

Both what if there are no spaces?

- Chinese, Japanese etc. don't use whitespace
- But still need to tokenize

Some techniques

Requires lexicon of words

- Take longest match in lexicon (that gives partition)
- or find

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w}} p(\mathbf{w} | \mathbf{c}) \quad (1)$$

- Lattice of all possible partitions and find most probable

Number pronunciation

In languages with gender, declensions etc.

1 niño → un niño (one boy)

1 niña → una niña (one girl)

1 hermano → un hermano (one brother)

1 hermana → una hermana (one sister)

Can't just look at a/o ending letter

1 pais → un pais (one country)

1 ra'iz → una ra'iz (one root)

Slavic languages have *many* variations for numbers making it harder.

End of Text Analysis

From strings of characters to lists of words

- Tokenize string of chars
- Chunk into utterance sized chunks
- Identify token types (homographs, numbers etc)
- Expand tokens with token to word rules

Linguistic Analysis

From lists of words to how to say them:

- segments, duration, F0.
- Lexical look up
- Prosody generation:
 - phrasing
 - intonation: accents and F0 contours
 - durations
 - power

Part of speech tagging

- Nouns, verbs, etc
- Needed for lexical lookup
- Needed for phrase prediction
- Most likely POS tags for a word gives:
 - 92% correct (+/-)
- Content/function word distinction easy
 - (and maybe sufficient)

Use standard Ngram model

find T_1, \dots, T_n that maximize $P(T_1, \dots, T_n \mid W_1, \dots, W_n)$

$$\approx \prod_{k=1}^n \frac{P(T_k \mid T_{k-1}, \dots, T_{k-N+1})P(W_k \mid T_k)}{P(W_k)}$$

- Lexical Probabilities
 - For each W_k hold converse probability $P(W_k \mid T_k)$.
- Ngram
 - $P(T_k \mid T_{k-1}, \dots, T_{k-N+1})$
- Viterbi decoder to find best tagging

Building a tagger

- From existing tagged corpus:
 - find $P(T | W)$ by counting occurrences
 - Build trigram from data
- But if no existing tagged corpus exists:
 - tag one by hand, or ...
 - tag it with naive method
 - collect stats for probabilistic tagger
 - re-label and re-collect stats
 - repeat until done

What tag set?

But in synthesis we only need n,v,adj

Reduce → build models → predict

build models → predict → reduce

Tagset	POS Ngram model			
	uni	bi	tri	quad
ts45	90.59%	94.03%	94.44%	93.51%
ts22	95.22%	96.08%	96.33%	96.28%
45/22			97.04%	96.37%

Lexicon

- Pronunciation from words plus POS tag
- In Festival includes stress and syllabification:
 - ("project" n (((p r aa jh) 1) ((eh k t) 0)))
 - ("project" v (((p r ax jh) 0) ((eh k t) 1)))
- But need extra flags for (some homographs)

Lexicon

- Lexicon *must* give pronunciation:
 - what about morphology
- Festival lexicons have three parts:
 - a large list of words
 - a (short) addenda of words
 - letter to sound rules for everything else

Different languages

- (US) English:
 - 100,000 words (CMUDICT)
 - 50 words in addenda (modes modify this)
 - Statistically trained LTS models
- Spanish:
 - 0 words in large list
 - 50 words (symbols) in addenda
 - Hand written LTS rules

Letter to Sound rules

If language is “easy” do it by hand

- ordered set of rules

```
( LEFTCONTEXT [ ITEMS ] RIGHTCONTEXT = NEWITEMS )
```

- For example:

```
( _edge_ [ c h ] C = k )
```

```
( _edge_ [ c h ] = ch )
```

- Often rules are done in multiple-passes:
 - case normalization
 - letter to phones
 - syllabification

Letter to Sound rules

If language is “hard” train them

- For English rules by hand can be done but
 - its is a skilled job
 - time consuming
 - rule interatcions are a pain
- Need it for new languages/dialects NOW

Letter to phone alignment

What is the alignment for

checked - ch eh k t

one-to-one letter/phone pairs desirable

c	h	e	c	k	e	d
ch	-	eh	-	k	-	t

Need to find *best* alignment automatically

Letter to phone alignment algorithms

Epsilon scattering algorithm (expectation maximization)

- find all possible alignments
- estimate $\text{prob}(L,P)$ on each alignment
- iterate

Hand seeded approach

- Identify all valid letter/phone pairs e.g.
 - $c \rightarrow _ k \text{ ch s sh}$
 - $w \rightarrow _ w v f$
- find all alignments (within constraints)
- find score of L/P
- find alignment with best score

Alignments – comments

- Sometimes letters go to more than one phone, e.g.
 - x → k-s, cf. “box”
 - l → ax-l, cf. “able”
 - e → y-uw, cf. “askew”dual-phones added as phones
- Some alignments aren’t sensible
 - dept → d ih p aa r t m ah n t
 - lieutenant → l eh f t eh n ax n t
 - CMU → s iy eh m y uwBut less than 1%

Alignment comparison

Models (described next) on OALD held-out test data

Method	Letters	Words
Epsilon scattering	90.69%	63.97%
Hand-seeded	93.97%	78.13%

Hand-seeded takes time, and a little skill so fully automatic would be better.

Training models

- We use decision trees (CART/C4)
- Predict phone (dual or epsilon)
- window of 3 letters before, 3 after
 - # # # c h e c → ch
 - c h e c k e d → _

Results

On held out test (every 10th word)

Lexicon	Correct	
	Letters	Words
OALD	95.80%	74.56%
CMUDICT	91.99%	57.80%
BRULEX	99.00%	93.03%
DE-CELEX	98.79%	89.38%
Thai	95.60%	68.76%

Reflects language and lexicon coverage.

Results (2)

Stop	Correct		Size
	Letters	Words	
8	92.89%	59.63%	9884
6	93.41%	61.65%	12782
5	93.70%	63.15%	14968
4	94.06%	65.17%	17948
3	94.36%	67.19%	22912
2	94.86%	69.36%	30368
1	95.80%	74.56%	39500

An example tree

For letter V:

```
if (n.name is v)
  return _
  if (n.name is #)
    if (p.p.name is t)
      return f
      return v
    if (n.name is s)
      if (p.p.p.name is n)
        return f
        return v
      return v
```

Stress assignment

The phone string isn't enough

- train separate stress assignment
- make stressed/unstressed phones (eh/eh1)

	LTP+S	LTPS
L no S	96.36%	96.27%
Letter	—	95.80%
W no S	76.92%	74.69%
Word	63.68%	74.56%

- includes POS in LTPS (71.28% word, without)
- still missing morphological information though

Does it really work

Analysis *real* unknown words

In 39923 words in WSJ (Penn Treebank),
1775 (4.6%) not in OALD

	Occurs	%
names	1360	76.6
unknown	351	19.8
American spelling	57	3.2
typos	7	0.4

“Real” unknown words

Synthesize them with LTS models and *listen*.

Stop	Lexicon Test set	Unknown Test set	size
1	74.56%	62.14%	39500
4	65.17%	67.66%	17948
5	63.15%	70.65%	14968
6	61.65%	67.49%	12782

Best lex test is *not* best for unknown

Bootstrapping Lexicons

- Lexicon is largest (size/expensive) part of system
- If you don't have one:
 - use someone else's
- Building your own takes time

Bootstrapping Lexicons

- Find 250 most frequent words:
 - build lexical entries for them
 - ensure letter coverage in base set
 - Build its rules from this base set
- Select articles of text
- Synthesis each unknown word
 - add correct words to base list
 - correct incorrect words and add to base list
 - rebuild its rules with larger list
 - repeat

Bootstrapping Lexicons: tests

- Using CMUDICT as “oracle”
 - start with 250 common words
 - 70% accuracy
 - 25 iterations gives 97% accuracy (24,000 entries)
- Using DE-CELLEX:
 - base 350 words: 35% accurate
 - ten iterations at 90% accurate
- Real “new” lexicons:
 - Nepali
 - Ceplex (English) 12,000 entries at 98%

Dialect Lexicons

- Need new lexicons for each dialect:
 - expensive and difficult to maintain

So build dialect independent lexicon

- Build lexicon with “key vowels”:
 - the vowel in *coffee*
- vowels in *pUll* and *pOOl*:
 - In Scots English map to same
 - In Southern (UK) English map to different
- word-final ‘r’
 - delete in Southern UK English
- Plus specific pronunciation differences:
 - *leisure, route, tortoise, poem*

Post-lexical rules

- Some pronunciations require context
- For example “the”
 - before vowel dh iy
 - before consonant dh ax
- Taps in US English
- nasals in Japanese (“san” to “sam”)
- Speaker/style specific rules:
 - vowel reduction
 - contractions
 - and others

Exercises for March 26th

- * Add a post-lexical rule to modify the pronunciation of “the” before vowels, can you make it work for UK and US English.
- * Use SABLE markup to tell a joke.
- ** Write letter to sound rules to pronounce Chinese proper names (in romanized form) in (US) English.

Variable `postlex_rules_hooks` is list of functions run on utterance after lexical lookup

```
(define (postlex_thethee utt)
  (mapcar
   (lambda (seg)
     (if word is the, this is last segment,
         and next segment is a vowel
         change vowel in segment)
    )
   (utt.relation.items utt 'Segment)))
```

```
(set! postlex_rules_hooks (cons postlex_thethee postlex_rules_hooks))
```

Features are:

```
R:SYLStructure.parent.parent.name
```

```
R:SYLStructure.n.name
```

```
n.name
```

Test is with

```
(set! utt1 (SayText "The oval table. "))
(set! utt2 (SayText "The round table. "))
(utt.features utt1 'Segment '(name))
```

Telling a joke

They say telling a joke is in the timing.

- Use different speakers, breaks, etc to get the joke over.
- A sample joke is in
`http://www.cs.cmu.edu/~awb/11752/joke.txt`
- A useful audio clip is in
`http://www.cs.cmu.edu/~awb/11752/laughter.au`