Tecnologie del Linguaggio Naturale

Parte Prima

Lezione n. 03

> PoS and NER Tagging

26 Febbraio 2024

(slide credits: many slides from D. Jurafsky!)

Part of Speech (PoS) tagging

Input:	*Plays*	*well*	*with*	*others*
Ambiguity:	NNS/VBZ	UH/JJ/NN/RB	IN	NNS
Output:	Plays/VBZ	well/RB	with/IN	others/NNS

Uses:

- Text-to-speech (how do we pronounce “lead”?)
- Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
- As input to or to speed up a full parser
- MT: reordering of adjectives and nouns (say from Spanish to English)
- Sentiment or affective tasks: may want to distinguish adjectives or other POS

- Need to control for POS when studying linguistic change like creation of new words, or meaning shift
- Or control for POS in measuring meaning similarity or difference

Sample "Tagged" English sentences

- There/PRO were/VERB 70/NUM children/NOUN there/ADV ./PUNC

- Preliminary/ADJ findings/NOUN were/AUX reported/VERB in/ADP today/NOUN 's/PART New/PROPN England/PROPN Journal/PROPN of/ADP Medicine/PROPN

How Hard is POS Tagging? Ambiguity

- 85% of word **types** are unambiguous
  - *Janet* is always PROPN, *hesitantly* is always ADV
- But those 15% of word types ambiguous tend to be very common.
- So ~60% of word **tokens** are ambiguous
- E.g., *back*

  earnings growth took a back/ADJ seat

  a small building in the back/NOUN

  a clear majority of senators back/VERB the bill

  enable the country to buy back/PART debt

  I was twenty-one back/ADV then

PoS tagging performance

How many tags are correct? (**Tag accuracy**)

- About 97% currently (English and other languages)
  - HMMs, CRFs, BERT perform similarly
- But baseline is already 92%: **Baseline** is performance of stupidest possible method
  - Tag every word with its most frequent tag
  - Tag unknown words as nouns
- Partly easy because
  - Many words are unambiguous
  - You get points for them (the, a, etc.) and for punctuation marks!

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*			

"Universal Dependencies" Tagset

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

Nivre et al. 2016

Sources of information for POS tagging

Janet will back the bill

AUX/NOUN/VERB? NOUN/VERB?

- Prior probabilities of word/tag
  - "will" is usually an AUX
- Identity of neighboring words
  - "the" means the next word is probably not a verb
- Morphology and wordshape:
  - Prefixes — unable: un- → ADJ
  - Suffixes — importantly: -ly → ADJ
  - Capitalization — Janet: CAP → PROPN

Standard algorithms for POS tagging

Supervised Machine Learning Algorithms:

- Hidden Markov Models
- Conditional Random Fields (CRF)/ Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)
- Large Language Models (like BERT), finetuned

All required a hand-labeled training set, all about equal performance (97% on English)

All make use of information sources:

- Via human created features: HMMs and CRFs
- Via representation learning: Neural LMs

Outline

- Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)
- 2 stochastic PoS Taggers
  - HMM
  - MEMM and CRF
- PoS tagging conclusion
- NER tagging

Outline

- **Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)**
- 2 stochastic PoS Taggers
  - HMM
  - MEMM and CRF
- PoS tagging conclusion
- NER tagging

RULES: JM^{3rd}, 8.7.1

While machine learned (neural or CRF) sequence models are the norm in academic research, commercial approaches to NER are often based on **pragmatic** combinations of lists and rules, with some smaller amount of supervised machine learning (Chiticariu et al., 2013). For example, in the IBM System T architecture, a user specifies declarative constraints for tagging tasks in a formal query language that includes regular expressions, dictionaries, semantic constraints, and other operators, which the system compiles into an efficient extractor (Chiticariu et al., 2018).

Rule-Based Tagging

Basic Idea:

1. Assign all possible tags to words (morphological analysis!)
2. Remove tags according to set of rules of type. For example:

   if word+1 is an adj, adv, or quantifier and the following is a sentence boundary and word-1 is not a verb like "consider"

   - **then** eliminate non-adv
   - **else** eliminate adv.

- Typically more than 1000 hand-written rules, but it may be machine-learned.

Sample ENGTWOL Lexicon

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	IMPERATIVE VFIN
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

Stage 1 of ENGTWOL Tagging

Run words through a morphological analyzer to get all parts of speech.

Example: *Pavlov had shown that salivation …*

Pavlov	PAVLOV N NOM SG PROPER
had	HAVE V PAST VFIN SVO
	HAVE PCP2 SVO
shown	SHOW PCP2 SVOO SVO SV
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	N NOM SG

Stage 2 of ENGTWOL Tagging

Apply constraints in a negative way

Example: Adverbial “that” rule -> Given input: “that”

```
IF (and
    (+1 A/ADV/QUANT)          /* if next word is adj, adverb, or quantifier */
    (+2 SENT-LIM)             /* and following which is a sentence boundary, */
    (NOT -1 SVOC/A) )         /* and the previous word is not a verb like
                                 ‘consider’ which allows adjs as object
                                  complements */
THEN
     eliminate non-ADV tags
ELSE
     eliminate ADV
```

Outline

- Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)
- **2 stochastic PoS Taggers**
  - **HMM**
  - MEMM and CRF
- PoS tagging conclusion
- NER tagging

POS Tagging as Sequence Labelling

- 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 $w_1 \ldots w_n$

Terminology

- **Modelling** -> *give a formal model*
- **Learning** -> *an algorithm for setting the parameters of the model*
- **Decoding** -> *algorithm for applying the model in order to compute results*

Modelling 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”

Modelling 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

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

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

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

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

Likelihood and Prior

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

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

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

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

Learning Two Kinds of Probabilities: PoS->PoS

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: **Learning**

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

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

Learning Two Kinds of Probabilities: PoS->word

Word likelihood probabilities $\mathbf{P(w_i|t_i)}$

- VBZ (3sg Pres Verb) likely to be “is”
- Compute P(is|VBZ) by counting in a labeled corpus: **Learning**

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

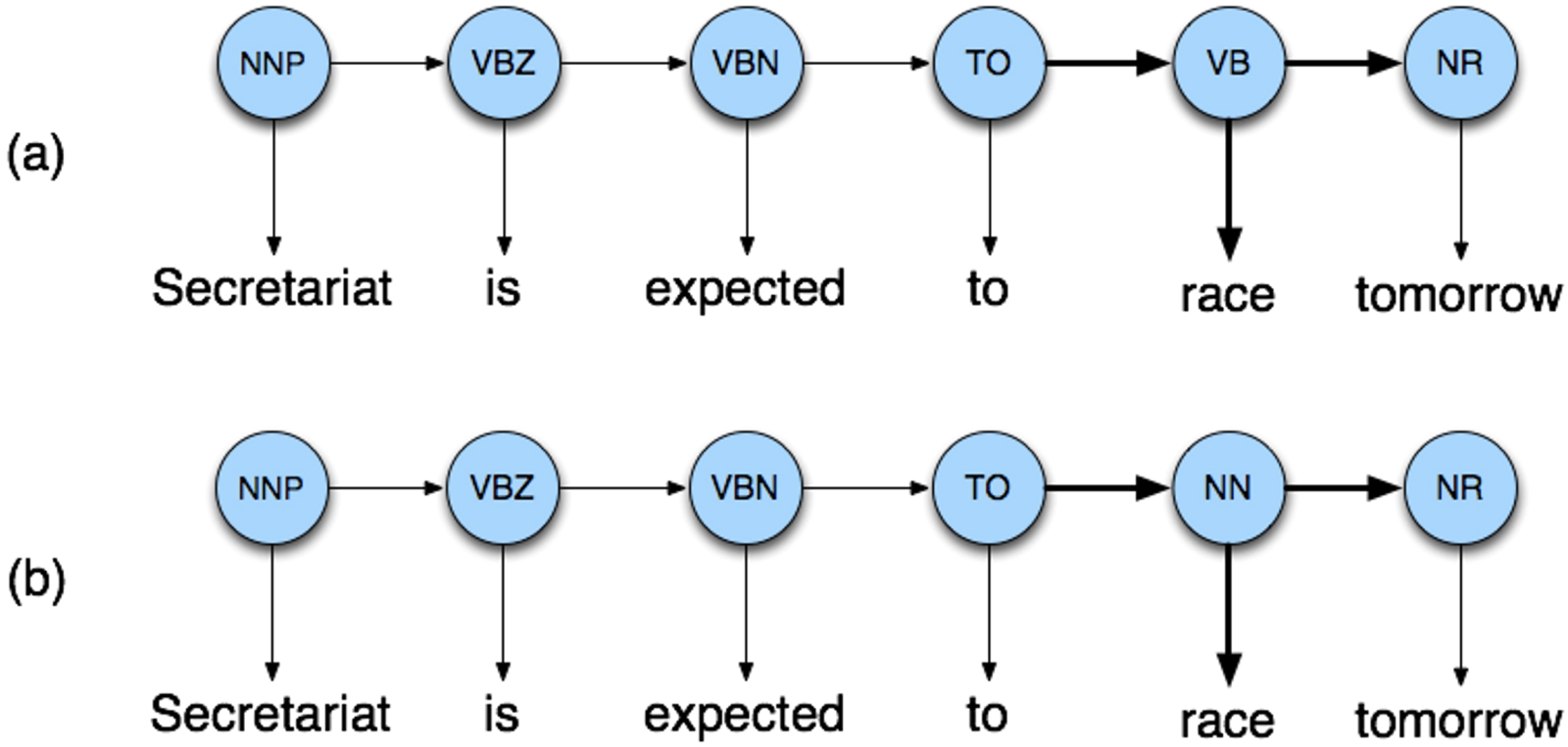
$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Example: The Verb “race”

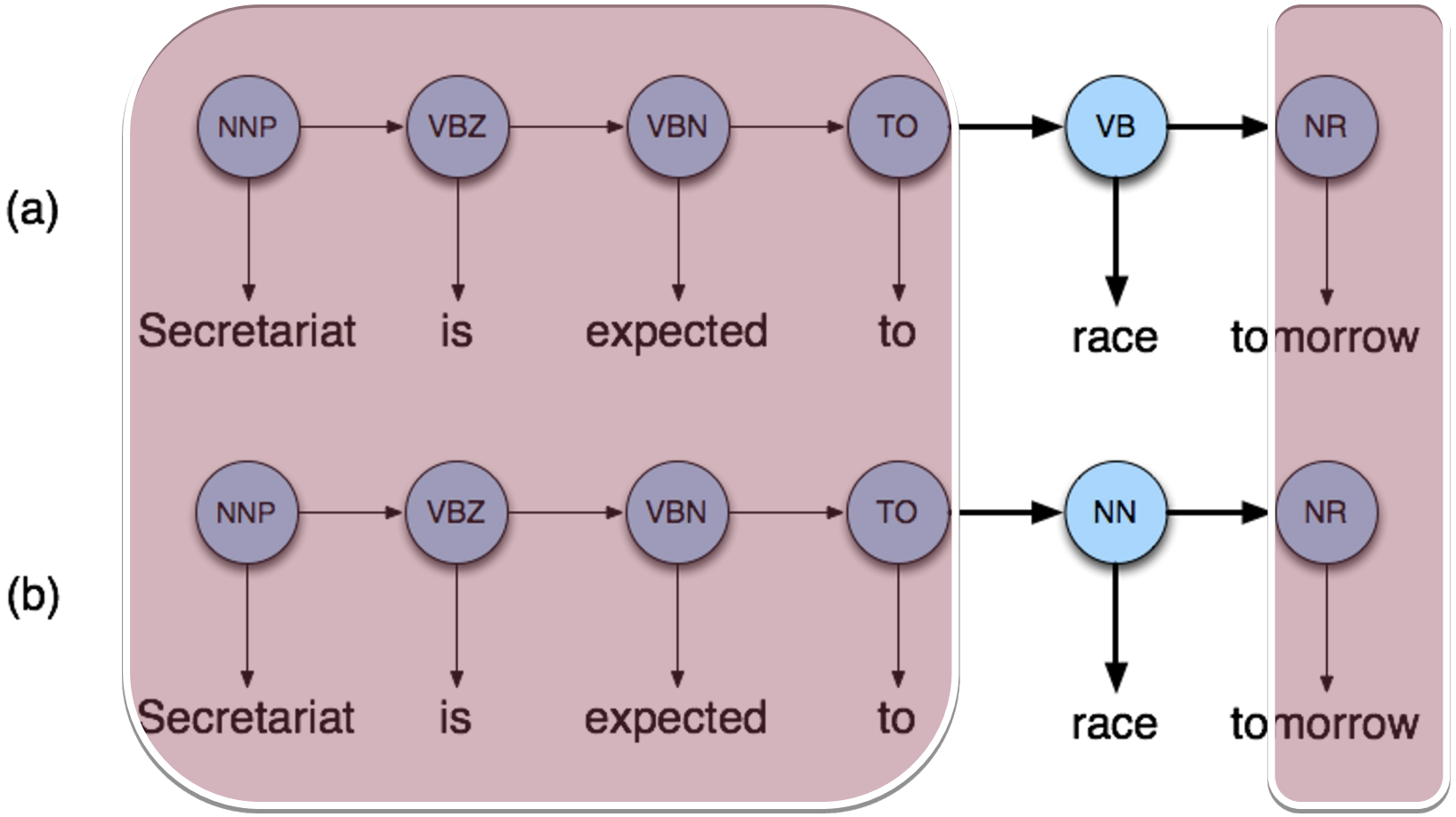
- Secretariat_NNP is_VBZ expected_VBN to_TO **race**_VB tomorrow_NR
- People_NNS continue_VB to_TO inquire_VB the_DT reason_NN for_IN the_DT **race**_NN for_IN outer_JJ space_NN

How do we pick the right tag?

Disambiguating “race”

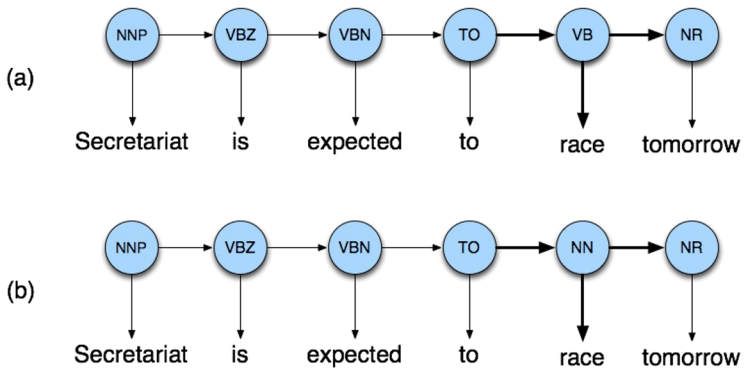


Disambiguating “race”



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”

Decoding HMMs

- Ok, now we have a complete model that can give us what we need. Recall that we need to get

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

- We could just enumerate all paths given the input and use the model to assign probabilities to each.

Example: Janet will back the bill

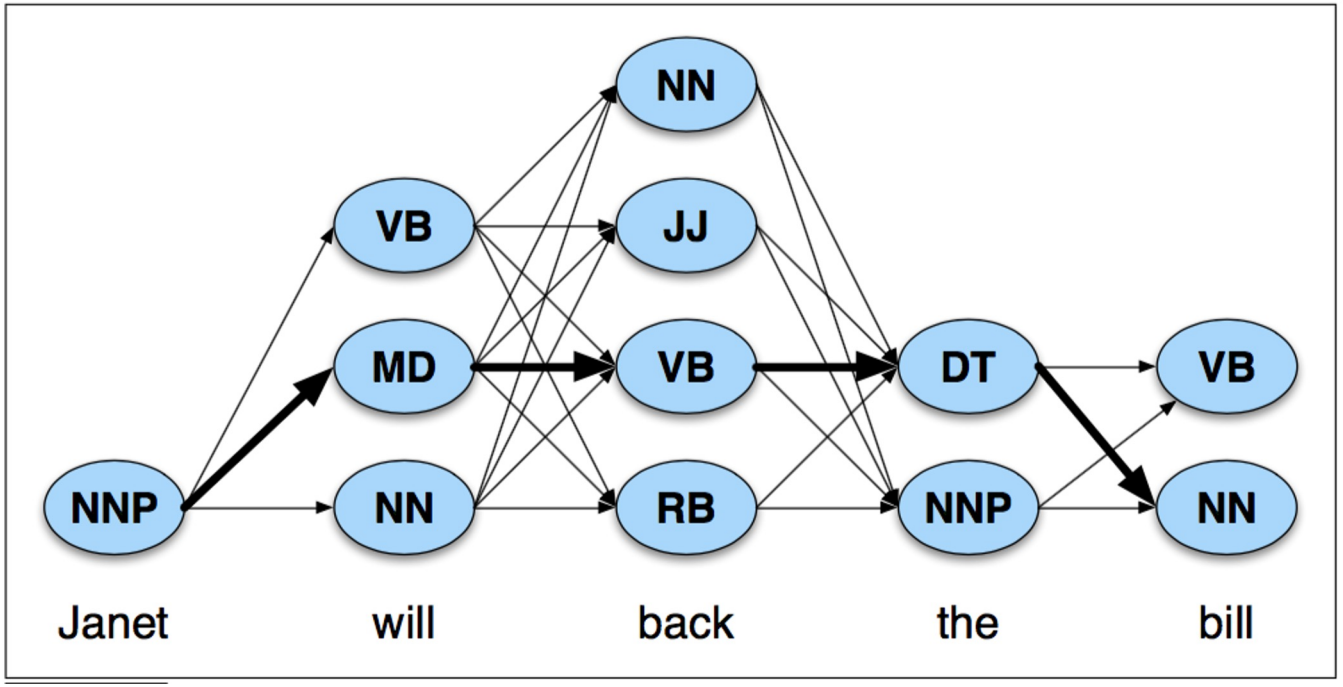
Janet/NNP will/MD back/VB the/DT bill/NN

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0.000097	0
NN	0	0.000200	0.000223	0.000006	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Example: Janet will back the bill

Janet/NNP will/MD back/VB the/DT bill/NN



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?

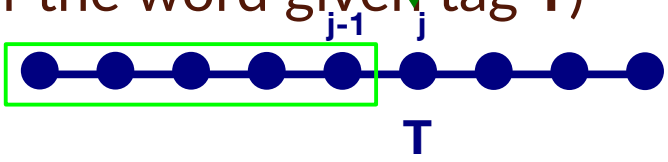
Question

- 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?

$$30^{20}$$

Dynamic Programming idea

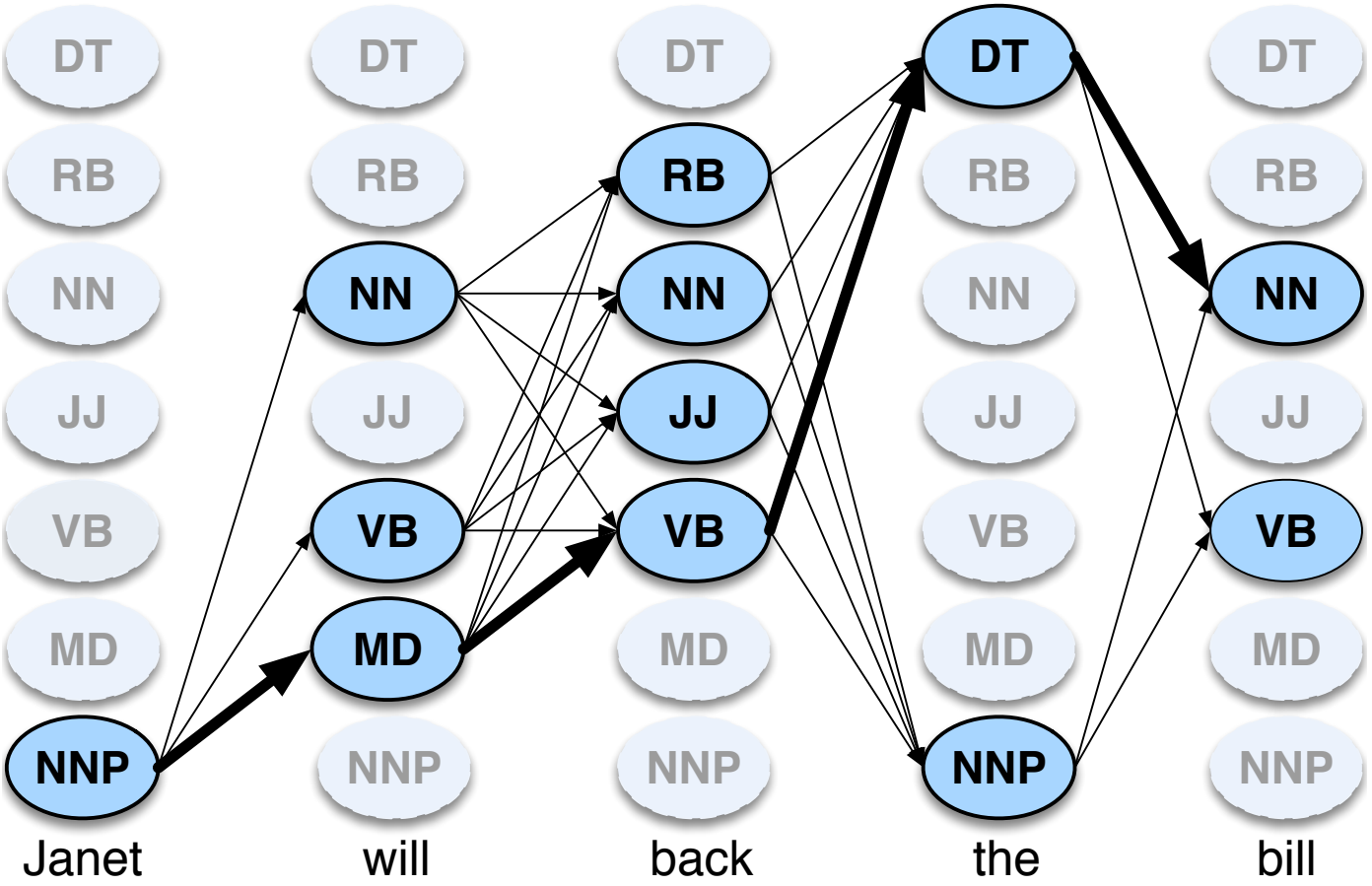
- Consider a state sequence (tag sequence) that ends at state **j** with a particular tag **T**.
- The probability of that tag sequence can be broken into two parts
  - The probability of the BEST tag sequence up through **j-1**
  - Multiplied by the transition probability from the tag at the end of the **j-1** sequence to **T** (and the observation probability of the word given tag **T**)



Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states -> (PoS_TAGS + S_{ini} + S_{fin})
- Sweep through the array in one pass filling the columns left to right using our transition probs and observations probs $$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$
- **Dynamic programming key is that we need only store the MAX prob path to each cell, and not all paths.**

Example



The Viterbi Matrix

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

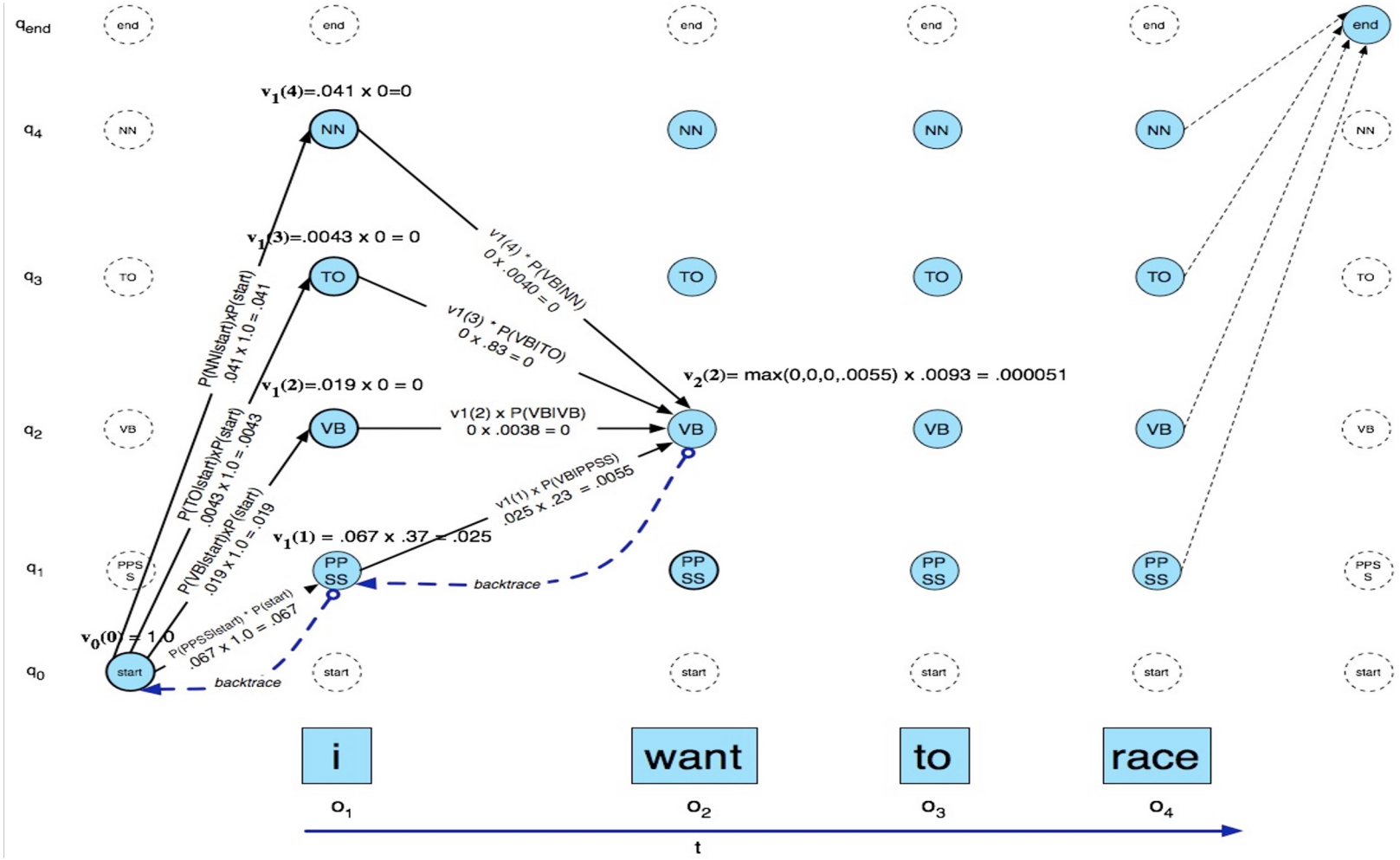
$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

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**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$$
$$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F} \quad \text{; termination step}$$
$$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F} \quad \text{; termination step}$$
return the backtrace path by following backpointers to states back in time from $backpointer[q_F,T]$

Example


q_{end}
q_4
q_3
q_2
q_1
q_0
end
NN
TO
VB
PPSS
start
$v_1(4)$=.041 x 0=0
$v_1(3)$=.0043 x 0 = 0
$v_1(2)$=.019 x 0 = 0
$v_1(1)$ = .067 x .37 = .025
$v_0(0)$ = 1.0
P(NN|start)xP(start) .041 x 1.0 = .041
P(TO|start)xP(start) .0043 x 1.0 = .0043
P(VB|start)xP(start) .019 x 1.0 = .019
P(PPSS|start) * P(start) .067 x 1.0 = .067
v1(4) * P(VB|NN) 0 x .0040 = 0
v1(3) * P(VB|TO) 0 x .83 = 0
v1(2) x P(VB|VB) 0 x .0038 = 0
v1(1) x P(VB|PPSS) .025 x .23 = .0055
$v_2(2)$= max(0,0,0,.0055) x .0093 = .000051
backtrace
backtrace
i
want
to
race
o_1
o_2
o_3
o_4
t

Example

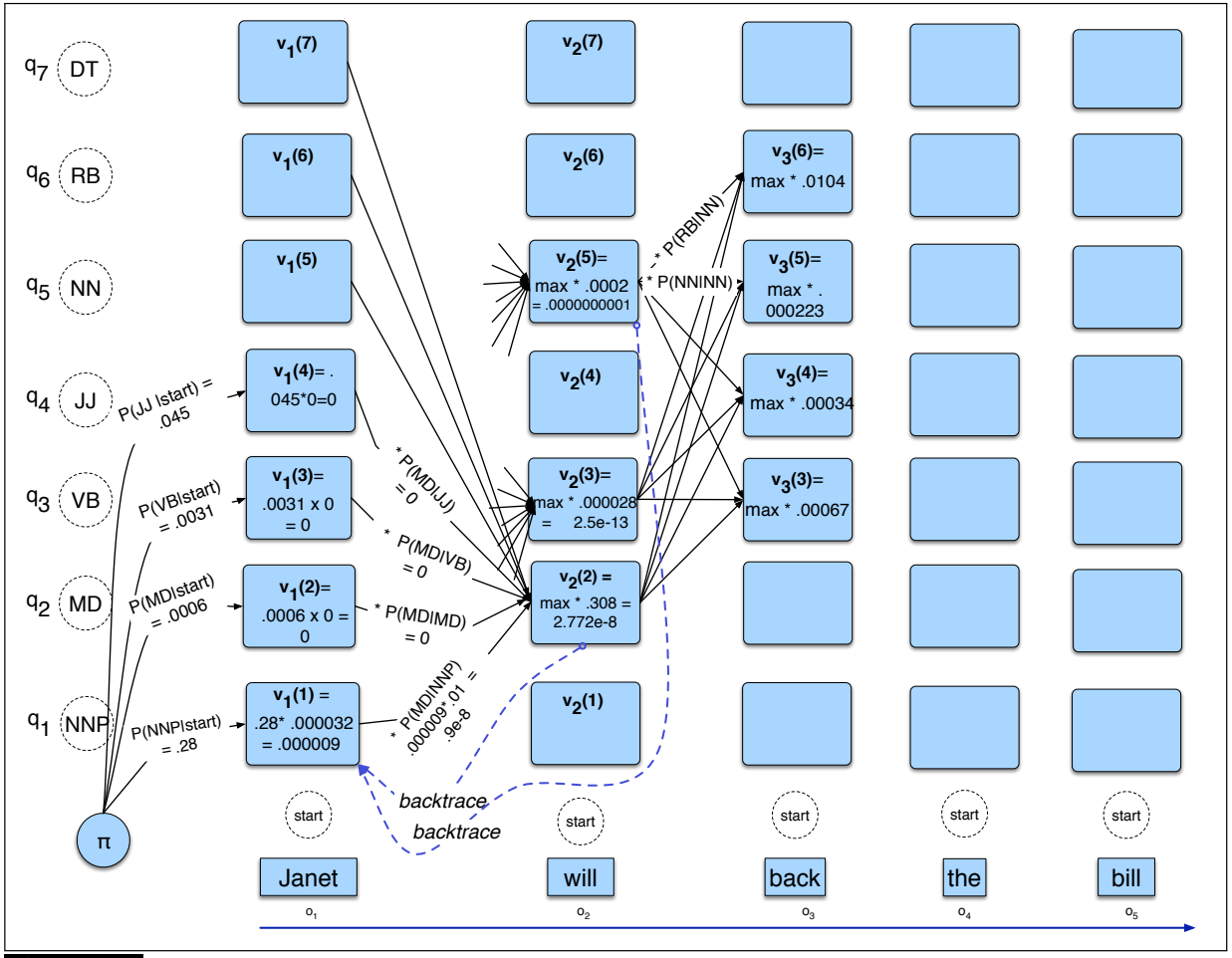


Figure 8.14 The first few entries in the individual state columns for the Viterbi algorithm. Each cell keeps the probability of the best path so far and a pointer to the previous cell along that path. We have only filled out columns 1 and 2; to avoid clutter most cells with value 0 are left empty. The rest is left as an exercise for the reader. After the cells are filled in, backtracing from the *end* state, we should be able to reconstruct the correct state sequence NNP MD VB DT NN.

Viterbi Summary

- **Dynamic programming key is that we need only store the MAX prob path to each cell, and not all paths.**
- “Life can only be understood backwards, but must be lived forwards”

Esercizio su HMM

- Fare il learning di un modello HMM sul corpus:
  - Paolo/N pesca/V
  - Giovanni/N ama/V i/D cani/N
  - Francesca/N ama/N
  - Una/D pesca/N Francesca/A
- Fare il decoding, facendo girare a mano l'algoritmo di Viterbi, della frase:
  - **Paolo ama Francesca**

Extending the HMM Algorithm to Trigrams

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

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}, t_{i-2})$$

Extending the HMM Algorithm to Trigrams

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1}, t_{i-2}) \right] P(t_{n+1}|t_n)$$

$$P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

Extending the HMM Algorithm to Trigrams

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(w_i|t_i) P(t_i|t_{i-1}, t_{i-2}) \right] P(t_{n+1}|t_n)$$

$$P(t_i|t_{i-1}, t_{i-2}) = \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})}$$

Sparsness!!!

Sparsity of data -> combining n-grams

$$\text{Trigrams} \quad \hat{P}(t_i|t_{i-1},t_{i-2}) = \frac{C(t_{i-2},t_{i-1},t_i)}{C(t_{i-2},t_{i-1})}$$

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

$$\text{Unigrams} \quad \hat{P}(t_i) = \frac{C(t_i)}{N}$$

$$P(t_i|t_{i-1}t_{i-2}) = \lambda_3\hat{P}(t_i|t_{i-1}t_{i-2}) + \lambda_2\hat{P}(t_i|t_{i-1}) + \lambda_1\hat{P}(t_i)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

Lambdas -> deleted interpolation algorithm

function DELETED-INTERPOLATION(*corpus*) **returns** $\lambda_1, \lambda_2, \lambda_3$

$\lambda_1 \leftarrow 0$
$\lambda_2 \leftarrow 0$
$\lambda_3 \leftarrow 0$
foreach trigram t_1, t_2, t_3 with $C(t_1, t_2, t_3) > 0$
  depending on the maximum of the following three values
    case $\frac{C(t_1,t_2,t_3)-1}{C(t_1,t_2)-1}$: increment λ_3 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_2,t_3)-1}{C(t_2)-1}$: increment λ_2 by $C(t_1, t_2, t_3)$
    case $\frac{C(t_3)-1}{N-1}$: increment λ_1 by $C(t_1, t_2, t_3)$
  end
end
normalize $\lambda_1, \lambda_2, \lambda_3$
return $\lambda_1, \lambda_2, \lambda_3$

Problems of HMM

- No **Rich Feature Information**, which are required
  - When x_k is complex
  - When data of x_k is sparse
- POS Tagging
  - How to evaluate $P(w_k|t_k)$ for unknown words w_k ?
  - **Useful features**
    - Suffix, e.g., -ed, -tion, -ing, etc.
    - Capitalization

Outline

- Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)
- **2 stochastic PoS Taggers**
  - HMM
  - **MEMM and CRF**
- PoS tagging conclusion
- NER tagging

HMM vs. MEMM (MAXIMUM ENTROPY MARKOV MODELS)

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T P(W|T)P(T) \\ &= \operatorname*{argmax}_T \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})\end{aligned}$$

$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T \prod_i P(t_i|w_i, t_{i-1})\end{aligned}$$

Generative vs. Discriminative (Ch.8 Intro, Ch.8 3rd ed, Oct. 2019 https://web.stanford.edu/~jurafsky/slp3/old_dec20/)

... generative model: a model that is trained to generate the data x from the class y. The likelihood term P(x|y) expresses that we are given the class y and are trying to predict which features we expect to see in the input x.

A discriminative model takes this direct approach, computing P(y|x) by discriminating among the different possible values of the class y rather than first computing a likelihood.

cats vs. dogs

Maximum Entropy modeling

- We may identify a more **heterogeneous set of features** which contribute in some way to the choice of the current word.
- Maxent combines these features in a **probabilistic model.** The given features provide a constraint on the model.
- We would like to have a probability distribution which, outside of these constraints, is **as uniform as possible** – has the **maximum entropy** among all models that satisfy these constraints.

MEMM Modelling

$$P(y|x) \quad ?= \quad \sum_{i=1}^{N} w_i f_i$$

$$?= \quad w \cdot f$$

$$p(y = c|x) = p(c|x) \quad = \quad \frac{1}{Z} \exp \sum_i w_i f_i$$

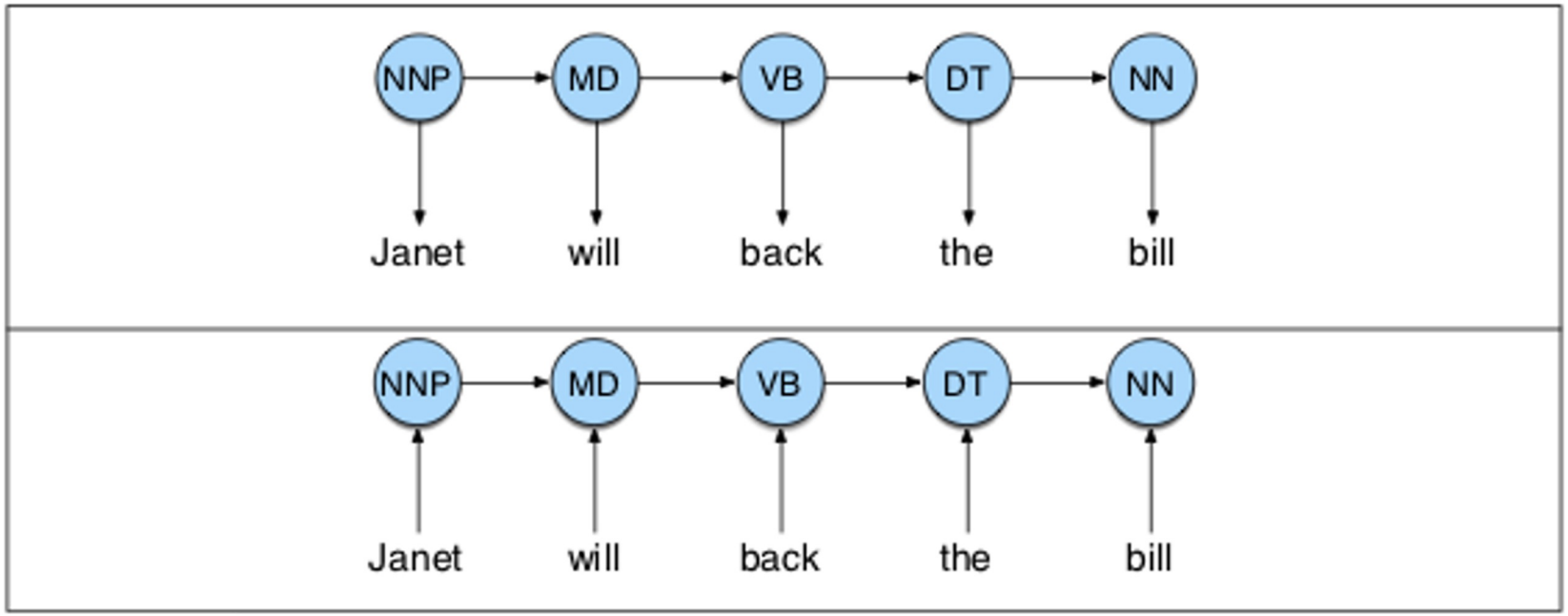
MEMM Modelling: 2 hypotheses

$$
\begin{aligned}
\hat{T} &= \operatorname*{argmax}_T P(T|W) \\
&= \operatorname*{argmax}_T \prod_i P(t_i|w_{i-l}^{i+l}, t_{i-k}^{i-1}) \\
&= \operatorname*{argmax}_T \prod_i \frac{\exp\left(\sum_i w_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in \text{tagset}} \exp\left(\sum_i w_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}
\end{aligned}
$$

1. t_i is conditionally independent of all previous observations and labels given x_t and t_{i-1}. (i.e. in the formula -> l=0, k=0)
2. the observations x_t are independent of one another.

https://awni.github.io/label-bias/

HMM vs. MEMM (MAXIMUM ENTROPY MARKOV MODELS)



Maxent P(tag|word)

- Features on a word by itself (no memory!):
  - Word the: the DT
  - Prefixes unfathomable: un- JJ
  - Suffixes Importantly: -ly RB
  - Cap. Meridian: CAP NNP
  - Shapes 35-year: d-x JJ
- Then build a classifier to predict tag:
  - Maxent P(tag|word): 93.7% overall / 82.6% unknown
  - Very good!

More features

- w_i contains a particular prefix (from all prefixes of length ≤ 4)
- w_i contains a particular suffix (from all suffixes of length ≤ 4)
- w_i contains a number
- w_i contains an upper-case letter wi contains a hyphen
- w_i is all upper case
- w_i's word shape
- w_i's short word shape
- w_i is upper case and has a digit and a dash (like CFC-12)
- w_i is upper case and followed within 3 words by Co., Inc., etc.

MEMM Learning

- MaxEnt learning -> Multinomial Logistic Regression

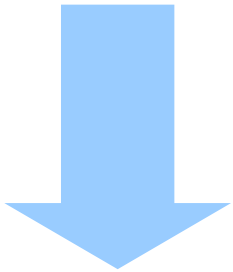
  Learning $$\hat{w} = \underset{w}{\operatorname{argmax}} \sum_{j} \log P(y^{(j)}|x^{(j)})$$

- **Conditional maximum likelihood estimation:** we choose the parameters w that maximize the (log) probability of the y labels in the training data given the observations x.

- Convex optimization problem, so we use hill-climbing methods like stochastic gradient ascent, L-BFGS (Nocedal 1980, Byrd et al. 1995)

MEMM Decoding: Viterbi!

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$



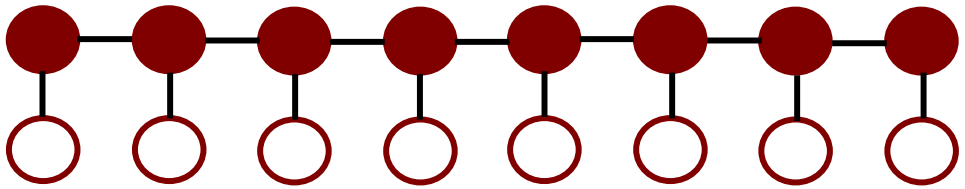
$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, P(s_j|s_i, o_t) \quad 1 \le j \le N, 1 < t \le T$$

MEMM Summary

- Modelling P(y|x) only, have no generative process
  - can use arbitrary features
  - can use for classification, or choosing from n-best list (reranking)
- Learning involves iteratively updating the weights, so typically: 1. slower than for generative models, 2. random behaviour
- Decoding -> Greedy/Viterbi

Conditional Random Fields

- **CRF**s: Globally-normalized discriminative sequence labeling models!
  - A family of graphical models where the probability of a configuration of variables is proportional to a product of scores across pairs of variables (suitable for structured prediction, not just sequence labeling).
- In NLP, usually we mean a **linear-chain CRF** where pairs of variables are labels for adjacent tokens.
- When labeling Y_i future observations are taken into account



http://demo.clab.cs.cmu.edu/11711fa20/slides/11711-08-ner-crfs.pdf

CRF - Jurafsky

- In a CRF, we're dealing with a sequence, so the function F maps an **entire** input sequence X and an **entire** output sequence Y to a feature vector.
- K functions $F_k(X,Y)$ global features, since each one is a property of the entire input sequence X and output sequence Y

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y')\right)}$$

$$F_k(X,Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

…. depend on the current and previous output tokens y_i and y_{i-1} are what characterizes a **linear chain** CRF.

CRF - features

$\langle y_i, x_i \rangle, \langle y_i, y_{i-1} \rangle, \langle y_i, x_{i-1}, x_{i+2} \rangle$

f_{3743}: y_i = VB and x_i = back
f_{156}: y_i = VB and y_{i-1} = MD
f_{99732}: y_i = VB and x_{i-1} = will and x_{i+2} = bill

x_i contains a particular prefix (perhaps from all prefixes of length ≤ 2)
x_i contains a particular suffix (perhaps from all suffixes of length ≤ 2)
x_i's word shape
x_i's short word shape

Remember that in a CRF we don't learn weights for each of these local features f_k. Instead, we first sum the values of each local feature (for example feature f_{3743}) over the entire sentence, to create each global feature (for example F_{3743}). It is those global features that will then be multiplied by weight w_{3743}. Thus for **training** and **inference** there is always a fixed set of K features with K weights, even though the length of each sentence is different.

Learning -> stochastic gradient descent

CRF – Decoding: Viterbi

$$
\begin{aligned}
\hat{Y} &= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} P(Y|X) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \frac{1}{Z(X)} \exp\left(\sum_{k=1}^{K} w_k F_k(X,Y)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \exp\left(\sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)\right) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{k=1}^{K} w_k \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i) \\
&= \underset{Y \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i=1}^{n} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i)
\end{aligned}
$$

$$
v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T
$$

$$
v_t(j) = \max_{i=1}^{N} v_{t-1}(i) \sum_{k=1}^{K} w_k f_k(y_{t-1}, y_t, X, t) \quad 1 \le j \le N, 1 < t \le T
$$

Outline

- Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)
- 2 stochastic PoS Taggers
  - HMM
  - MEMM and CRF
- **PoS tagging conclusion**
- NER tagging

Tagging Unknown Words (P=0)

- New words added to (newspaper) language 20+ per month
- Plus many proper names …
- **Increases error rates by 1-2%**

Possible approaches for Unknown Words:

- *Method 1*: assume they are nouns
- *Method 2*: *Assume uniform distribution over PoS*
- *Method 3*: *External dictionary* (e.g. morph-it)
- *Method 4*: Use morphological information, e.g., words ending with –ed tend to be tagged VBN
- *Method 5*: assume the unknown words have a probability distribution similar to words only occurring once in the training/developing set.

Evaluation

- The result is compared with a manually coded "Gold Standard"
- Gold standard = Corpus tagged by-hand
- Usually:
  - Training data: 80%
  - Development data: 10%
  - Test data: 10%

Outline

- Rule-Based PoS Tagger: ENGTWOL (ENGlish TWO Level analysis)
- 2 stochastic PoS Taggers
  - HMM
  - MEMM and CRF
- PoS tagging conclusion
- **NER tagging**

Named Entities

Named entity, in its core usage, means anything that can be referred to with a **proper name**. Most common 4 tags:

- PER (Person): “Marie Curie”
- LOC (Location): “New York City”
- ORG (Organization): “Stanford University”
- GPE (Geo-Political Entity): "Boulder, Colorado"

- Often multi-word phrases
- But the term is also extended to things that aren't entities:
  - dates, times, prices

Named Entity tagging

The task of named entity recognition (NER) ->

2 subtasks:

- find spans of text that constitute proper names
- tag the type of the entity.

NER output

Citing high fuel prices, [$_{ORG}$ **United Airlines**] said [$_{TIME}$ **Friday**] it has increased fares by [$_{MONEY}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{ORG}$ **American Airlines**], a unit of [$_{ORG}$ **AMR Corp.**], immediately matched the move, spokesman [$_{PER}$ **Tim Wagner**] said. [$_{ORG}$ **United**], a unit of [$_{ORG}$ **UAL Corp.**], said the increase took effect [$_{TIME}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{LOC}$ **Chicago**] to [$_{LOC}$ **Dallas**] and [$_{LOC}$ **Denver**] to [$_{LOC}$ **San Francisco**].

Why NER?

- **Sentiment analysis**: consumer's sentiment toward a particular company or person?
- **Question Answering**: answer questions about an entity?
- **Information Extraction**: Extracting facts about entities from text.

Why NER is hard(er than PoS)

1) Segmentation: *spans of text*
   - In PoS tagging, no segmentation problem since each word gets one tag.
   - In NER we have to find and segment the entities!

2) Type ambiguity

[PER Washington] was born into slavery on the farm of James Burroughs.
[ORG Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [LOC Washington] for what may well be his last state visit.
In June, [GPE Washington] passed a primary seatbelt law.

BIO Tagging

- How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

- [PER Jane Villanueva] of [ORG United Airlines Holding] , discussed the [LOC Chicago] route.

BIO Tagging

- [PER Jane Villanueva] of [ORG United Airlines Holding] , discussed the [LOC Chicago] route.

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

Now we have one tag per token!!!

BIO Tagging

- B: token that *begins* a span
- I: tokens *inside* a span
- O: tokens outside of any span

# of tags (where n is #entity types):

- 1 **O** tag,
- *n* **B** tags,
- *n* **I** tags

total of ***2n+1***

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

Standard algorithms for NER

Supervised Machine Learning given a human-labeled training set of text annotated with tags

- Hidden Markov Models
- Maximum Entropy Markov Models/Conditional Random Fields
- Neural sequence models (RNNs or Transformers)
- Large Language Models (like BERT), finetuned

Conclusions

- The analysis lexical-morphological level can be approched with different algorithms
- All these algorithms model analysis as **sequential learning** producing in output a **sequence** of tags