CS674 Natural Language Processing

- Last class
  - SENSEVAL
  - Spelling correction
  - Pronunciation variation
- Today
  - Noisy channel model
    - » Bayesian approach to spelling correction
    - » Bayesian method for pronunciation

Probabilistic transduction

- surface representation → lexical representation
- sequence of letters in a mis-spelled word → sequence of letters in correctly spelled words
  - *acress* → *actress, cress, acres*
- string of symbols representing the pronunciation of a word in context → string of symbols representing the dictionary pronunciation
  - [er] → *her, were, are, their, your*
  - exacerbated by **pronunciation variation**
    - » *the* pronounced as THEE or THUH
    - » some aspects of this variation are systematic, like spelling error patterns

Noisy channel model

- Channel introduces noise which makes it hard to recognize the true word.
- **Goal:** build a model of the channel so that we can figure out how it modified the true word…so that we can recover it.

Decoding algorithm

- Special case of **Bayesian inference**
  - Bayesian classification
    - » Given observation, determine which of a set of classes it belongs to.
    - » Observation
      - ◆string of phones or string of letters
    - » Classify into
      - ◆words

Pronunciation subproblem

- Given a string of phones, e.g. [ni], determine which word corresponds to this string of phones
  - Consider all words in the vocabulary, *V*
  - Select the single word such that *P (word|observation)* is highest

$$\hat{w} = \arg\max_{w \in V} \quad P(w \mid O)$$

Bayesian approach

- Use Bayes' rule to transform into a product of two probabilities, each of which is easier to compute than *P(w|O)*

$$P(x \mid y) = \frac{P(y \mid x) \quad P(x)}{P(y)}$$

$$\hat{w} = \arg\max_{w \in V} \quad \frac{\overbrace{P(O \mid w)}^{\text{likelihood}} \quad \overbrace{P(w)}^{\text{prior}}}{P(O)}$$

Bayesian spelling correction

- Given a misspelled word/typo, *t*, (e.g. *acress*), determine which of the candidate corrections is the most likely
  - Consider all candidate corrections, *C*
  - Select the correction, *c*, such that *P (c|t)* is highest

$$\hat{c} = \arg\max_{c \in C} \quad \overbrace{P(t \mid c)}^{\text{likelihood}} \quad \overbrace{P(c)}^{\text{prior}}$$

Computing the prior

- $P(c) = \frac{C(c)}{N}$
- Problem: counts of 0
- Solution: *smoothing*

$$P(c) = \frac{C(c) + 0.5}{N + 0.5 \mid V \mid}$$

Resulting prior probabilities

c	freq(c)	P(c)
actress	1343	.0000315
cress	0	.000000014
caress	4	.0000001
access	2280	.000058
across	8436	.00019
acres	2879	.000065

Computing the likelihood

- Computing the likelihood term *P(t|c)* exactly is an unsolved problem
- Can estimate its value
  - The most important factors predicting an insertion, deletion, transposition are simple local factors
- Simple method: estimate the number of times that a single-letter error occurs in some large corpus of errors
  - E.g. estimate P(*acress* | *across*) using the number of times that *e* was substituted for *o*

Confusion matrices

- One for each type of single-error
  - sub[x,y]
    - » # of times that *x* was typed as *y*
    - » sub[*o,e*] = # of times that *e* was substituted for *o*
  - trans[x,y]
    - » # of times that *xy* was typed as *yx*
  - del[x,y]
    - » # of times that the characters *xy* in the correct word were typed as *x*
  - ins[x,y]
    - » # of times that the character *x* in the correct word was typed as *xy*

Estimating *P(t|c)*

- If deletion, e.g.

  P(acress|actress) =

  $$\frac{\text{\# times } ct \text{ is mistyped as } c}{\text{\# times } ct \text{ appears}}$$

- More generally,

$$P(t \mid c) = \frac{del[c_{p-1}, c_p]}{count(c_{p-1}c_p)}$$

where c_p is the *p*th character of the word *c*
t_p is the *p*th character of the word *t*

Estimating *P(t|c)*

- If substitution, e.g.

  P(acress|across) =

  $$\frac{\text{\# times } e \text{ is substituted for } o}{\text{\# times } o \text{ appears}}$$

- More generally,

$$P(t \mid c) = \frac{sub[t_p, c_p]}{count(c_p)}$$

where c_p is the *p*th character of the word *c*
t_p is the *p*th character of the word *t*

Estimating *P(t|c)*

$$P(t|c) = \begin{cases} del[c_{p-1}, c_p] / count(c_{p-1}c_p) & \text{if deletion} \\ ins[c_{p-1}, t_p] / count(c_{p-1}) & \text{if insertion} \\ sub[t_p, c_p] / count(c_p) & \text{if substitution} \\ trans[c_p, c_{p+1}] / count(c_p c_{p+1}) & \text{if transposition} \end{cases}$$

where c_p is the *p*th character of the word *c*
t_p is the *p*th character of the word *t*

Final probabilities

c	freq(c)	p(c)	p(t|c)	p(t|c)p(c)	%
actress	1343	.0000315	.000117	3.69×10^{-9}	**37%**
cress	0	.000000014	.00000144	2.02×10^{-14}	**0%**
caress	4	.0000001	.00000164	1.64×10^{-13}	**0%**
access	2280	.000058	.000000209	1.21×10^{-11}	**0%**
across	8436	.00019	.0000093	1.77×10^{-9}	**18%**
acres	2879	.000065	.0000321	2.09×10^{-9}	**21%**
acres	2879	.000065	.0000342	2.22×10^{-9}	**23%**

Context: *…was called a "stellar and versatile* ***acress*** *whose combination of sass and glamour has defined her"…*

Pronunciation subproblem

- Compute

$$\hat{w} = \arg\max_{w \in W} \overbrace{P(y \mid w)}^{\text{likelihood}} \overbrace{P(w)}^{\text{prior}}$$

- where *y* represents the sequence of phones (e.g. [ni])
- and *w* represents the candidate word

Probabilistic rules for generating pronunciation likelihoods

- Take the rules of pronunciation (see chapter 4 of J&M) and associate them with probabilities
  - Nasal assimilation rule:
- Compute the probabilities from a large labeled corpus (like the transcribed portion of Switchboard)
- Run the rules over the lexicon to generate different possible surface forms each with its own probability

Sample rules that account for [ni]

Word	Rule Name	Rule	P
the	nasal assimilation	$\eth \Rightarrow \text{n} \;/\; [+nasal]\ \#\ \_\_$	[.15]
neat	final t deletion	$t \Rightarrow \emptyset \;/\; V\ \_\_\ \#$	[.52]
need	final d deletion	$d \Rightarrow \emptyset \;/\; V\ \_\_\ \#$	[.11]
new	u fronting	$u \Rightarrow i \;/\; \_\_\ \#\ [y]$	[.36]

Computing the prior

- Using the relative frequency of the word in a large corpus
  - Brown corpus and Switchboard Treebank

w	freq(w)	P(w)
knee	61	.000024
the	114,834	.046
neat	338	.00013
need	1417	.00056
new	2625	.001

Final results

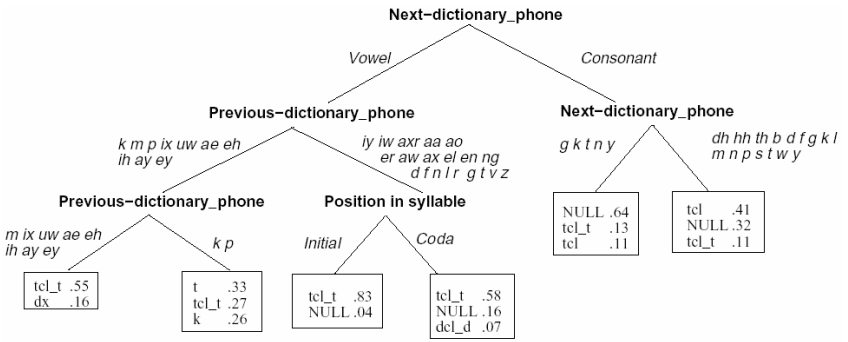
- *new* is the most likely
- Turns out to be wrong
  - *"I [ni]…"*

w	p(y|w)	p(w)	p(y|w)p(w)
new	.36	.001	.00036
neat	.52	.00013	.000068
need	.11	.00056	.000062
knee	1.00	.000024	.000024
the	0	.046	0

Decision trees for encoding lexical-to-surface pronunciation mappings

- Alternative to writing probabilistic pronunciation rules by hand is to learn the rules
- Decision tree approach
  - Riley (1991), Withgott and Chen (1993)
- Input to decision tree: a lexical phone described in terms of a set of features
- Output: classification and a probability

Example



Automatic induction of decision trees

- Riley / Withgott and Chen
  - Used CART (Breiman et al. 1984)
  - C4.5 is an alternative
- How are decision trees induced automatically?
  - Training examples
  - Top-down induction

Training data

- One tree for each lexical phone, p
  - One example for each occurrence lexical phone in corpus
  - Class value: surface realization of p
  - Features: previous-lexical-phone, next-lexical-phone, position-in-syllable