following sections:

- Text normalisation
- POS tagging/homographs
- Pronunciation
- Phrase break prediction
- Waveform generation
- TTS pipeline discussion

6 sections, 10 marks each

Write-up

Sections 1-5: identify mistakes in the listed categories

- Concisely describe the mistakes (use figures/tables to help illustrate)
- Identify where the mistake originated from
- Provide evidence to support your analysis

Section 6: TTS Pipeline discussion

- Identify any general findings and implications of your investigations
- Give recommendations for how to improve the voice
- Identify tradeoffs involved in improvement solutions

Write-up

You may also get more marks for showing more depth of analysis: e.g.

- Further analysis/more evidence of mistakes' origins, e.g. if it's due to interactions of modules
- Analysis of severity of mistakes: would it affect output frequently or only rarely?
- Further analysis of the error from a phonetics perspective (why does it sound bad?)
- Discussion of potential solutions
- Other relevant insights into the errors!

There's more guidance in the assignment specification on speech.zone

This week

- Get access to Festival: in actual AT lab or remote desktop
- Follow the assignment instructions and inspect what's happening at each step of the pipeline
- Think about potential sources of errors from the front-end: This voice definitely doesn't do all the things we'd like it to do!
- This is a deliberately error-prone voice!

This year (2023-24) you don't need to submit a separate background section in your report, but you may wish to at least make some notes/figures to consolidate your understanding of what is happening in the pipeline.

Assignment 1: Festival TTS

Extra slides

Front-end output: Festival

In festival, the linguistic specification generated by the front-end is a set of relations, e.g.:

- Token
- Word
- Phrase
- Syllable
- SylStructure
- Segment

These all hold different bits of of information, e.g. words, POS tags, break positions, syllable boundaries….

TTS Front-end - Festival

See assignment instructions in lab

Front-end:

```
festival> (set! myutt (Utterance Text "Put your own text here."))
festival> (Initialize myutt)
festival> (Text myutt)
festival> (Token_POS myutt)
festival> (Token myutt)
festival> (POS myutt)
festival> (Phrasify myutt)
festival> (Word myutt)
festival> (Pauses myutt)
festival> (PostLex myutt)
```

Back-end:

```
festival> (Wave_Synth myutt)
```

http://festvox.org/docs/manual-2.4.0/festival.html
http://festvox.org/bsv/

Festival: Front-end Modules and Relations

Festival module	Added relation	Task
Text	Token	Whitespace tokenization
Token_POS		basic token identification/homograph disambiguation
Token	Word	Token to word rules building the Word relation
POS		Part of Speech tagger (e.g.HMM-based model)
Phrasify	Phrase	Predict phrase breaks (decision tree)
Word	Syllable, Segment, SylStructure	Lexical look up/LTS rules
Pauses		Prediction of pauses (e.g. decision tree)
PostLex		Post lexicon rules: modify segments based on their context. E.g. vowel reduction, contractions, etc.

Festival: Front-end Modules and Relations

Festival Relations	Description
Tokens	Token properties
Word	Word properties
Phrase	Groups of words that form phrases
Syntax	Relates on words via tree provided by parser
SylStructure	Relates words to syllables and segments
Syllable	Groups of phones that form syllables
Segment	Phones
IntEvent	Accents and boundary labels
Intonation	Relates syllables to IntEvents

Easier just to inspect the Festival output

Not all relations are generated in the voice you are analyzing for the assignment!

‘I work in Edinburgh’

Word and SylStructure relations both tell us something about words, but looking at this view doesn’t tell us the difference….

```
festival> (set! myutt (SayText "I work in Edinburgh."))

festival> (utt.relation.print myutt 'Word)
()
id _5 ; name I ; pos nn ; pos_index 0 ; pos_index_score 0 ; pbreak NB ;
id _6 ; name work ; pos_index 0 ; pos_index_score 0 ; pos nn ; pbreak NB ;
id _7 ; name in ; pos_index 4 ; pos_index_score 0 ; pos in ; pbreak NB ;
id _8 ; name Edinburgh ; pos_index 2 ; pos_index_score 0 ; pos nnp ; pbreak BB ;
Nil

festival> (utt.relation.print myutt 'SylStructure)
()
id _5 ; name I ; pos nn ; pos_index 0 ; pos_index_score 0 ; pbreak NB ;
id _6 ; name work ; pos_index 0 ; pos_index_score 0 ; pos nn ; pbreak NB ;
id _7 ; name in ; pos_index 4 ; pos_index_score 0 ; pos in ; pbreak NB ;
id _8 ; name Edinburgh ; pos_index 2 ; pos_index_score 0 ; pos nnp ; pbreak BB ;
id _9 ; name . ; pos_index 1 ; pos_index_score 0 ; pos punc ; pbreak BB ;
nil
```

```
festival> (utt.relation_tree  myutt 'SylStructure)
((("I"
   ((id "_5")
    (name "I")
    (pos "nn")
    (pos_index 0)
    (pos_index_score 0)
    (pbreak "NB")))
  (("syl" ((id "_11") (name "syl") (stress 1)))
   (("ae" ((id "_12") (name "ae") (end 0.23012498))))))
 (("work"
   ((id "_6")
    (name "work")
    (pos_index 0)
    (pos_index_score 0)
    (pos "nn")
    (pbreak "NB")))
  (("syl" ((id "_13") (name "syl") (stress 1)))
   (("w" ((id "_14") (name "w") (end 0.35974997))))
   (("@@r" ((id "_15") (name "@@r") (end 0.40056247))))
   (("r" ((id "_16") (name "r") (end 0.46806258))))
   (("k" ((id "_17") (name "k") (end 0.53968757))))))
 (("in"
   ((id "_7")
    (name "in")
    (pos_index 4)
….etc
```

Words are have word attributes: Id, name, pos,...,pbreak

Each words is made up of syllables

Each syllable is made up of segments

Morphology still helps in neural TTS (state of the art)

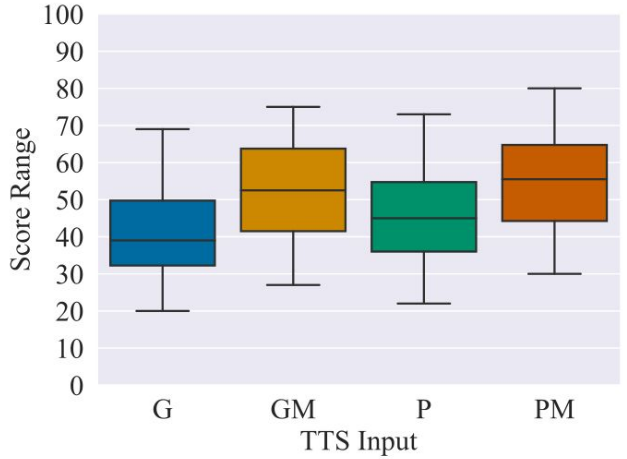


Figure 2: *Range of scores from MUSHRA listening test of each system with phones or grapheme-based input with or without morphology*

Taylor, J., Richmond, K. (2020) Enhancing Sequence-to-Sequence Text-to-Speech with Morphology. Proc. Interspeech 2020, 1738-1742, doi: 10.21437/Interspeech.2020-1547

Lexical Stress

- The syllable in a word that “attracts” prominence
- Chan change when we put words together?

e.g. Noun compounds: lot’s of influences

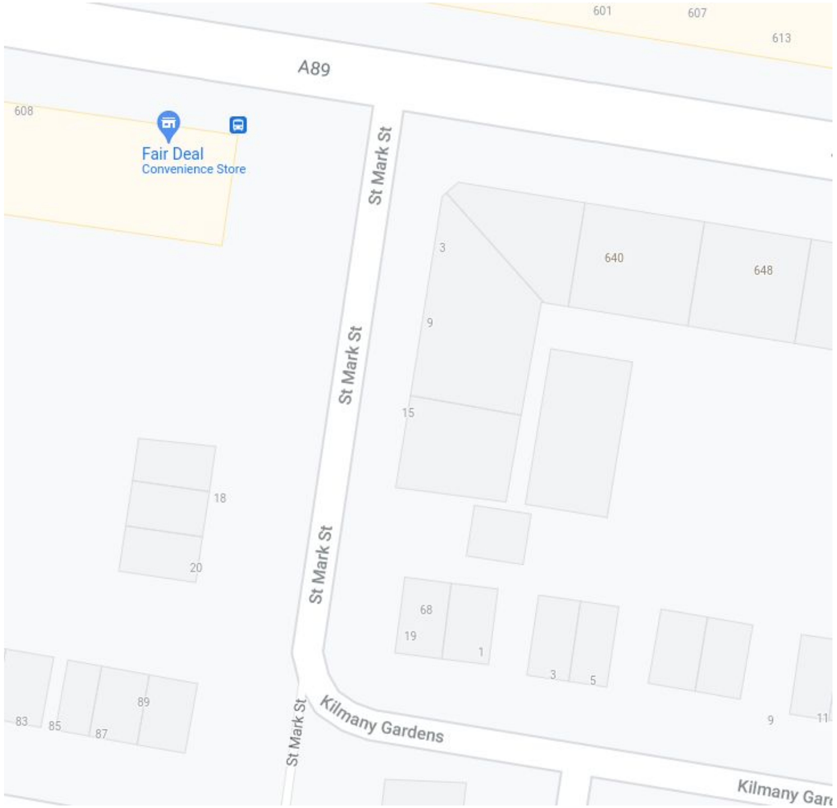
Phonological	Tennes**see**	*Ten*nessee **whis**key
Contrast/info structure	A *Kan*garoo **hunt**	A Kanga**roo** *hunt*
Syntax	**Choc**olate *cake* **ic**ing	*Choc*olate **cake** *ic*ing

Non-Standard Words: ambiguity

We can use our world language knowledge to guess how to pronounce the following:

- St Mark St Glasgow
- St Mark St, Glasgow
- St Mark St. Glasgow

But a computer needs to learn the rules...



Phoneset choice

- Unilex is more generalizable than CMUDict
- Also more compact: 1 base lexicon + rules
- But we need to define rules to convert from one accent to another
- This leads us to revisit the concept of phoneme...

Concatenative synthesis (next week)

- Concatenate units of real speech together to form new words
- Select units based on linguistic specification
- Speech within each unit is guaranteed to sound as good as human!

Some Questions

- What sized units to use?
- How to ensure they join up well? What happens if they don’t?