6.863J Natural Language Processing
Lecture 7: The Red Pill or the Blue Pill, Episode 2: part-of-speech tagging

Instructor: Robert C. Berwick
berwick@ai.mit.edu

The Menu Bar

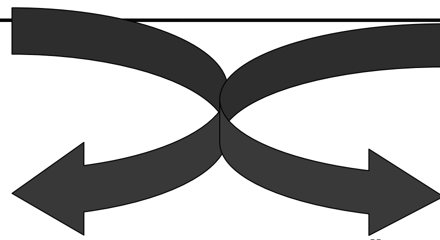
- Administrivia:
  - Schedule alert: Lab1b due today
  - Lab 2b released, this Weds (later today)

Agenda:

Red vs. Blue:

- Part of speech 'tagging' via statistical models
- Part of speech tagging via rules
- Ch. 6 & 8 in Jurafsky

The Great Divide in NLP: the red pill or the blue pill?



"Knowledge Engineering" approach
Rules built by hand w/ K of Language
"Text understanding"

"Trainable Statistical" Approach
Rules inferred from lots of data ("corpora")
"Information retrieval"

Two approaches

1. Statistical model
2. Deterministic baseline tagger composed with a cascade of fixup transducers

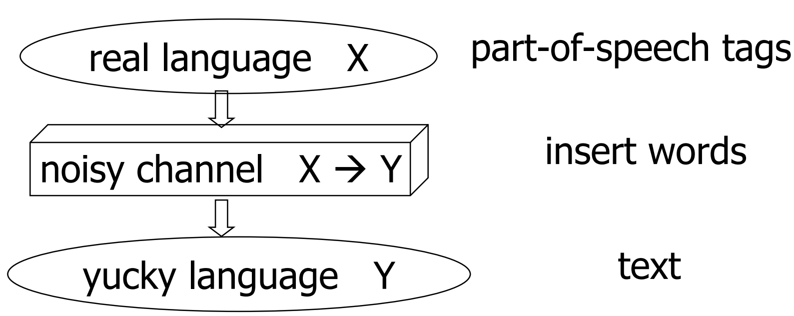
These two approaches are the guts of Lab 2

(lots of others methods: decision trees, …)

The problem

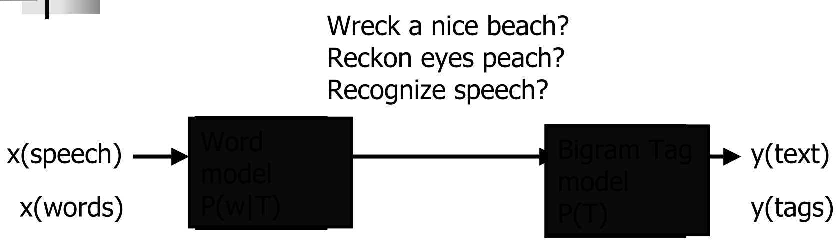
- In unseen data,we wish to find the part of speech tags
- The set of part of speech tags are decided by experts

Noishy Chunnel Muddle (statistical)



want to recover X from Y

A picture: the statistical, noisy channel view



Formulation, in general

General probabilistic decision problem

- E.g.: data = bunch of text
  - label = language
  - label = topic
  - label = author
- E.g.2: (sequential prediction)
  - label = translation or summary of entire text
  - label = part of speech of current word
  - label = identity of current word (ASR) or character (OCR)

Language models – statistical view

- Application to speech recognition (and parsing, generally)
  - x= Input (speech/words)
  - y= output (text/Tags)
  - We want to find max P(y|x) Problem: we don't know the tags – that is what we want to find!
  - Solution: We have an estimate of P(y) [the language model] and P(x|y) [the prob. of some sound/words given text/Tags = an <u>acoustic model</u> or <u>Tag model</u>]

Finding inner form given outside:

From Bayes' law, we have,

max P(y|x) = max P(x|y) • P(y) =

max Pr acoustic model x lang model

(hold P(x) fixed, i.e., P(x|y) • P(y) / P(x), but max is same for both)

So, in tagging case, we have a word model instead - so we find max P(tags|w) from: max P(words|tags) • P(tags)

HMM for POS tagging

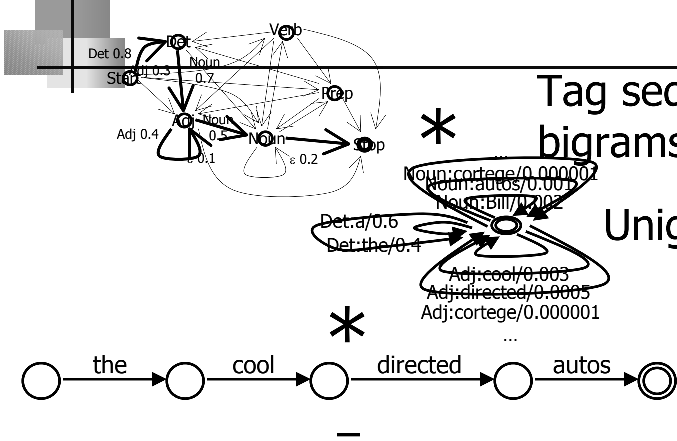
- In a Hidden Markov model, it is hypothesized that there is an underlying finite state machine (not directly observable, hence hidden) that changes state with each input element
- For us, the hidden states are the tags, and the input elements are the words

Hidden Markov Model tagging for POS

- Prob(Tag sequence) – based on n-grams: train on marked up, tagged text
- Prob(W|T) – unigram prob, based on tagged text
- Prob(T, w) computed from Viterbi trellis computation - max over all possible tag sequence paths, and ‘emission’ probabilities of word, tag combination

- Unseen tag sequence

Cartoon form Review



Tag sequence bigrams: P(T)

*

Unigram: p(W | T)

=

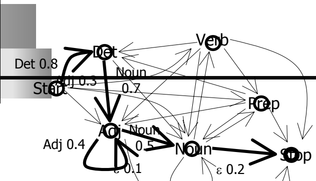
[redacted]

p(T, w)

HMM construction

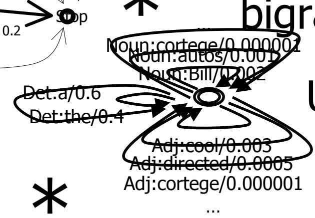
- Hidden state transition model governs observed word sequences
- Transitions probabilistic
- Estimate transition probabilities from an annotated corpus state 's' = tag state
  - $P(s_j \mid s_{j-1}, w_j)$
  - Based just on prior state and current word seen (hence Markovian assumption)
- At runtime, find maximum likelihood path through the network, using a max-flow algorithm (Viterbi)

Cartoon form Review



*

Tag sequence bigrams: P(T)

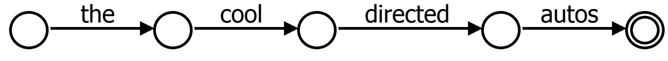


*

Unigram: p(W | T)

*

*



p(w | W)

=

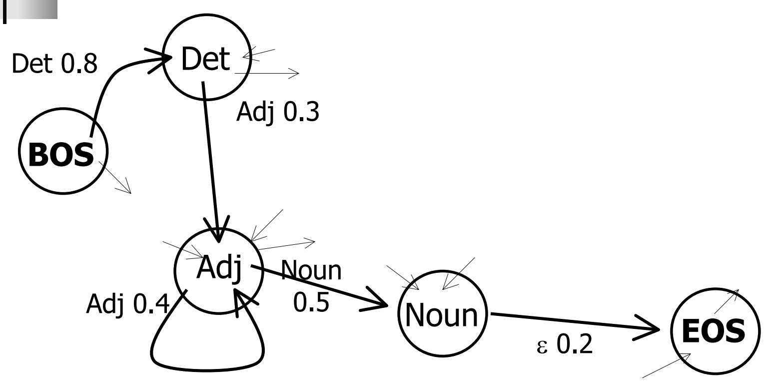
=

[redacted]

p(T, w)

P(T) - Tag bigram picture

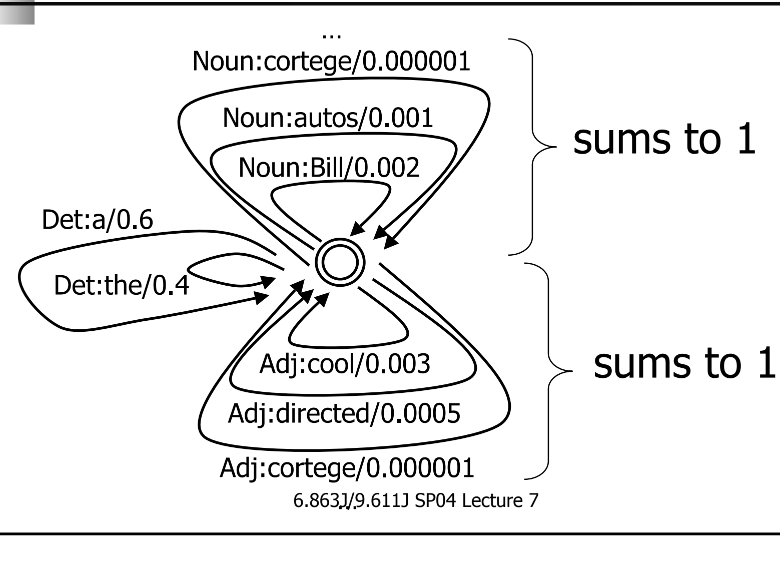
p(tag seq)



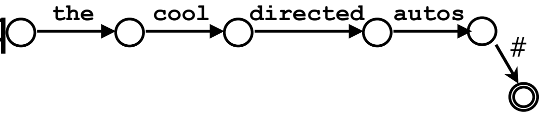
BOS Det Adj Adj Noun **EOS** = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Unigram replacement model

P(word| tag)



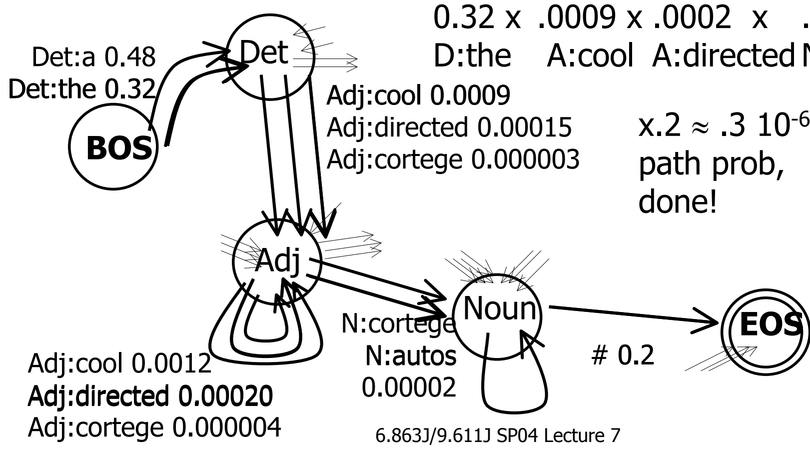
Compose with actual word seq



p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)

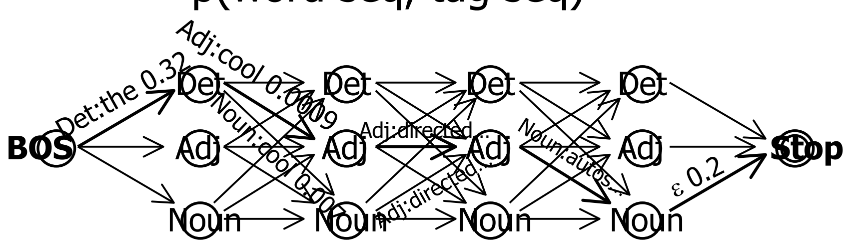
0.32 x .0009 x .0002 x .00002 x
D:the A:cool A:directed N:autos

x.2 ≈ .3 10^{-6} total path prob, done!



Unroll the fsa - All paths together form 'trellis'

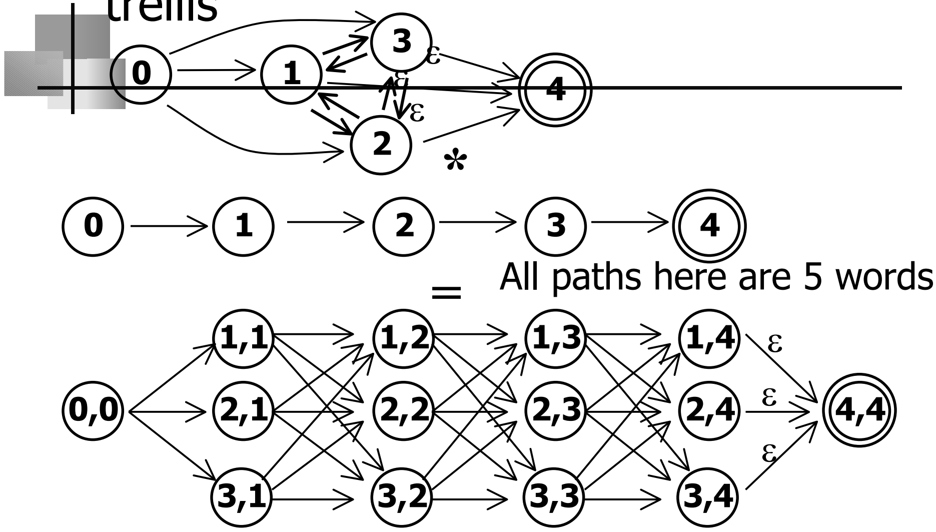
p(word seq, tag seq)



WHY?

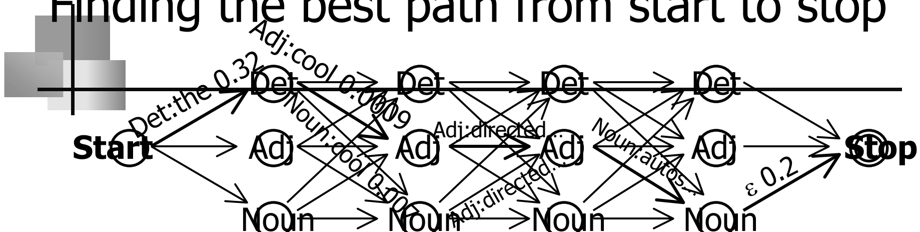
The best path:
BOS Det Adj Adj Noun **EOS** = 0.32 * 0.0009 …
the cool directed autos

Cross-product construction forms trellis



So all paths here must have 5 words on output side

Finding the best path from start to stop



- Use dynamic programming
- What is best path from Start to *each* node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)
- Special acyclic case of Dijkstra's shortest-path algorithm
- Faster if some arcs/states are absent

Method to find max probability path: Viterbi algorithm

- For <u>each</u> path reaching state s at step (word) w, we compute a path probability. We call the <u>max</u> of these <u>viterbi(s,w)</u>
- [Base step] Compute viterbi(0,0)=1
- [Induction step] Compute viterbi(s',w+1), assuming we know viterbi(s,w) for all s

Viterbi recursion

path-prob(s'|s,w) = viterbi(s,w) * a[s,s']

probability of path to s' through s — max path score for state s at word w * transition prob $s \rightarrow s'$

viterbi(s',w+1) = $\max_{\text{s in STATES}}$ path-prob(s' | s,w)

$\downarrow$ Bi-gram POS probability

Method…

- This is *almost* correct…but again, we need to also factor in the *unigram* prob of a state s' 'emitting' a word *w* given an observed surface word *w,* or $b(o_w)$ at tag state s'
- So the correct formula for the path prob is:

path-prob(s'|s,w) = viterbi(s,w) * $\underbrace{a[s,s']}_{\text{bigram}}$ * $\underbrace{b_{s'}(o_w)}_{\text{unigram}}$

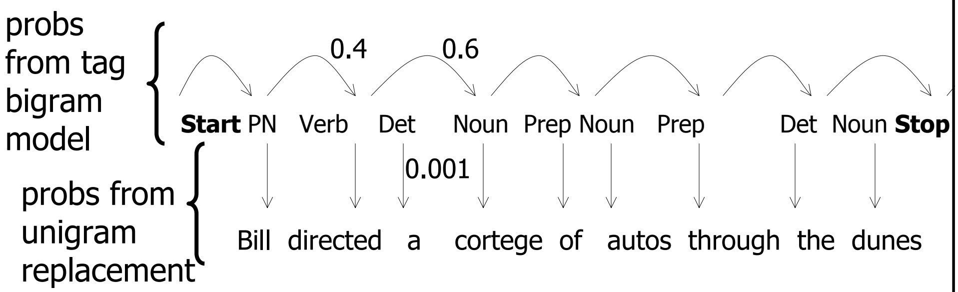
Or as in your text…p. 179 (NB: *t* used instead of *w* for index)

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

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step *t* **from** 0 **to** *T* **do**
for each state *s* **from** 0 **to** *num-states* **do**
for each transition *s'* from *s* specified by *state-graph*
new-score ← *viterbi*[*s*, *t*] * *a*[*s*,*s'*] * $b_{s'}(o_t)$
if ((*viterbi*[*s'*,*t+1*] = 0) || (*new-score* > *viterbi*[*s'*, *t+1*]))
then
viterbi[*s'*, *t+1*] ← *new-score*
back-pointer[*s'*, *t+1*] ← *s*
Backtrace from highest probability state in the final column of *viterbi[]* and return path

Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- This is a "Hidden Markov Model":



- Find X that maximizes probability product

How much data is needed?

- System performance bears a roughly log-linear relationship to the training data quantity, at least up to about 1.2 million words
- Obtaining 1.2 million words of training data requires transcribing and annotating approximately 200 hours of broadcast news programming, or if annotating text, this would amount to approximately 1,777 average-length Wall Street Journal articles

Other tasks

If you think POS tagging is not relevant, then…

- Microsoft announced plans to include "Smart Tags" in its browser and other products. This is a feature that automatically inserts hyperlinks from concepts in text to related web pages chosen by Microsoft.
- The best way to make automatic hyperlinking unbiased is to base it on an unbiased source of web pages, such as Google.

How to do this?

- The main technical problem is to find pieces of text that are good concept anchors… like names!
- So: Given a text, find the starting and ending points of all the names. Depending on our specific goals, we can include the names of people, places, organizations, artifacts (such as product names), etc.
- . Once we have the anchor text, we can send it to a search engine, retrieve a relevant URL (or set of URLs, once browsers can handle multi-way hyperlinks), and insert them into the original text on the fly.

Example – "Message understanding" (MUC)

ST1-MUC3-0011

SANTIAGO, 18 MAY 90 (RADIO COOPERATIVA NETWORK) -- [REPORT] [JUAN ARAYA]
[TEXT]
EDMUNDO VARGAS CARRENO, CHILEAN FOREIGN MINISTRY UNDER SECRETARY, HAS STATED THAT THE BRYANT TREATY WITH THE UNITED STATES WILL BE APPLIED IN THE LETELIER CASE ONLY TO COMPENSATE THE RELATIVES OF THE FORMER CHILEAN FOREIGN MINISTER MURDERED IN WASHINGTON AND THE RELATIVES OF HIS U.S. SECRETARY, RONNIE MOFFIT. THE CHILEAN FOREIGN UNDER SECRETARY MADE THIS STATEMENT IN REPLY TO U.S. NEWSPAPER REPORTS STATING THAT THE TREATY WOULD BE PARTIALLY RESPECTED.

FOLLOWING ARE VARGAS CARRENO'S STATEMENTS AT A NEWS CONFERENCE HE HELD IN BUENOS AIRES BEFORE CONCLUDING HIS OFFICIAL VISIT TO ARGENTINA:

Extracted info – names, events

0. MESSAGE: ID	TST1-MUC3-0011
1. MESSAGE: TEMPLATE	1
2. INCIDENT: DATE	18 MAY 90
3. INCIDENT: LOCATION	UNITED STATES: WASHINGTON D.C. (CITY)
4. INCIDENT: TYPE	ATTACK
5. INCIDENT: STAGE OF EXECUTION	ACCOMPLISHED
6. INCIDENT: INSTRUMENT ID	-
7. INCIDENT: INSTRUMENT TYPE	-
8. PERP: INCIDENT CATEGORY	STATE-SPONSORED VIOLENCE
9. PERP: INDIVIDUAL ID	-
10. PERP: ORGANIZATION ID	"CHILEAN GOVERNMENT"
11. PERP: ORGANIZATION CONFIDENCE REPORTED AS FACT:	"CHILEAN GOVERNMENT"
12. PHYS TGT: ID	-
13. PHYS TGT: TYPE	-
14. PHYS TGT: NUMBER	-
15. PHYS TGT: FOREIGN NATION	-
16. PHYS TGT: EFFECT OF INCIDENT	-
17. PHYS TGT: TOTAL NUMBER	-
18. HUM TGT: NAME	"ORLANDO LETELIER" "RONNIE MOFFIT"
19. HUM TGT: DESCRIPTION	"FORMER CHILEAN FOREIGN MINISTER": "ORLANDO LETELIER" "U.S. SECRETARY" / "ASSISTANT" / "SECRETARY": "RONNIE MOFFIT"
20. HUM TGT: TYPE	GOVERNMENT OFFICIAL: "ORLANDO LETELIER" CIVILIAN: "RONNIE MOFFIT"
21.	

Recognizing domain patterns

- Match domain patterns against complex phrase heads
  - <company> <form><joint venture> with <company>
  - <company> capitalized at <currency>

"Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.
"The joint venture, Bridgestone Sports Taiwan Co., capitalized at 20 million new Taiwan dollars, will start production in January 1990."

Relationship:	TIE-UP
Entities:	"Bridgestone Sports Co." "a local concern" "a Japanese trading house"
JV Company:	--
Capitalization:	--

Relationship:	TIE-UP
Entities:	--
JV Company:	"Bridgestone Sports Taiwan Co."
Capitalization:	20000000 TWD

What about part of speech tagging here?

- Advantages
  - Ambiguity can be potentially reduced (but we shall see in our laboratory if this is true)
  - Avoid errors due to incorrect categorization of rare senses e.g., “has been” as noun
- Disadvantages
  - Errors taggers make often those you’d most want to eliminate
  - High performance requires training on similar genre
  - Training takes time

Proper names…

- Proper names are particularly important for extraction systems
- Because typically one wants to extract events, properties, and relations about some particular object, and that object is usually identified by its name

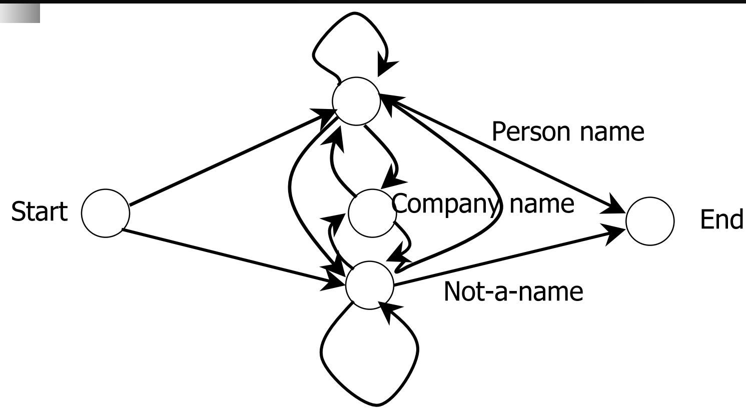
…A challenge…

- Problems though…
  - proper names are huge classes and it is difficult, if not impossible to enumerate their members
  - Hundreds of thousands of names of locations around the world
  - Many of these names are in languages other than the one in which the extraction system is designed

How are names extracted?

- (Hidden) Markov Model
- Hypothesized that there is an underlying finite state machine (not directly observable, hence hidden) that changes state with each input element
- probability of a recognized constituent is conditioned not only on the words seen, but the state that the machine is in at that moment
- "John" followed by "Smith" is likely to be a person, while "John" followed by "Deere" is likely to be a company (a manufacturer of heavy farming and construction equipment).

HMM statistical name tagger

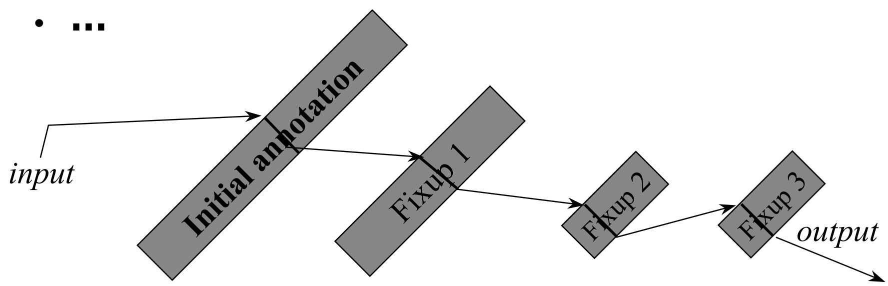


Method 2: Rule system (but learned)

- Error-based tagging

Another FST Paradigm: Successive Fixups

- Like successive markups but *alter*
- Morphology
- Phonology
- Part-of-speech tagging
- **…**



Brown/Upenn corpus tags

J. text, p. 297
Fig 8.6
1M words
60K tag counts

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 ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	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*			

figure from Brill's thesis

Transformation-Based Tagging
(Brill 1995)

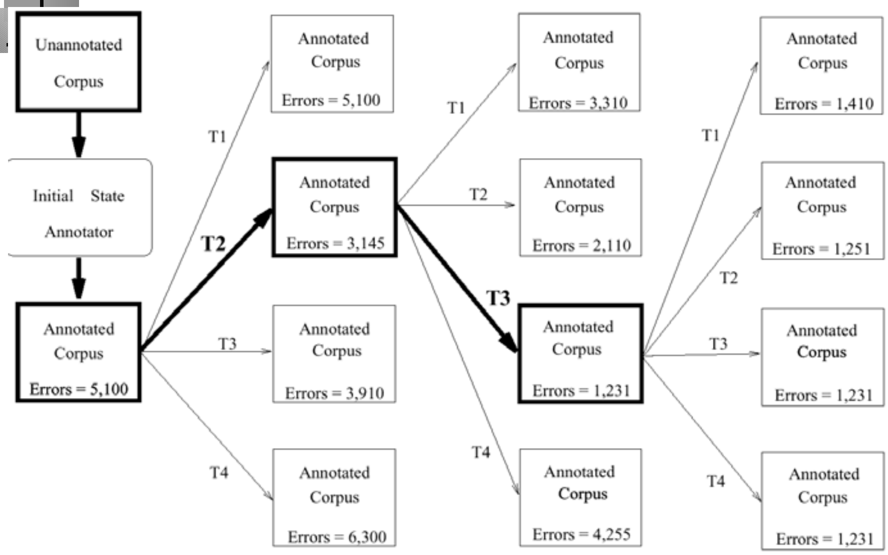


figure from Brill's thesis

Transformations Learned

	Change Tag		
#	From	To	Condition
1	NN	VB	Previous tag is *TO*
2	VBP	VB	One of the previous three tags is *MD*
3	NN	VB	One of the previous two tags is *MD*
4	VB	NN	One of the previous two tags is *DT*
5	VBD	VBN	One of the previous three tags is *VBZ*
6	VBN	VBD	Previous tag is *PRP*
7	VBN	VBD	Previous tag is *NNP*
8	VBD	VBN	Previous tag is *VBD*
9	VBP	VB	Previous tag is *TO*
10	POS	VBZ	Previous tag is *PRP*
11	VB	VBP	Previous tag is *NNS*
12	VBD	VBN	One of previous three tags is *VBP*
13	IN	WDT	One of next two tags is *VB*
14	VBD	VBN	One of previous two tags is *VB*
15	VB	VBP	Previous tag is *PRP*
16	IN	WDT	Next tag is *VBZ*
17	IN	DT	Next tag is *NN*
18	JJ	NNP	Next tag is *NNP*
19	IN	WDT	Next tag is *VBD*
20	JJR	RBR	Next tag is *JJ*

BaselineTag*
NN @→ VB // TO _
VBP @→ VB // ... _
etc.

Compose this cascade of FSTs.

Get a big FST that does the initial tagging and the sequence of fixups "all at once."

figure from Brill's thesis

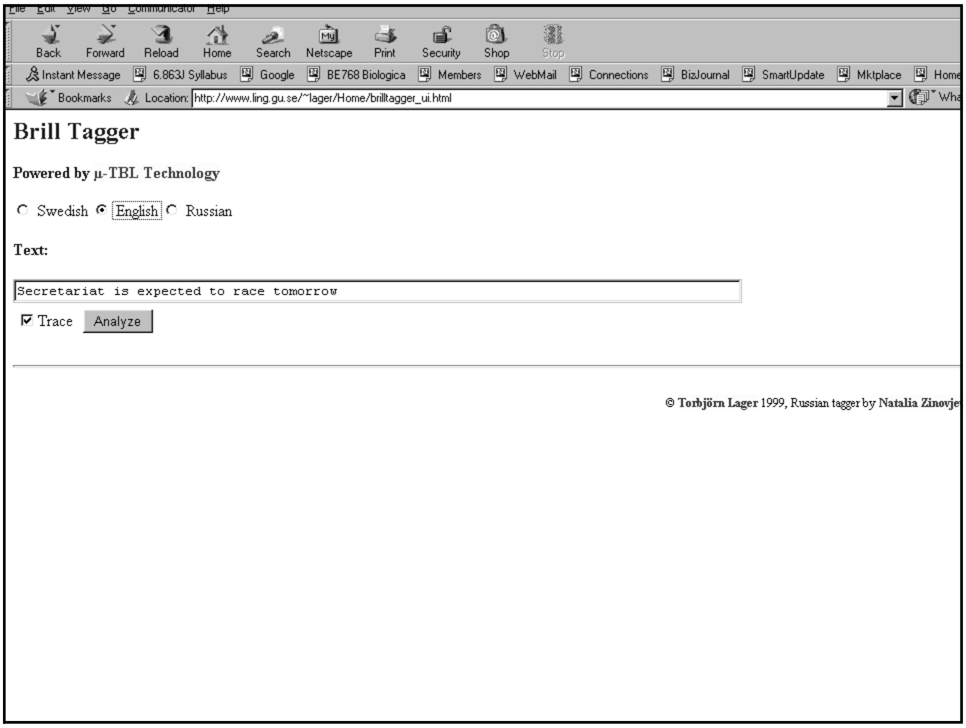
Initial Tagging of OOV Words

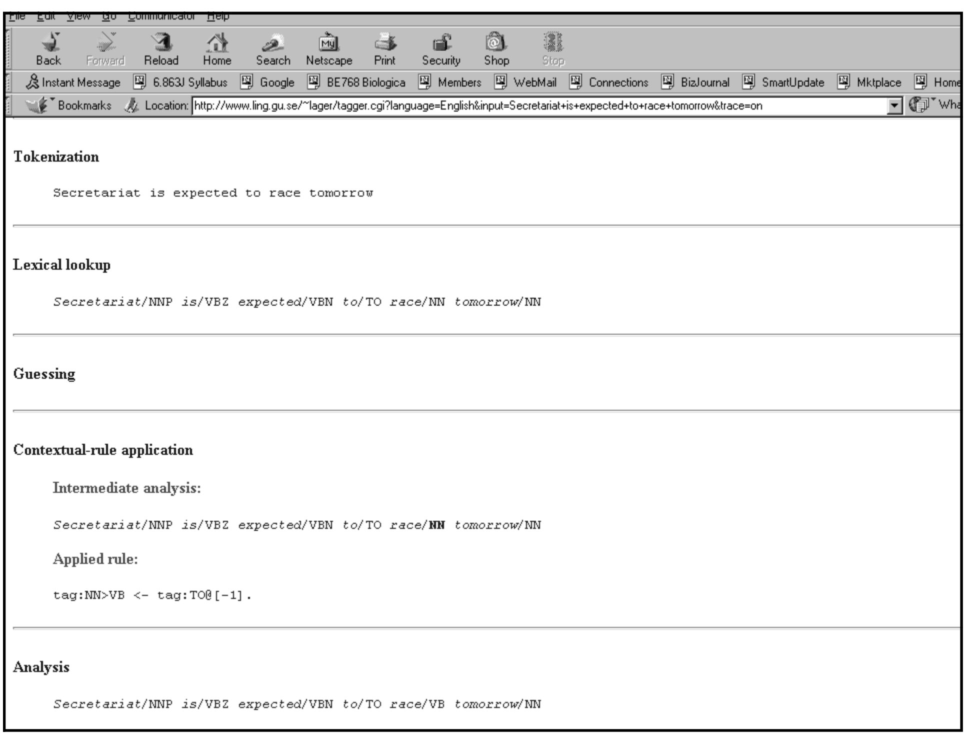
	Change Tag		
#	From	To	Condition
1	NN	NNS	Has suffix **-s**
2	NN	CD	Has character **.**
3	NN	JJ	Has character **-**
4	NN	VBN	Has suffix **-ed**
5	NN	VBG	Has suffix **-ing**
6	??	RB	Has suffix **-ly**
7	??	JJ	Adding suffix **-ly** results in a word.
8	NN	CD	The word **$** can appear to the left.
9	NN	JJ	Has suffix **-al**
10	NN	VB	The word **would** can appear to the left.
11	NN	CD	Has character **0**
12	NN	JJ	The word **be** can appear to the left.
13	NNS	JJ	Has suffix **-us**
14	NNS	VBZ	The word **it** can appear to the left.
15	NN	JJ	Has suffix **-ble**
16	NN	JJ	Has suffix **-ic**
17	NN	CD	Has character **1**
18	NNS	NN	Has suffix **-ss**
19	??	JJ	Deleting the prefix **un-** results in a word
20	NN	JJ	Has suffix **-ive**

Laboratory 2

- Goals:

1. Use both HMM and Brill taggers
2. Find errors that both make
3. Compare performance – use of kappa & 'confusion matrix'
4. All the slings & arrows of corpora – use Wall Street Journal excerpts


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message 6.863J Syllabus Google BE768 Biologica Members WebMail Connections BizJournal SmartUpdate Mktplace Home
Bookmarks Location: http://www.ling.gu.se/~lager/Home/brilltagger_ui.html
Brill Tagger
Powered by µ-TBL Technology
Swedish English Russian
Text:
Secretariat is expected to race tomorrow
Trace Analyze
© Torbjörn Lager 1999, Russian tagger by Natalia Zinovje


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message 6.863J Syllabus Google BE768 Biologica Members WebMail Connections BizJournal SmartUpdate Mktplace Home
Bookmarks Location: http://www.ling.gu.se/~lager/tagger.cgi?language=English&input=Secretariat+is+expected+to+race+tomorrow&trace=on
Tokenization
Secretariat is expected to race tomorrow
Lexical lookup
Secretariat/NNP is/VBZ expected/VBN to/TO race/NN tomorrow/NN
Guessing
Contextual-rule application
Intermediate analysis:
Secretariat/NNP is/VBZ expected/VBN to/TO race/NN tomorrow/NN
Applied rule:
tag:NN>VB <- tag:TO@[-1].
Analysis
Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NN

Transformation based tagging

- Combines symbolic and stochastic approaches: uses machine learning to refine its tags, via several passes
- Analogy: painting a picture, use finer and finer brushes - start with broad brusch that covers a lot of the canvas, but colors areas that will have to be repainted. Next layer colors less, but also makes fewer mistakes, and so on.
- Similarly: tag using broadest (most general) rule; then an narrower rule, that changes a smaller number of tags, and so on. (We haven't said how the rules are learned)
- First we will see how the TBL rules are *applied*

Contextual Rules

- *Change tag a to tag b when:*
- 1. The preceding (following) word is tagged *z*.
  2. The word two before (after) is tagged *z.*
  3. One of the two preceding (following) words is tagged *z*.
  4. One of the three preceding (following) words if tagged *z.*
  5. The preceding word is tagged *z* and the following word is tagged *w*.
  6. The preceding (following) word is tagged *z* and the word two before (after) is tagged *w*.
  7. The preceding (following) word *x*.
  …

Lexical Rules

Change the tag of an unknown word (from X) to Y if:

- 1. Deleting the prefix (suffix) *x*, |*x*| =< 4, results in a word (*x* is any string of length 1 to 4).
  2. The first (last) (1,2,3,4) characters of the word are *x*.
  3. Adding the character string *x* as a prefix (suffix) results in a word (|*x*| =< 4).
  4. Word *w* ever appears immediately to the left (right) of the word.
  5. Character *z* appears in the word.

Example Lexical Rules

NN s fhassuf 1 NNS

change the tag of an unknown word from NN to NNS if it has suffix -s

webpages/NN $\rightarrow$ *webpages/NNS*

Example 2

NN - fchar JJ

change the tag of an unknown word from NN to JJ if it has character '-'

man-made, rule-based, three-year-old, etc.

Applying the rules

1. First label every word with its most-likely tag (as we saw, this gets 90% right…!) for example, in Brown corpus, *race* is most likely to be a Noun:

   P(NN|race)= 0.98

   P(VB|race)= 0.02

2. …expected/VBZ to/T TO race/VB morrow/NN

   …the/DT race/NN for/IN outer/JJ space/NN

3. Use transformational (learned) rules to change tags:

   Change **NN** *to* **VB** *when the previous tag is* **TO**

figure from Brill's thesis

Initial Tagging of OOV Words

	Change Tag		
#	From	To	Condition
1	NN	NNS	Has suffix **-s**
2	NN	CD	Has character **.**
3	NN	JJ	Has character **-**
4	NN	VBN	Has suffix **-ed**
5	NN	VBG	Has suffix **-ing**
6	??	RB	Has suffix **-ly**
7	??	JJ	Adding suffix **-ly** results in a word.
8	NN	CD	The word **$** can appear to the left.
9	NN	JJ	Has suffix **-al**
10	NN	VB	The word **would** can appear to the left.
11	NN	CD	Has character **0**
12	NN	JJ	The word **be** can appear to the left.
13	NNS	JJ	Has suffix **-us**
14	NNS	VBZ	The word **it** can appear to the left.
15	NN	JJ	Has suffix **-ble**
16	NN	JJ	Has suffix **-ic**
17	NN	CD	Has character **1**
18	NNS	NN	Has suffix **-ss**
19	??	JJ	Deleting the prefix **un-** results in a word
20	NN	JJ	Has suffix **-ive**

OK, the proof is in the (supervised) learning pudding - How?

- 3 stages

1. Start by labeling every word with most-likely tag
2. Then examine every possible transformation, and selects one that results in most improved tagging
3. Finally, re-tags data according to this rule
4. Repeat 1-3 until some stopping criterion (no new improvement, or small improvement)

- Output is ordered list of transformations that constitute a tagging procedure

How this works

- Set of possible 'transforms' is infinite, e.g., "transform NN to VB if the previous word was *MicrosoftWindoze* & word *braindead* occurs between 17 and 158 words before *that*"
- To limit: start with small set of abstracted transforms, or *templates*

Templates used: Change *a* to *b* when…

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

Variables ***a*, *b, z, w,*** range over parts of speech

Examples of Contextual Rules

- NN VB PREVTAG TO
- change tag NN to tag VB when the preceding word is tagged TO

  to/TO run/NN

would be changed to

  to/TO run/VB

- VBP VB PREV1OR2OR3TAG MD
- Chane tag VBP(verb, non-3rd person singular present) to VB(verb, base form) when one of the three preceding words is tagged MD (modal verb)

Method

1. Call `Get-best-transform` with list of potential templates; this calls
2. `Get-best-instance` which instantiates each template over all its variables (given specific values for where we are)
3. Try it out, see what score is (improvement over known tagged system -- supervised learning); pick best one locally

```
function TBL(corpus) returns transforms-queue
 INTIALIZE-WITH-MOST-LIKELY-TAGS(corpus)
 until end condition is met do
  templates ← GENERATE-POTENTIAL-RELEVANT-TEMPLATES
  best-transform ← GET-BEST-TRANSFORM(corpus, templates)
  APPLY-TRANSFORM(best-transform, corpus)
  ENQUEUE(best-transform-rule, transforms-queue)
 end
 return(transforms-queue)

 function GET-BEST-TRANSFORM(corpus, templates) returns transform
  for each template in templates
   (instance, score) ← GET-BEST-INSTANCE(corpus, template)
   if (score > best-transform.score) then best-transform ← (instance, score)
  return(best-transform)
```



```
function GET-BEST-INSTANCE(corpus, template) returns transform
 for from-tag ← from tag−1 to tag−n do
  for to-tag ← from tag−1 to tag−n do
   for pos ← from 1 to corpus-size do
    if (correct-tag(pos) == to-tag && current-tag(pos) == from-tag)
        num-good-transforms(current-tag(pos−1))++
    elseif (correct-tag(pos)==from-tag && current-tag(pos)==from-tag)
        num-bad-transforms(current-tag(pos−1))++
   end
   best-Z ← ARGMAX_t(num-good-transforms(t) - num-bad-transforms(t))
   if(num-good-transforms(best-Z) - num-bad-transforms(best-Z)
             > best-instance.Z) then
    best-instance ← "Change tag from from-tag to to-tag
                     if previous tag is best-Z"
return(best-instance)

procedure APPLY-TRANSFORM(transform, corpus)
for pos ← from 1 to corpus-size do
 if (current-tag(pos)==best-rule-from)
     && (current-tag(pos−1)==best-rule-prev))
   current-tag(pos) = best-rule-to
```

nonlexicalized rules learned by TBL tagger

#	Change tags From	To	Condition	Example
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

figure from Brill's thesis

Transformations Learned

#	Change Tag From	To	Condition
1	NN	VB	Previous tag is *TO*
2	VBP	VB	One of the previous three tags is *MD*
3	NN	VB	One of the previous two tags is *MD*
4	VB	NN	One of the previous two tags is *DT*
5	VBD	VBN	One of the previous three tags is *VBZ*
6	VBN	VBD	Previous tag is *PRP*
7	VBN	VBD	Previous tag is *NNP*
8	VBD	VBN	Previous tag is *VBD*
9	VBP	VB	Previous tag is *TO*
10	POS	VBZ	Previous tag is *PRP*
11	VB	VBP	Previous tag is *NNS*
12	VBD	VBN	One of previous three tags is *VBP*
13	IN	WDT	One of next two tags is *VB*
14	VBD	VBN	One of previous two tags is *VB*
15	VB	VBP	Previous tag is *PRP*
16	IN	WDT	Next tag is *VBZ*
17	IN	DT	Next tag is *NN*
18	JJ	NNP	Next tag is *NNP*
19	IN	WDT	Next tag is *VBD*
20	JJR	RBR	Next tag is *JJ*

<u>BaselineTag*</u>
NN @→ VB // TO _
VBP @→ VB // ... _
etc.

Compose this cascade of FSTs.

Get a big FST that does the initial tagging and the sequence of fixups "all at once."

Error analysis: what's hard for taggers

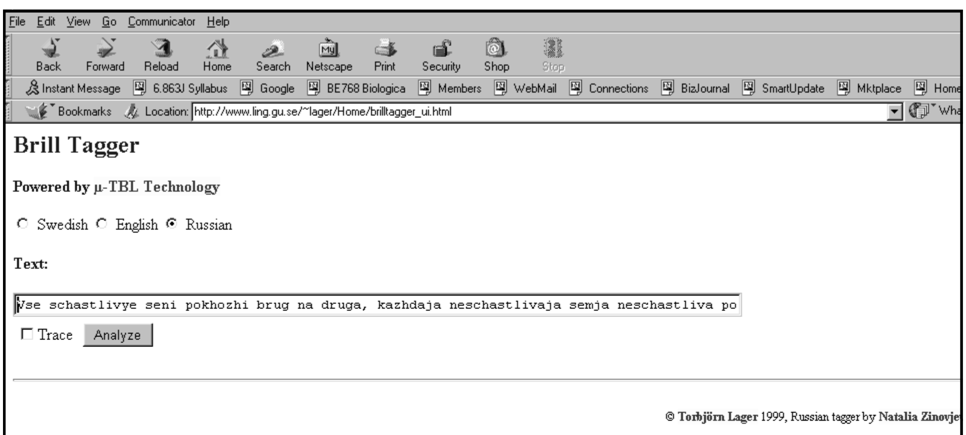
- Common errors (> 4%)
  - NN vs .NNP (proper vs. other nouns) vs. JJ (adjective): hard to distinguish prenominally; important to distinguish esp. for information extraction
  - RP vs. RB vs IN: all can appear in sequences immed. after verb
  - VBD vs. VBN vs. JJ: distinguish past tense, past participles (*raced* vs. *was raced* vs. *the out raced horse*)

What's hard

- Unknown words
  - Order 0 idea: equally likely over all parts of speech
  - Better idea: same distribution as 'Things seen once' estimator of 'things never seen' - theory for this done by Turing (again!)
  - *Hapax legomenon*
  - Assume distribution of unknown words is like this
  - But most powerful methods make use of how word is spelled
- See file in the course tagging dir on this

Or unknown language

- Vse schastlivye sen'i pokhozhi brug na druga, kazhdaja neschastlivaja sem'ja neschastliva po-svoemu



Most powerful unknown word detectors

- 3 inflectional endings (*-ed, -s, -ing*); 32 derivational endings (*-ion,* etc.); capitalization; hyphenation
- More generally: should use morphological analysis! (and some kind of machine learning approach)
- How hard is this? We don't know - we actually don't know how children do this, either (they make mistakes)

Laboratory 2

- Goals:

1. Use both HMM and Brill taggers
2. Find errors that both make, relative to genre
3. Compare performance – use of kappa & 'confusion matrix'
4. All the slings & arrows of corpora – use Wall Street Journal excerpts, as well as 'switchboard' corpus

Brown/Upenn corpus tags

J. text,
p. 297
Fig 8.6
1M words
60K tag
counts

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 ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	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*			

Evaluation of systems

- The principal measures for information extraction tasks are recall and precision.

- *Recall* is the number of answers the system got right divided by the number of possible right answers
  - It measures how complete or comprehensive the system is in its extraction of relevant information

- *Precision* is the number of answers the system got right divided by the number of answers the system gave
  - It measures the system's correctness or accuracy
  - Example: there are 100 possible answers and the system gives 80 answers and gets 60 of them right, its recall is 60% and its precision is 75%.

A better measure - Kappa

- Takes baseline & complexity of task into account – if 99% of tags are Nouns, getting 99% correct no great shakes
- Suppose no "Gold Standard" to compare against?
- P(A) = proportion of times hypothesis *agrees* with standard (% correct)
- P(E) = proportion of times hypothesis and standard would be *expected* to agree by chance (computed from some other knowledge, or actual data)

Kappa [p. 315 J&M text]

- Note K ranges between 0 (no agreement, except by chance; to complete agreement, 1)
- Can be used even if no 'Gold standard' that everyone agrees on
- K> 0.8 is good

Kappa

- *A* = actual agreement; *E* = expected agreement
- consistency is more important than "truth"
- methods for raising consistency
  - style guides (often have useful insights into data)
  - group by task, not chronologically, etc.
  - annotator acclimatization

Coda on kids

C: "Mommy, nobody don't like me"

A: No, say, "nobody likes me"

C: Nobody don't likes me

A: Say, "nobody likes me"

C: Nobody don't likes me
[7 repetitions]

C: Oh! Nobody don't like me!

Is that all there is?

What have we done so far?

- Only information we represent: is whether an item precedes (or follows) another
- Inventory of vocabulary items (classes)
- = Finite state machines

- Is there anything else in language???

Motivation

- What, How, and Why
- <u>What:</u> word *chunks* behave as units, like words or endings (morphemes), like *ing*
- <u>How:</u> we have to recover these from input
- <u>Why:</u> chunks used to discover *meaning*
- Parsing: mapping from *strings* to *structured representation*

Programming languages

```
printf ("/charset [%s",
        (re_opcode_t) *(p - 1) == charset_not ? "^" : "");

assert (p + *p < pend);

for (c = 0; c < 256; c++)
  if (c / 8 < *p && (p[1 + (c/8)] & (1 << (c % 8)))) {
     /* Are we starting a range?  */
     if (last + 1 == c && ! inrange) {
        putchar ('-');
        inrange = 1;
     }
     /* Have we broken a range?  */
     else if (last + 1 != c && inrange) {
        putchar (last);
        inrange = 0;
     }

     if (! inrange)
        putchar (c);

     last = c;
  }
```

- Easy to parse.
- Designed that way!

Natural languages

```
printf "/charset %s", re_opcode_t *p - 1 == charset_not ? "^" : "";
assert p + *p < pend; for c = 0; c < 256; c++ if c / 8 < *p && p1 + c/8
& 1 << c % 8 Are we starting a range? if last + 1 == c && !  inrange
putchar '-'; inrange = 1; Have we broken a range? else if last + 1 != c
&& inrange putchar last; inrange = 0; if !  inrange putchar c; last =
c;
```

- No {} () [] to indicate scope & precedence
- Lots of overloading (arity varies)
- Grammar isn't known in advance!
- What is the best formalism?

What can't linear relations represent?

- wine dark sea $\rightarrow$ (wine (dark sea)) *or*
  ((wine dark) sea) ?
- deep blue sky
- Can fsa's *represent* this?
- Not really: algebraically, *defined* as being associative (doesn't matter about concatenation order)