CS 224S / LINGUIST 281
Speech Recognition, Synthesis, and Dialogue

Dan Jurafsky

Lecture 4: Intro to Festival; rest of Text Normalization; Letter-to-Sound

IP Notice: lots of info, text, and diagrams on these slides comes (thanks!) from Alan Black's excellent lecture notes and from Richard Sproat's great new slides.

Outline

- Overview of Festival
  - Where it lives, its components
  - Its scripting language: Scheme
- Finishing up Part of Speech Tagging
- Phonetic Analysis
  - Dictionaries
  - Names
  - Letter-to-Sound Rules
    - (or "Grapheme-to-Phoneme Conversion")

Festival

- Open source speech synthesis system
- Designed for development and runtime use
  - Use in many commercial and academic systems
  - Distributed with RedHat 9.x, etc
  - Hundreds of thousands of users
  - Multilingual
    - No built-in language
    - Designed to allow addition of new languages
  - Additional tools for rapid voice development
    - Statistical learning tools
    - Scripts for building models

Text from Richard Sproat

Festival as software

- http://festvox.org/festival/
- General system for multi-lingual TTS
- C/C++ code with Scheme scripting language
- General replaceable modules:
  - Lexicons, LTS, duration, intonation, phrasing, POS tagging, tokenizing, diphone/unit selection, signal processing
- General tools
  - Intonation analysis (f0, Tilt), signal processing, CART building, N-gram, SCFG, WFST

Text from Richard Sproat

Festival as software

- http://festvox.org/festival/
- No fixed theories
- New languages without new C++ code
- Multiplatform (Unix/Windows)
- Full sources in distribution
- Free software

Text from Richard Sproat

CMU FestVox project

- Festival is an engine, how do you make voices?
- Festvox: building synthetic voices:
  - Tools, scripts, documentation
  - Discussion and examples for building voices
  - Example voice databases
  - Step by step walkthroughs of processes
- Support for English and other languages
- Support for different waveform synthesis methods
  - Diphone
  - Unit selection
  - Limited domain

Text from Richard Sproat

Synthesis tools

- I want my computer to talk
  - Festival Speech Synthesis
- I want my computer to talk in my voice
  - FestVox Project
- I want it to be fast and efficient
  - Flite

Text from Richard Sproat

Using Festival

- How to get Festival to talk
- Scheme (Festival's scripting language)
- Basic Festival commands

Text from Richard Sproat

Getting it to talk

- Say a file
  - **`festival --tts file.txt`**
- From Emacs
  - **`say region, say buffer`**
- Command line interpreter
  - **`festival> (SayText “hello”)`**

Scheme: the scripting lg

- Advantages of a scripting lg
  - Convenient, easy to add functionality
- Why Scheme?
  - Holdover from the LISP days of AI.
  - Many people like it.
  - It's very simple

Quick Intro to Scheme

- Scheme is a dialect of LISP
- expressions are
  - atoms or
  - lists

```
a bcd "hello world" 12.3
(a b c)
(a (1 2) seven)
```

- Interpreter evaluates expressions
  - Atoms evaluate as variables
  - Lists evaluate as functional calls

```
bxx
3.14
(+ 2 3)
```

Quick Intro to Scheme

- Setting variables

```
(set! a 3.14)
```

- Defining functions

```
(define (timestwo n) (* 2 n))

(timestwo a)
6.28
```

Lists in Scheme

- **festival> *(set! alist '(apples pears bananas))***
- (apples pears bananas)
- **festival> *(car alist)***
- apples
- **festival> *(cdr alist)***
- (pears bananas)
- **festival> *(set! blist (cons 'oranges alist))***
- (oranges apples pears bananas)
- **festival> *append alist blist***
- #<SUBR(6) append>
- (apples pears bananas)
- (oranges apples pears bananas)
- **festival> *(append alist blist)***
- (apples pears bananas oranges apples pears bananas)
- **festival> *(length alist)***
- 3
- **festival> *(length (append alist blist))***
- 7

Text from Richard Sproat

Scheme: speech

- Make an utterance of type text

```
festival> (set! utt1 (Utterance Text “hello”))
#<Utterance 0xf6855718>
```

- Synthesize an utterance

```
festival> (utt.synth utt1)
#<Utterance 0xf6855718>
```

- Play waveform

```
festival> (utt.play utt1)
#<Utterance 0xf6855718>
```

- Do all together

```
festival> (SayText “This is an example”)
#<Utterance 0xf6961618>
```


Text from Richard Sproat

Scheme: speech

- In a file

```
(define (SpeechPlus a b)
  (SayText
   (format nil
    "%d plus %d equals %d"
    a b (+ a b))))
```

- Loading files

```
festival> (load "file.scm")
t
```

- Do all together

```
festival> (SpeechPlus 2 4)
#<Utterance 0xf6961618>
```

Scheme: speech

```
(define (sp_time hour minute)
      (cond
        (( < hour 12)
         (SayText
          (format nil
          “It is %d %d in the morning”
          hour minute )))
        (( < hour 18)
        (SayText
          (format nil
          “It is %d %d in the afternoon”
          (- hour 12) minute )))
        (t
           (SayText
          (format nil
          “It is %d %d in the evening”
          (- hour 12) minute )))))
```


Text from Richard Sproat

Getting help

- Online manual
  - http://festvox.org/docs/manual-1.4.3
- Alt-h (or esc-h) on current symbol short help
- Alt-s (or esc-s) to speak help
- Alt-m goto man page
- Use TAB key for completion

Festival Structure

Utterance structure in Festival

- http://www.festvox.org/docs/manual-1.4.2/festival_14.html
- Features in festival
- http://www.festvox.org/docs/manual-1.4.2/festival_32.html

Hidden Markov Model Tagging

- Using an HMM to do POS tagging
- Is a special case of Bayesian inference
  - Foundational work in computational linguistics
  - Bledsoe 1959: OCR
  - Mosteller and Wallace 1964: authorship identification
- It is also related to the "noisy channel" model that we'll do when we do ASR (speech recognition)

POS tagging as a sequence classification task

- We are given a sentence (an "observation" or "sequence of observations")
  - Secretariat is expected to race tomorrow
- What is the best sequence of tags which corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words w1…wn.

Getting to HMM

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Hat ^ means "our estimate of the best one"
- Argmax_x f(x) means "the x such that f(x) is maximized"

Getting to HMM

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

Likelihood and prior

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \; \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \operatorname*{argmax}_{t_1^n} P(t_1^n|w_1^n) \approx \operatorname*{argmax}_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Two kinds of probabilities (1)

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be:
  - Compute P(NN|DT) by counting in a labeled corpus:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two kinds of probabilities (2)

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus:

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

POS tagging: likelihood and prior

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \; \overbrace{P(t_1^n)}^{\text{prior}}$$

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

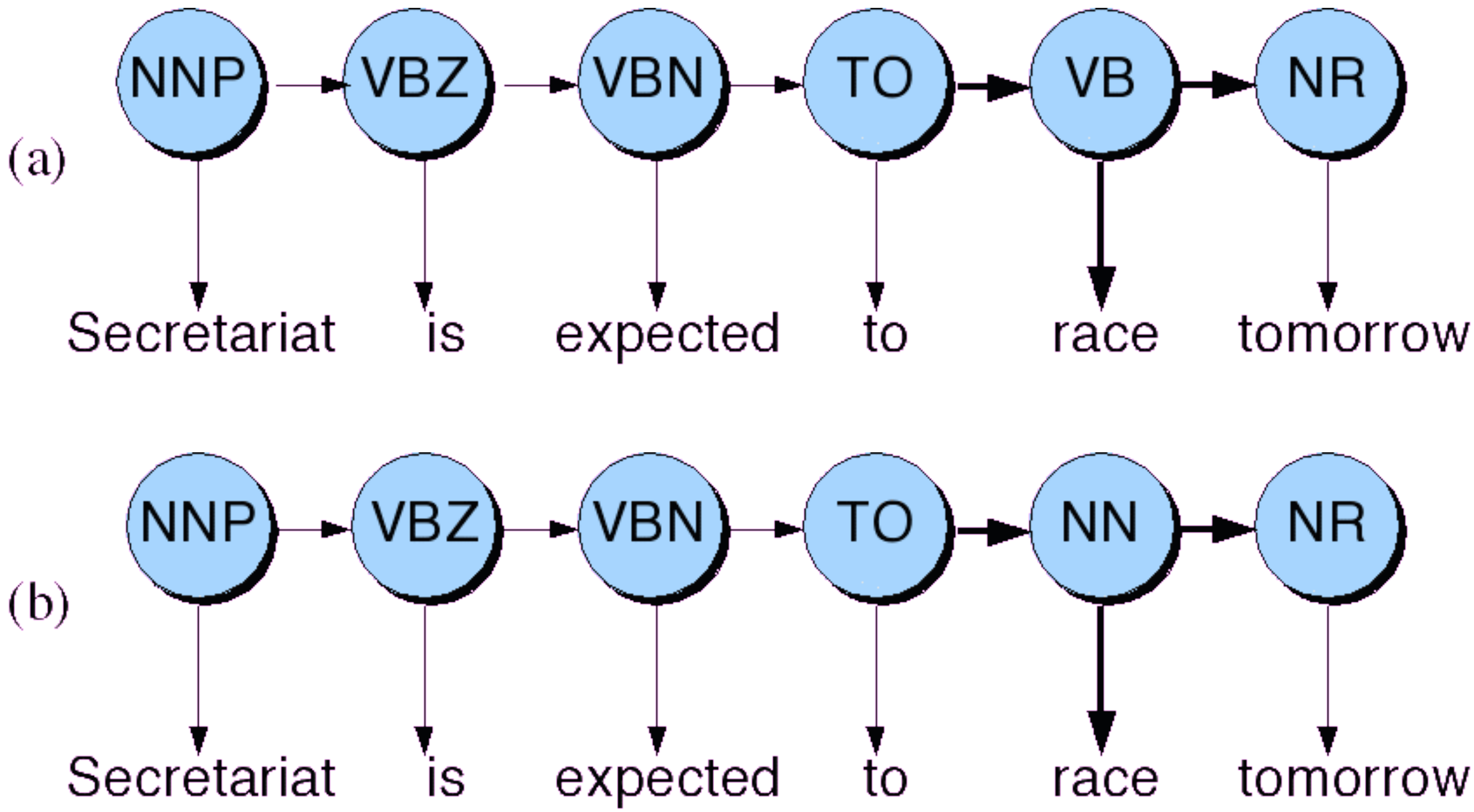
$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

An Example: the verb "race"

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating "race"



- $P(NN|TO) = .00047$
- $P(VB|TO) = .83$
- $P(race|NN) = .00057$
- $P(race|VB) = .00012$
- $P(NR|VB) = .0027$
- $P(NR|NN) = .0012$

- $P(VB|TO)P(NR|VB)P(race|VB) = .00000027$
- $P(NN|TO)P(NR|NN)P(race|NN)=.00000000032$

- So we (correctly) choose the verb reading

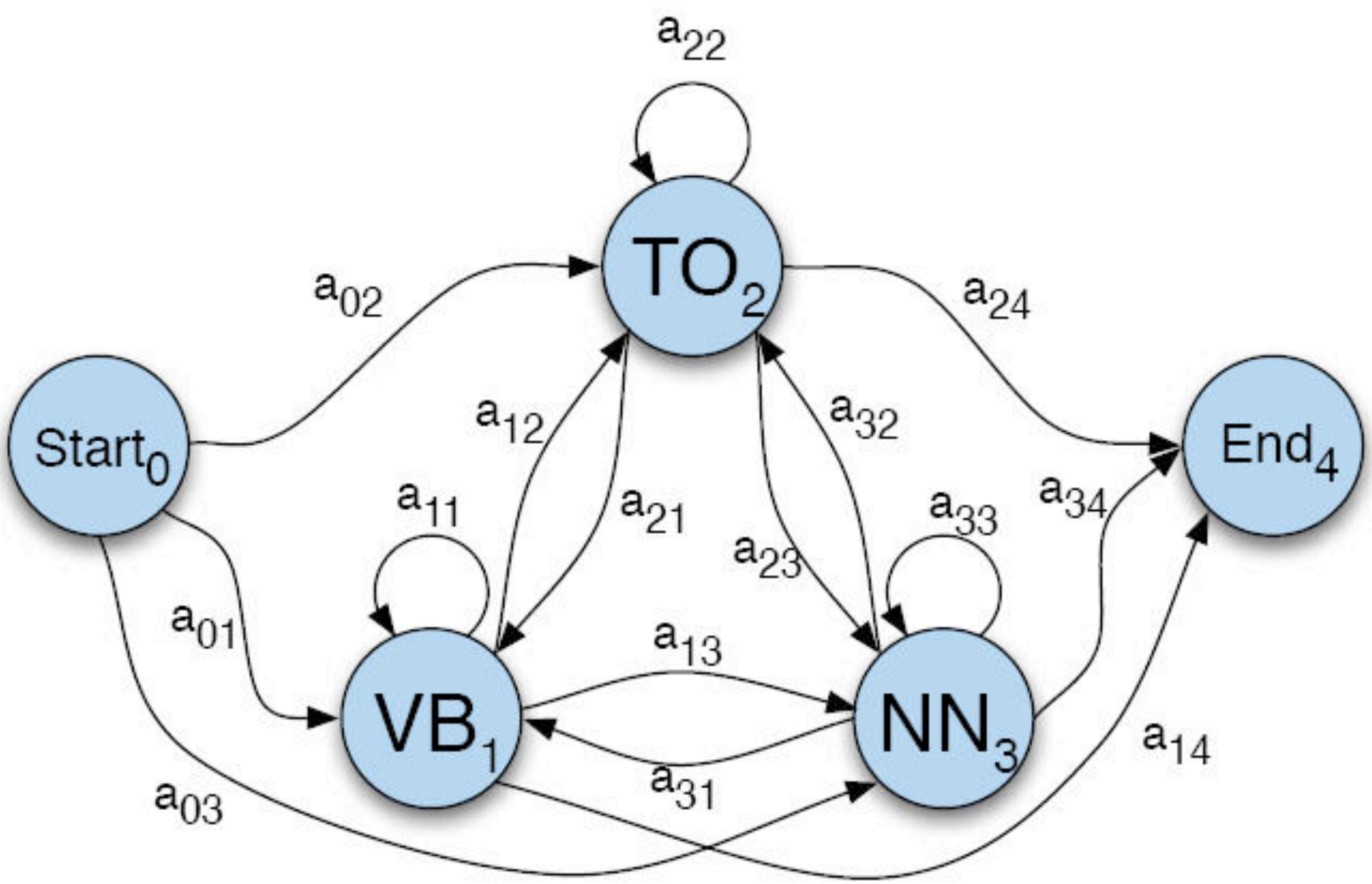
Hidden Markov Models

- What we've described with these two kinds of probabilities is a Hidden Markov Model
- A Hidden Markov Model is a particular probabilistic kind of automaton
- Let's just spend a bit of time tying this into the model
- We'll return to this in much more detail in 2 weeks when we do ASR

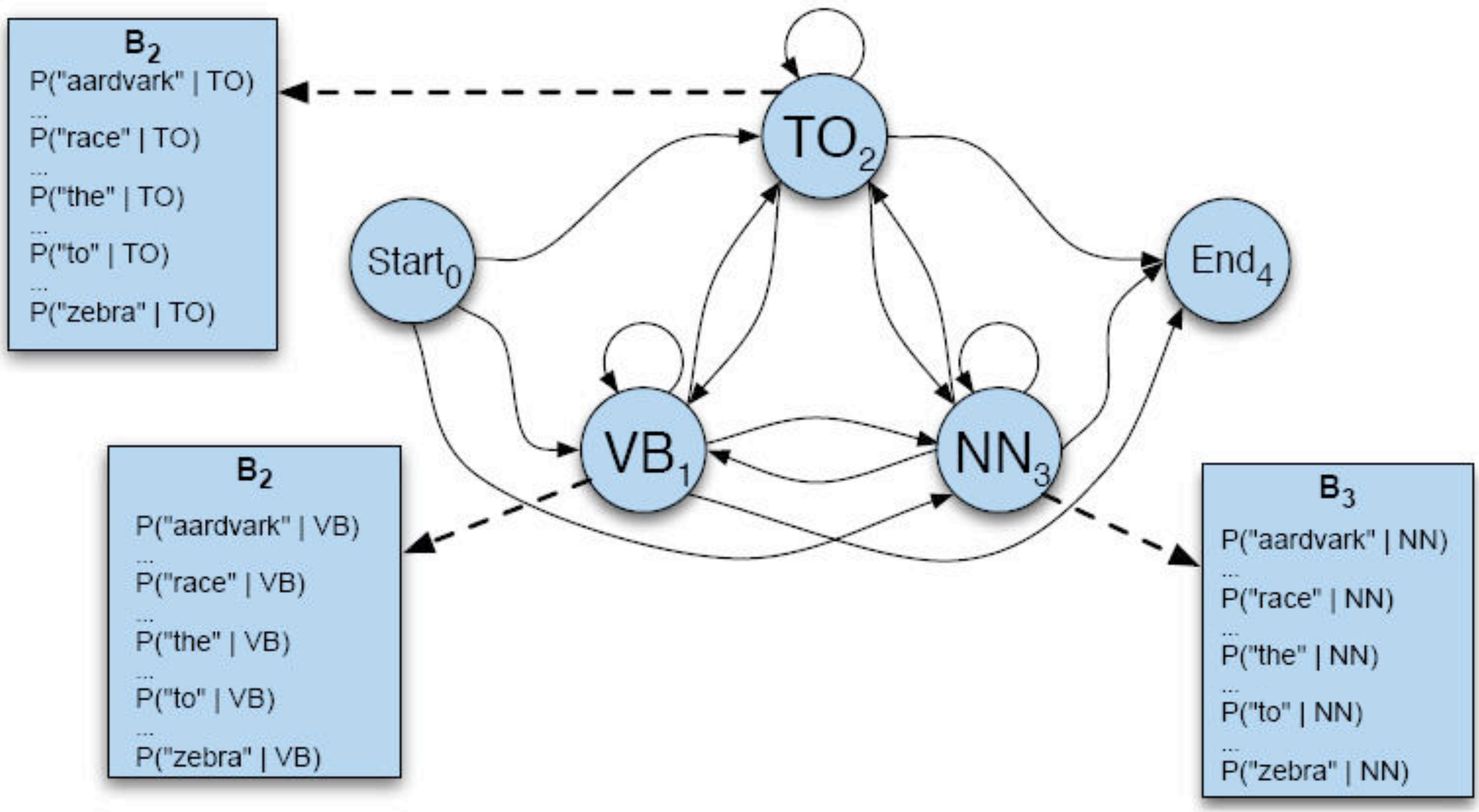
Hidden Markov Model

$Q = q_1 q_2 \ldots q_N$	a set of N **states**.
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$.
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

Transitions between the hidden states of HMM, showing A probs



B observation likelihoods for POS HMM



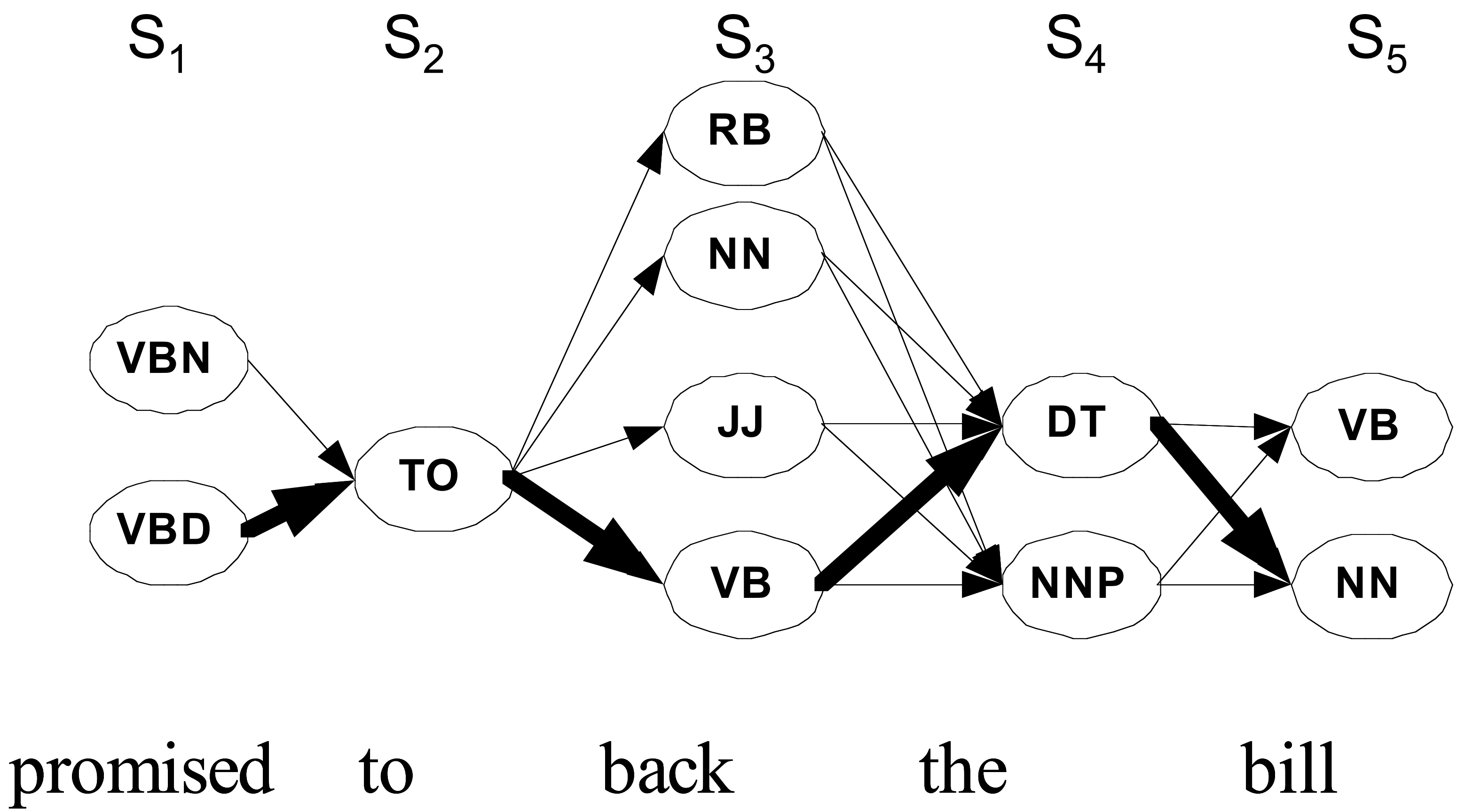
The A matrix for the POS HMM

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

The B matrix for the POS HMM

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

Viterbi intuition: we are looking for the best 'path'



Slide from Dekang Lin

The Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

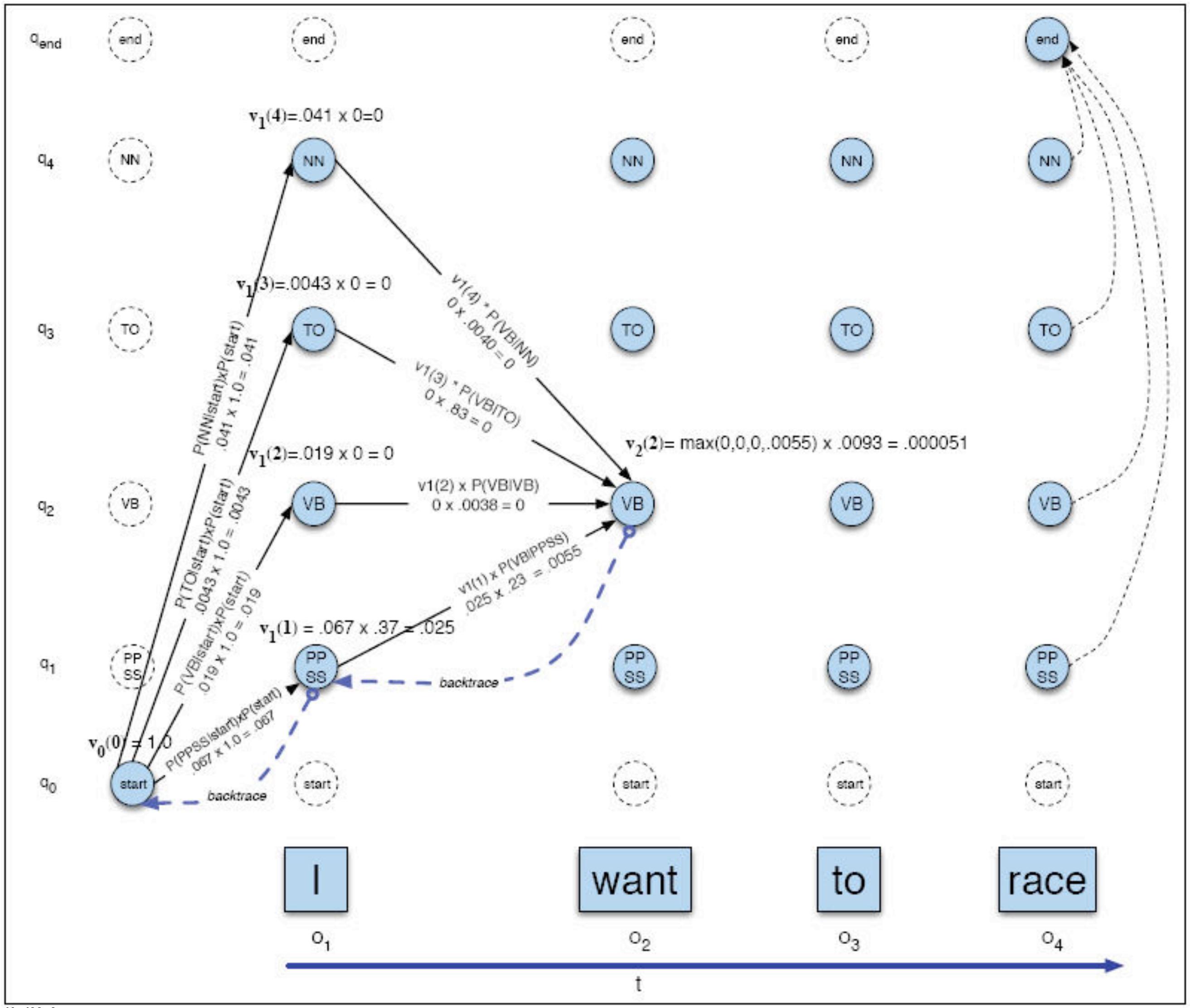
create a path probability matrix $viterbi[N+2,T]$
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Intuition

- The value in each cell is computed by taking the MAX over all paths that lead to this cell.
- $$v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$
- An extension of a path from state i at time t-1 is computed by multiplying:

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

Viterbi example

- 

Error Analysis: ESSENTIAL!!!

- Look at a confusion matrix

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

- See what errors are causing problems
  - Noun (NN) vs ProperNoun (NN) vs Adj (JJ)
  - Adverb (RB) vs Particle (RP) vs Prep (IN)
  - Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

Evaluation

- The result is compared with a manually coded "Gold Standard"
  - Typically accuracy reaches 96-97%
  - This may be compared with result for a baseline tagger (one that uses no context).
- Important: 100% is impossible even for human annotators.

Baseline

- Most frequent class baseline

Summary

- Part of speech tagging plays important role in TTS
- Most algorithms get 96-97% tag accuracy
- Not a lot of studies on whether remaining error tends to cause problems in TTS

Summary

I. Text Processing
   1) Text Normalization
      - Tokenization
      - End of sentence detection
        - Methodology: decision trees
   2) Homograph disambiguation
   3) Part-of-speech tagging
      - Methodology: Hidden Markov Models

II. Phonetic Analysis

Converting from words to phones

- Most important: dictionary

Dictionaries

- CMU dictionary: 127K words
  - http://www.speech.cs.cmu.edu/cgi-bin/cmudict

ANTECEDENTS	AE2 N T IH0 S IY1 D AH0 N T S	*PAKISTANI*	P AE2 K IH0 S T AE1 N IY0
CHANG	CH AE1 NG	*TABLE*	T EY1 B AH0 L
DICTIONARY	D IH1 K SH AH0 N EH2 R IY0	*TROTSKY*	T R AA1 T S K IY2
DINNER	D IH1 N ER0	*WALTER*	W AO1 L T ER0
LUNCH	L AH1 N CH	*WALTZING*	W AO1 L T S IH0 NG
MCFARLAND	M AH0 K F AA1 R L AH0 N D	*WALTZING(2)*	W AO1 L S IH0 NG

- Unisyn dictionary

going: { g * ou }.> i ng >
antecedents: { * a n . t^ i . s ~ ii . d n! t }> s >
dictionary: { d * i k . sh @ . n ~ e . r ii }

Lexicons and Lexical Entries

- In Festival you can explicitly give pronunciations for words
  - Each lg/dialect has its own lexicon
  - You can lookup words with
    - `(lex.lookup WORD)`
  - You can add entries to the current lexicon
    - `(lex.add.entry NEWENTRY)`
  - Entry: `(WORD POS (SYL0 SYL1…))`
  - Syllable: `((PHONE0 PHONE1 …) STRESS )`
  - Example:

```
'("cepstra" n ((k eh p) 1) ((s t r aa) 0))))
```

Dictionaries aren't always sufficient

- Unknown words
  - Seem to be linear with number of words in unseen text
  - Mostly person, company, product names
  - But also foreign words, etc.
  - From a Black et al analysis
    - Of 39K tokens in part of the Wall Street Journal
    - 1775 (4.6%) were not in the OALD dictionary:

Names	Unknown	Typos and other
1360	351	64
76.6%	19.8%	3.6%

- So commercial systems have 3-part system:
  - Big dictionary
  - Special code for handling names
  - Machine learned LTS system for other unknown words

Names

- Big problem area is names
- Names are common
  - 20% of tokens in typical newswire text will be names
  - Spiegel (2003) estimate of US names:
    - 2 million surnames
    - 100,000 first names
  - Personal names: McArthur, D'Angelo, Jiminez, Rajan, Raghavan, Sondhi, Xu, Hsu, Zhang, Chang, Nguyen
  - Company/Brand names: Infinit, Kmart, Cytyc, Medamicus, Inforte, Aaon, Idexx Labs, Bebe

Names

- Methods:
  - Can do morphology (Walters -> Walter, Lucasville)
  - Can write stress-shifting rules (Jordan -> Jordanian)
  - Rhyme analogy: Plotsky by analogy with Trostsky (replace tr with pl)
  - Liberman and Church: for 250K most common names, got 212K (85%) from these modified-dictionary methods, used LTS for rest.
  - Can do automatic country detection (from letter trigrams) and then do country-specific rules

Letter-to-Sound Rules

- Festival LTS rules:
- (LEFTCONTEXT [ITEMS] RIGHTCONTEXT = NEWITEMS)
- Example:
  - (# [c h] C = k)
  - (# [c h] = ch)
- # denotes beginning of word
- C means all consonants
- Rules apply in order
  - "christmas" pronounced with [k]
  - But word with ch followed by non-consonant pronounced [ch]
    - E.g., "choice"

What about stress: practice

- Generally
- Pronounced
- Exception
- Dictionary
- Significant
- Prefix
- Exhale
- Exhalation
- Sally

Stress rules in LTS

- English famously evil: one from Allen et al 1987
- V -> [1-stress] / X_C* {Vshort C C?|V} {[Vshort C*|V}
- Where X must contain all prefixes:
- Assign 1-stress to the vowel in a syllable preceding a weak syllable followed by a morpheme-final syllable containing a short vowel and 0 or more consonants (e.g. difficult)
- Assign 1-stress to the vowel in a syllable preceding a weak syllable followed by a morpheme-final vowel (e.g. oregano)
- etc

Modern method: Learning LTS rules automatically

- Induce LTS from a dictionary of the language
- Black et al. 1998
- Applied to English, German, French
- Two steps: alignment and (CART-based) rule-induction

Alignment

- Letters: c h e c k e d
- Phones: ch _ eh _ k _ t
- Black et al Method 1:
  - First scatter epsilons in all possible ways to cause letters and phones to align
  - Then collect stats for P(phone|letter) and select best to generate new stats

L:	c	a	k	e
P:	K	EY	K	ϵ

$$p(p_i|l_j) = \frac{\text{count}(p_i, l_j)}{\text{count}(l_j)}$$

  - This iterated a number of times until settles (5-6)
  - This is EM (expectation maximization) alg

Alignment

- Black et al method 2
- Hand specify which letters can be rendered as which phones
  - C goes to k/ch/s/sh
  - W goes to w/v/f, etc
  - An actual list:

```
c: k ch s sh t-s ϵ
e: ih iy er ax ah eh ey uw ay ow y-uw oy aa ϵ
```

- Once mapping table is created, find all valid alignments, find p(letter|phone), score all alignments, take best

Alignment

- Some alignments will turn out to be really bad.
- These are just the cases where pronunciation doesn't match letters:
  - Dept   d ih p aa r t m ah n t
  - CMU   s iy eh m y uw
  - Lieutenant   l eh f t eh n ax n t (British)
- Also foreign words
- These can just be removed from alignment training

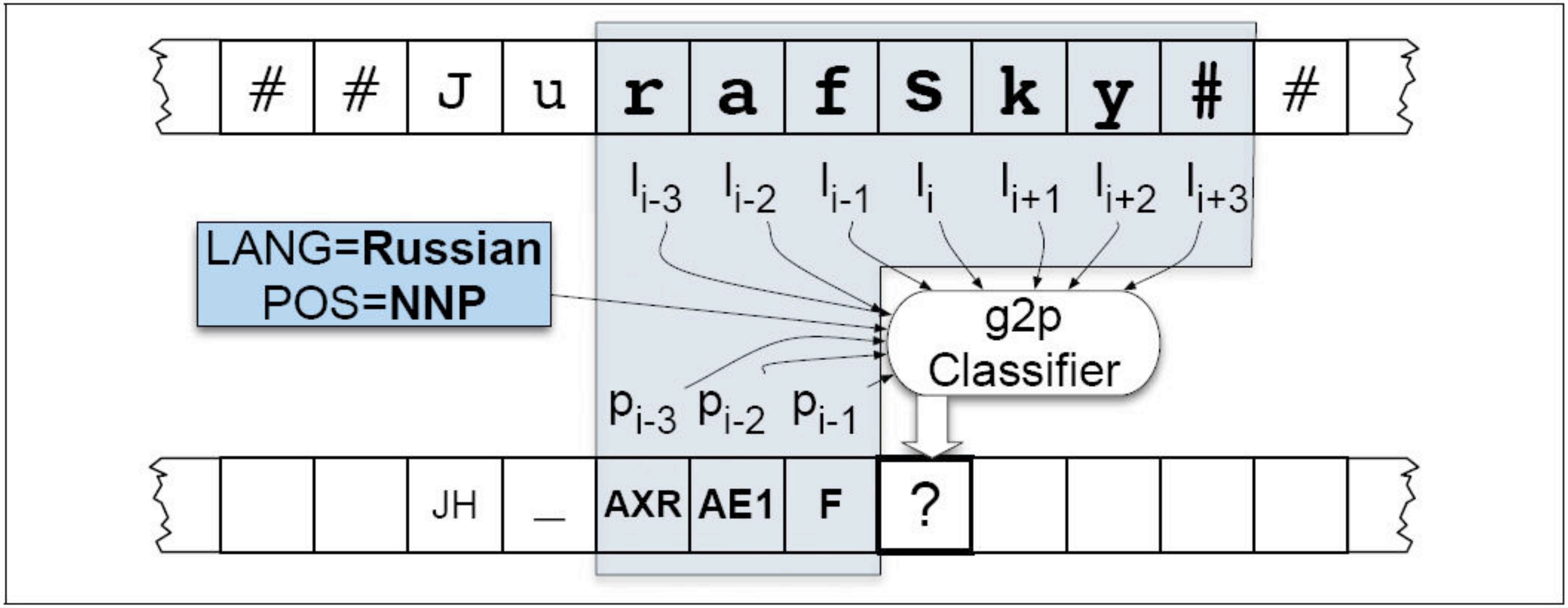
Building CART trees

- Build a CART tree for each letter in alphabet (26 plus accented) using context of +-3 letters
- # # # c h e c -> ch
- c h e c k e d -> _
- This produces 92-96% correct LETTER accuracy (58-75 word acc) for English

Improvements

- Take names out of the training data
- And acronyms
- Detect both of these separately
- And build special-purpose tools to do LTS for names and acronyms

Add more features

- 

Summary

- Overview of Festival
  - Where it lives, its components
  - Its scripting language: Scheme
- Finishing up Part of Speech Tagging
- Phonetic Analysis
  - Dictionaries
  - Names
  - Letter-to-Sound Rules
    - (or "Grapheme-to-Phoneme Conversion")