Natural Language Processing

Lecture 8—9/17/2015

Jim Martin

Today

- Finish up smoothing
  - Kneser-Ney example
- HMMs
  - POS tagging example
  - Basic HMM model
    - Decoding
      - Viterbi

Absolute Discounting

- Just subtract a fixed amount from all the observed counts (call that d).
- Redistribute it proportionally based on observed data

Absolute Discounting w/ Interpolation

$$P_{\text{AbsoluteDiscounting}}(w_i \mid w_{i-1}) = \underbrace{\frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})}}_{\text{discounted bigram}} + \underbrace{\lambda(w_{i-1})}_{\text{Interpolation weight}} \underbrace{P(w)}_{\text{unigram}}$$

4

Kneser-Ney Smoothing

- Better estimate for probabilities of lower-order unigrams!
  - Shannon game: *I can't see without my reading*__Francisco glasses___?
  - "Francisco" is more common than "glasses"
  - … but "Francisco" frequently follows "San"
- So P(w) isn't what we want

Kneser-Ney Smoothing

- $P_{continuation}(w)$: "How likely is a word to appear as a novel continuation?
  - For each word, count the number of bigram *types* it completes

$$P_{CONTINUATION}(w) \propto \left| \{ w_{i-1} : c(w_{i-1}, w) > 0 \} \right|$$

Kneser-Ney Smoothing

- Normalize that by the total number of word bigram types to get a true probability

$$P_{CONTINUATION}(w) = \frac{\left|\{w_{i-1} : c(w_{i-1}, w) > 0\}\right|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}\right|}$$

Kneser-Ney Smoothing

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left|\{w : c(w_{i-1}, w) > 0\}\right|$$

the normalized discount

The number of word types that can follow w_{i-1}

Bigram Counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

BERP

- Let's look at "chinese food". We'll need:
  - Count("chinese food") 82
  - Count("chinese") 158
  - P_continuation("food")
    - Count of bigrams "food" completes 110
    - Count of all bigram types 9421
  - Count of bigrams that "chinese" starts 17

BERP

- Let's look at "chinese food". We'll need:
  - Count("chinese food")
  - Count("chinese")
  - P_continuation("food")
    - Count of bigrams "food" completes
    - Count of all bigram types
  - Count of bigrams that "chinese" starts

Break

Word Classes: Parts of Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc
  - Called: parts-of-speech, lexical categories, word classes, morphological classes, lexical tags...
  - Lots of debate within linguistics about the number, nature, and universality of these
    - We' ll completely ignore this debate.

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a collection.

WORD	tag
the	**DET**
koala	**N**
put	**V**
the	**DET**
keys	**N**
on	**P**
the	**DET**
table	**N**

Penn TreeBank POS Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two, three*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb, base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb, past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb, gerund	*eating*
JJ	adjective	*yellow*	VBN	verb, past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb, non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb, 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, singular	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	#
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

POS Tagging

- Words often have more than one part of speech: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word in context
  - Usually a sentence

POS Tagging

- Note this is distinct from the task of identifying which sense of a word is being used given a particular part of speech. That's called word sense disambiguation. We'll get to that later.
  - "… *backed* the car into a pole"
  - "… *backed* the wrong candidate"

How Hard is POS Tagging? Measuring Ambiguity

		87-tag Original Brown	45-tag Treebank Brown
Unambiguous (1 tag)		**44,019**	**38,857**
Ambiguous (2–7 tags)		**5,490**	**8844**
Details:	2 tags	4,967	6,731
	3 tags	411	1621
	4 tags	91	357
	5 tags	17	90
	6 tags	2 (*well, beat*)	32
	7 tags	2 (*still, down*)	6 (*well, set, round, open, fit, down*)
	8 tags		4 (*'s, half, back, a*)
	9 tags		3 (*that, more, in*)

Two Methods for POS Tagging

1. Rule-based tagging
2. Stochastic
   1. Probabilistic sequence models
      - HMM (Hidden Markov Model) tagging
      - MEMMs (Maximum Entropy Markov Models)

POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that 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 $w_1 \ldots w_n$.

Getting to HMMs

- 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 HMMs

- This equation should 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 inference:
  - Use Bayes rule to transform this equation into a set of probabilities that are easier to compute (and give the right answer)

Using Bayes Rule

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

Know this.

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

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

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

Likelihood and Prior

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

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

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

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

Two Kinds of Probabilities

- 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
  - 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

- 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:

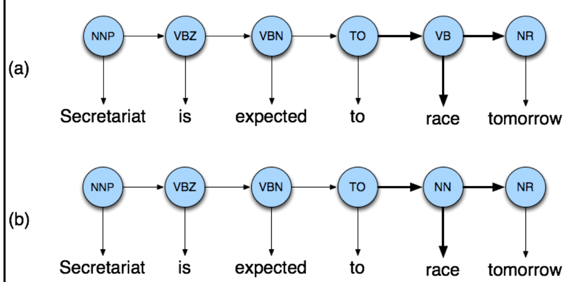
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

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

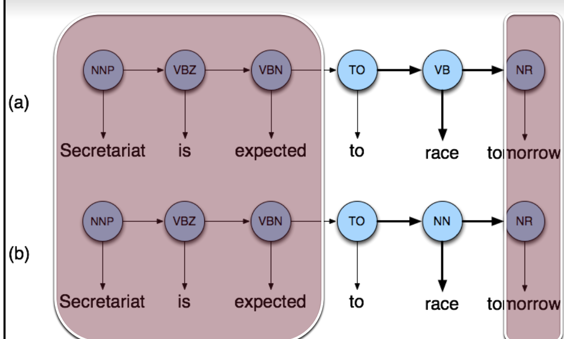
Example: The Verb "race"

- 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”

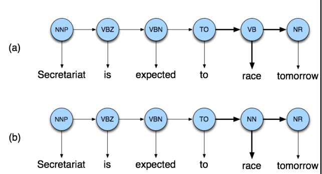


Disambiguating “race”



Example

- 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 tag for “race”

Hidden Markov Models

- What we've just described is called a Hidden Markov Model (HMM)
- This is a kind of *generative* model.
  - There is a hidden underlying generator of observable events
  - The hidden generator can be modeled as a network of states and transitions
  - We want to infer the underlying state sequence given the observed event sequence

Hidden Markov Models

- States $Q = q_1, q_2 \ldots q_N;$
- Observations $O = o_1, o_2 \ldots o_N;$
  - Each observation is a symbol from a vocabulary $V = \{v_1, v_2, \ldots v_V\}$
- Transition probabilities
  - Transition probability matrix $A = \{a_{ij}\}$

$$a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$$

- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$

$$b_i(k) = P(X_t = o_k \mid q_t = i)$$

- Special initial probability vector π

$$\pi_i = P(q_1 = i) \quad 1 \le i \le N$$

HMMs for Ice Cream

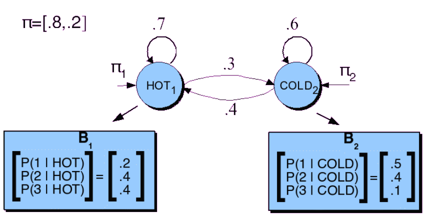
- You are a climatologist in the year 2799 studying global warming
- You can't find any records of the weather in Baltimore for summer of 2007

- But you find Jason Eisner's diary which lists how many ice-creams Jason ate every day that summer
- Your job: figure out how hot it was each day

Eisner Task

- Given
  - Ice Cream Observation Sequence: 1,2,3,2,2,2,3…
- Produce:
  - Hidden Weather Sequence:
    H,C,H,H,H,C, C…

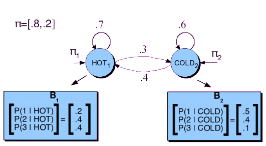
HMM for Ice Cream



Ice Cream HMM

- Let's just do 131 as the sequence
  - How many underlying state (hot/cold) sequences are there?

HHH
HHC
HCH
HCC
CCC
CCH
CHC
CHH



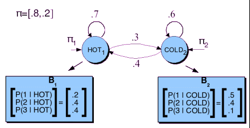
  - How do you pick the right one?

Argmax P(sequence | 1 3 1)

Ice Cream HMM

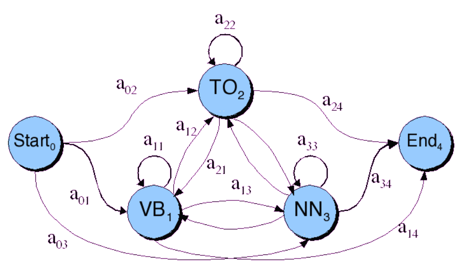
Let's just do 1 sequence: CHC

Cold as the initial state P(Cold|Start)	.2
Observing a 1 on a cold day P(1 | Cold)	.5
Hot as the next state P(Hot | Cold)	.4
Observing a 3 on a hot day P(3 | Hot)	.4
Cold as the next state P(Cold|Hot)	.3
Observing a 1 on a cold day P(1 | Cold)	.5

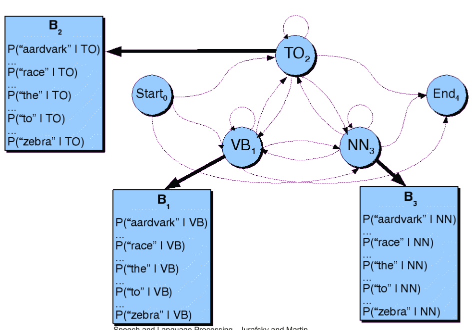


.0024

POS Transition Probabilities



Observation Likelihoods



Question

- If there are 30 or so tags in the Penn set
- And the average sentence is around 20 words...
- How many tag sequences do we have to enumerate to argmax over in the worst case scenario?

30^{20}