Natural Language Processing

Class 05: Part-of-speech tagging

- Syntactic word categories
- POS tagsets
- Sequential tagging algorithms
- Evaluation

VRIJE UNIVERSITEIT BRUSSEL

16 March 2018

Katrien Beuls

Parts of Speech

Aristotle (384-322 BCE) already had the idea of having parts of speech

- lexical categories, word classes, “tags”, POS

Dionysius Thrax of Alexandria (c. 100 BCE) suggested that there are 8 parts of speech

- Thrax: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
- School grammar: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection

Parts of Speech

Based on

- syntactic function
- morphological function

Words that function similarly w.r.t.

- what can occur nearby
- the affixes they take

Grouped into classes

Two broad super categories

Closed class types:

- Relatively fixed membership (e.g. prepositions)
- Generally function words

Four open class types:

- **nouns**: proper vs. common (count vs. mass)
- **verbs**
- **adjectives**
- **adverbs**
  - *Unfortunately*, John walked *home extremely slowly yesterday*

Closed classes differ more from language to language

prepositions: on, under, over, near, by, at, from, to, with
determiners: a, an, the
pronouns: she, who, I, others
conjunctions: and, but, or, as, if, when
auxiliary verbs: can, may, should, are
particles: up, down, on, off, in, out, at, by
numerals: one, two, three, first, second, third

interjections, negatives, politeness markers, greetings, existential there, …

Penn Treebank Part-of-Speech Tagset

- 45 tags
- Used to label a variety of corpora: Brown corpus, Wall Street Journal corpus, Switchboard corpus
- Tags are typically placed after each word, delimited by a slash

(9.1) The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

(9.2) **There/EX** are/VBP 70/CD children/NNS **there/RB**

(9.3) Preliminary/JJ findings/NNS were/VBD **reported/VBN** in/IN today/NN **'s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	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, sing.	*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*			

Figure 9.1 Penn Treebank part-of-speech tags (including punctuation).

Corpora labeled with POS are crucial training and testing sets for statistical tagging algorithms

Brown corpus: 1 million words (500 written texts from different genres, USA, 1961)

WSJ corpus: 1 million words (newspaper, 1989)

Switchboard corpus: 2 million words (telephone conversations, 1990-1991)

Created by running an automatic POS tagger on the texts and correcting each tag by hand

Tagging algorithms assume tokenization

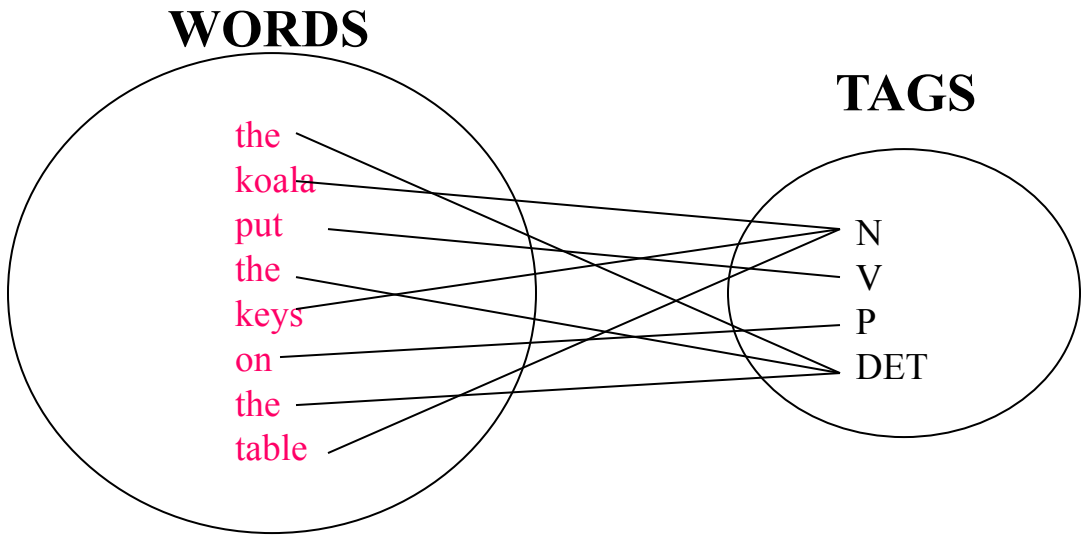
Contractions and ‘s-genitives split from their stems:

would/MD n’t/RB
children/NNS ’s/POS

Multipart words: separate words vs one word

- *a/DT New/NNP York/NNP City/NNP firm/NN* (Penn)
- *in/II31 terms/II32 of/II33* (C5 tagset BNC)

Objectives



Assigning a POS marker to each word in an input text

- Input = sequence of words + tagset
- Output = sequence of tags

Disambiguation task

- *book that flight* vs. *hand me that book*
- *does that flight serve dinner* vs. *I thought that your flight was earlier*

Applications for POS tagging

Speech synthesis pronunciation

Lead	*Lead*
INsult	*inSULT*
OBject	*obJECT*
OVERflow	*overFLOW*
DIScount	*disCOUNT*

Parsing: e.g. *Time flies like an arrow*

- Is *flies* an N or V?

Word prediction in speech recognition

- Possessive pronouns (*my, your, her*) are likely to be followed by nouns
- Personal pronouns (*I, you, he*) are likely to be followed by verbs

Machine Translation

How hard is the tagging problem?

		WSJ		Brown	
Types:					
Unambiguous	(1 tag)	44,432	(**86%**)	45,799	(**85%**)
Ambiguous	(2+ tags)	7,025	(**14%**)	8,050	(**15%**)
Tokens:					
Unambiguous	(1 tag)	577,421	(**45%**)	384,349	(**33%**)
Ambiguous	(2+ tags)	711,780	(**55%**)	786,646	(**67%**)

Figure 9.2 The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case.

Some of the most ambiguous frequent words are *that, back, down, put* or *set*

earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** about debt
I was twenty-one **back/RB** then

Even ambiguous tokens are easy to disambiguate

Why?

Different tags associated with a word are not equally likely

Simplistic baseline algorithm: choose most frequent tag in case of ambiguity

Most Frequent Class Baseline: Always compare a classifier against a baseline at least as good as the most frequent class baseline (assigning each token to the class it occurred in most often in the training set).

Accuracy (sections 22-24 WSJ corpus): **92.34%**

- **State of the art = 97%**

Sources of information

The word itself

- Some words may only be nouns, e.g. arrow
- Some words are ambiguous, e.g. like, flies
- Probabilities may help, if one tag is more likely than another

Local context

- Two determiners rarely follow each other
- Two base form verbs rarely follow each other
- Determiner is almost always followed by adjective or noun

Some ways to do POS Tagging

Rule-based tagging

- E.g. EnCG ENGTWOL tagger

Transformation-based tagging

- Learned rules (statistic and linguistic)
- E.g., Brill tagger

Stochastic, or, Probabilistic tagging

- HMM (Hidden Markov Model) tagging
- or more recently MEMM tagging

POS tagging

- **Rule-based tagging**
- HMM tagging
- MEMM tagging

Rule-based tagging

Use dictionary to assign each word a list of potential tags

Large list of hand-written disambiguation rules

One part of speech per word

Different rule sets for different tag sets

Different rule sets for different languages

Start with a dictionary

she	PRP
promised	VBN, VBD
to	TO
back	VB, JJ, RB, NN
the	DT
bill	NN, VB

Assign every possible tag

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	*promised*	*to*	*back*	*the*	*bill*

Write rules to eliminate tags

Eliminate VBN if VBD is an option when VBN|VBD follows "<start> PRP"

…

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	*promised*	*to*	*back*	*the*	*bill*

EngCG Tagger (Voutilainen)

Rules based on English Constraint Grammar

Two stage design

Uses ENGTWOL Lexicon

Hand written disambiguation rules

A. Voutilainen, *Morphological disambiguation*, in Karlsson, Voutilainen, Heikkila, Anttila (eds) Constraint Grammar pp165-284, Mouton de Gruyter, 1995. See [e-book]

ENGTWOL Lexicon

Based on TWO-Level morphology of English

56,000 entries for English word stems

Each entry annotated with morphological and syntactic features

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

Actual constraint syntax

ADVERBIAL-THAT RULE
Given input: "that"
if
(+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 tag

Eliminates all readings of "that" except the one in
"It isn't that odd"
"You can't go that far"

ENGCG Tagger

Stage 1: Run words through morphological analyzer to get all parts of speech

- E.g. for the phrase “the tables”, we get the following output:

```
"<the>" "the"
 <Def> DET CENTRAL ART SG/PL
"<tables>" "table"
 N NOM PL "table"
 <SVO> V PRES SG3 VFIN
```

Stage 2: Apply constraints to rule out incorrect POSs

Performance

Tested on examples from Wall St Journal, Brown Corpus, Lancaster-Oslo-Bergen Corpus

After application of the rules 93-97% of all words are fully disambiguated, and 99.7% of all words retain correct reading

At the time, this was superior performance to other taggers

However, one should not discount the amount of effort needed to create this system

POS tagging

- Rule-based tagging
- **HMM tagging**
- MEMM tagging

HMM tagging

Training on fully labeled dataset, setting HMM parameters by maximum likelihood estimates on this training set

Viterbi algorithm for decoding

Goal = choose tag sequence that is most probable given the observation sequence of *n* words

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

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

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

Two further simplifying assumptions

Probability of a word appearing depends only on its own tag and is independent of neighboring tags:

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

Bigram assumption: probability of a tag is dependent only on the previous tag (rather than entire tag sequence):

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

Emission and transition probabilities

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

Estimating probabilities

MLE of a transition probability is computing by counting (out of the times we see the first tag in a labeled corpus) how often the first tag is followed by the second:

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

$$P(VB|MD) = \frac{C(MD, VB)}{C(MD)} = \frac{10471}{13124} = .80$$

Estimating probabilities

The emission probabilities represent the probability given a tag that it will be associated with a given word

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

$$P(will|MD) = \frac{C(MD, will)}{C(MD)} = \frac{4046}{13124} = .31$$

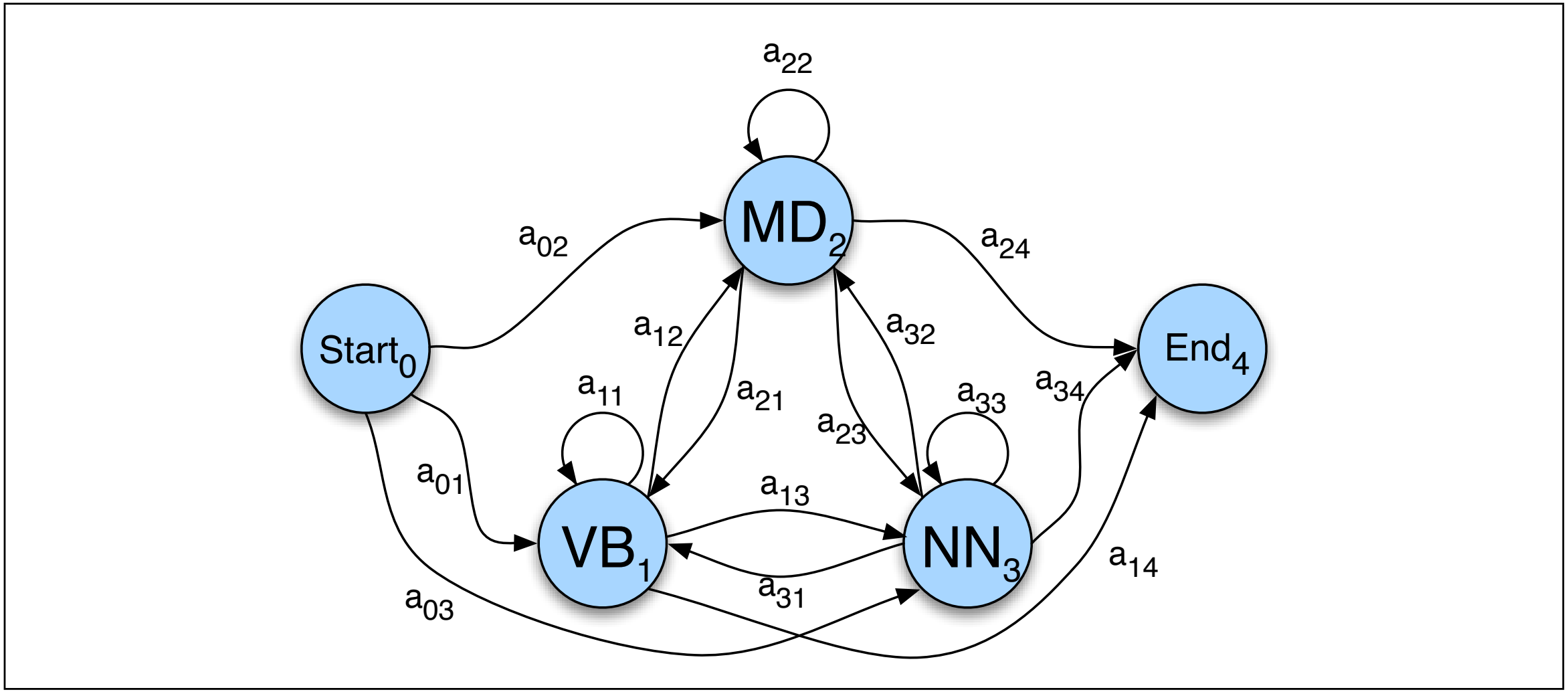


Figure 9.3 A piee of the Markov chain corresponding to the hidden states of the HMM. The *A* transition probabilities are used to compute the prior probability.

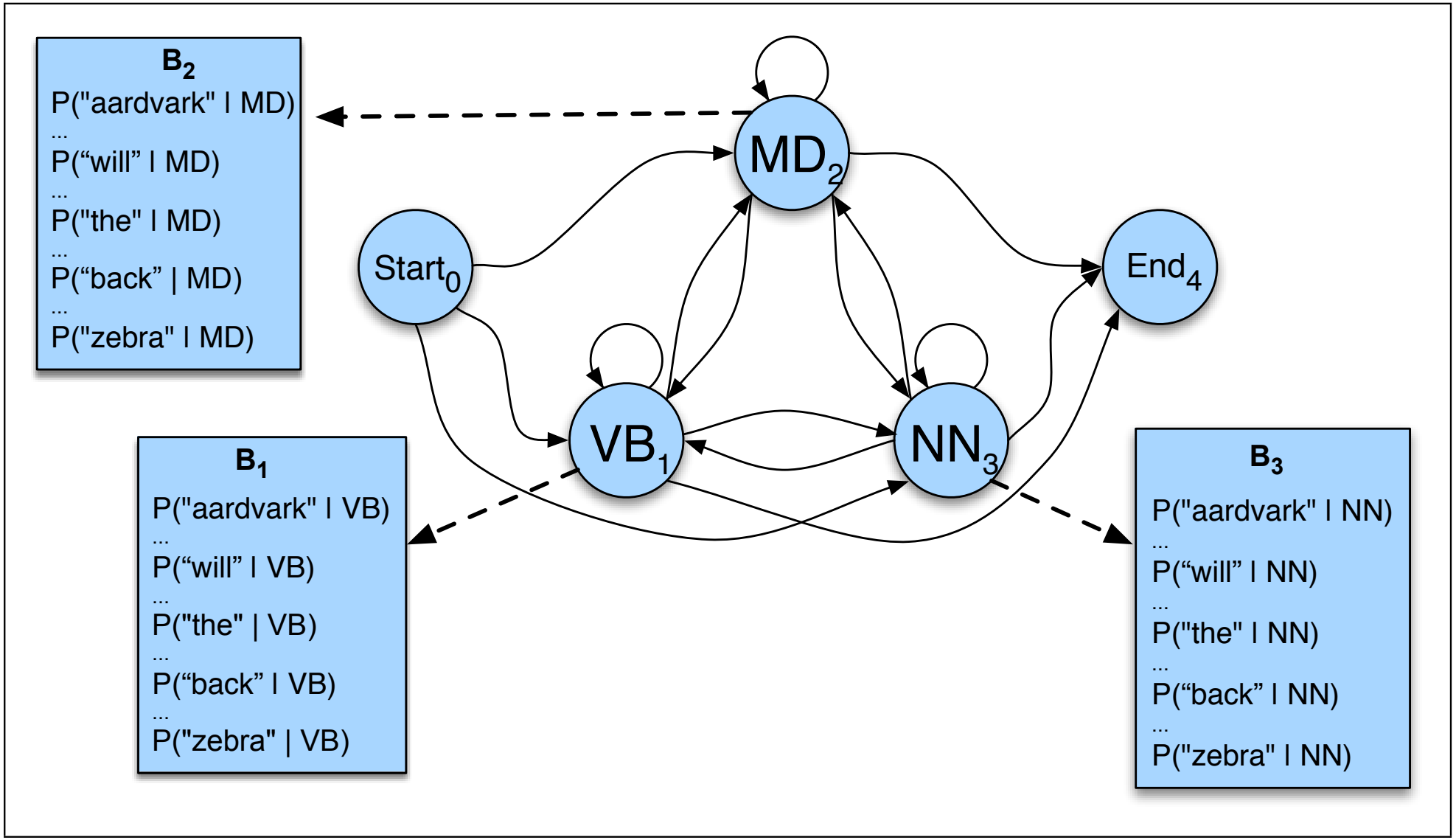


Figure 9.4 Some of the *B* observation likelihoods for the HMM in the previous figure. Each state (except the non-emitting start and end states) is associated with a vector of probabilities, one likelihood for each possible observation word.

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

See transition probabilities and observation likelihoods from WSJ corpus in Figs 9.5 and 9.6

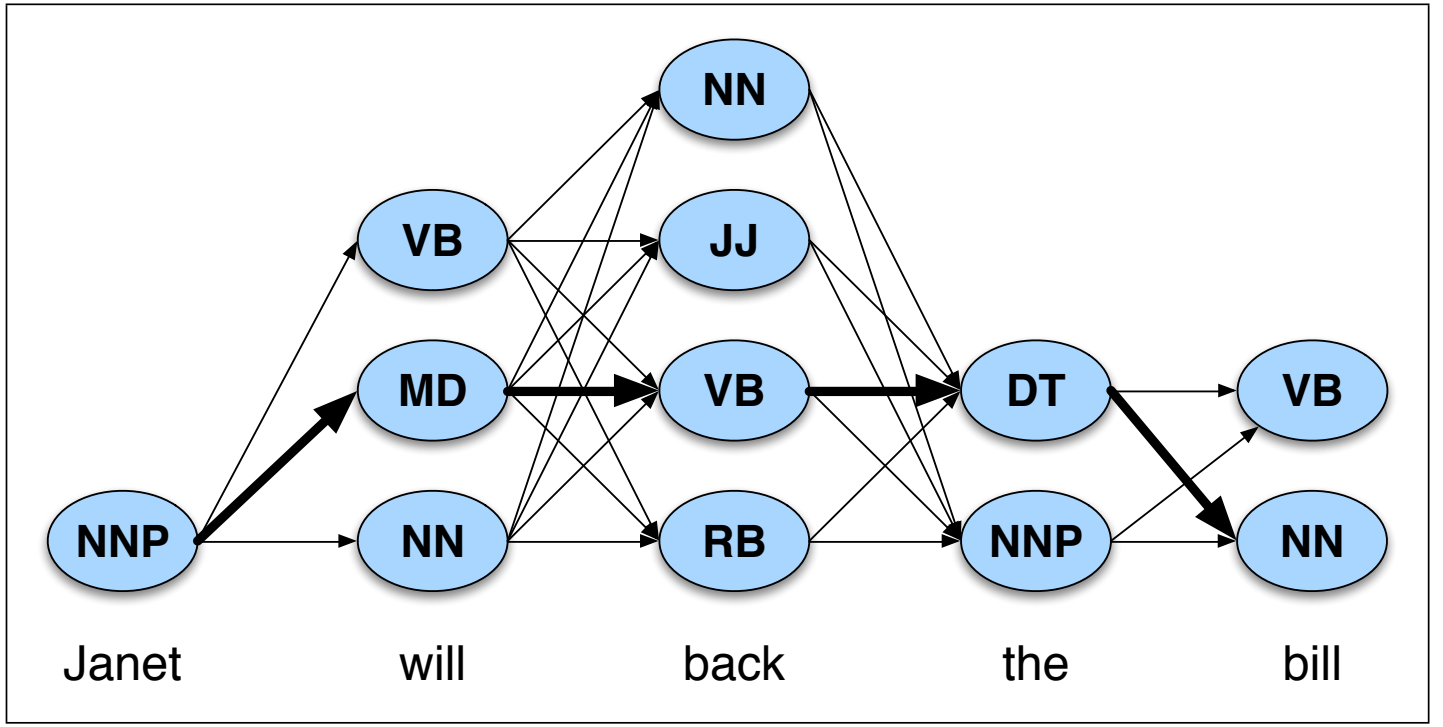


Figure 9.7 A schematic of the tagging task for the sample sentence, showing the ambiguities for each word and the correct tag sequence as the highlighted path through the hidden states.

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

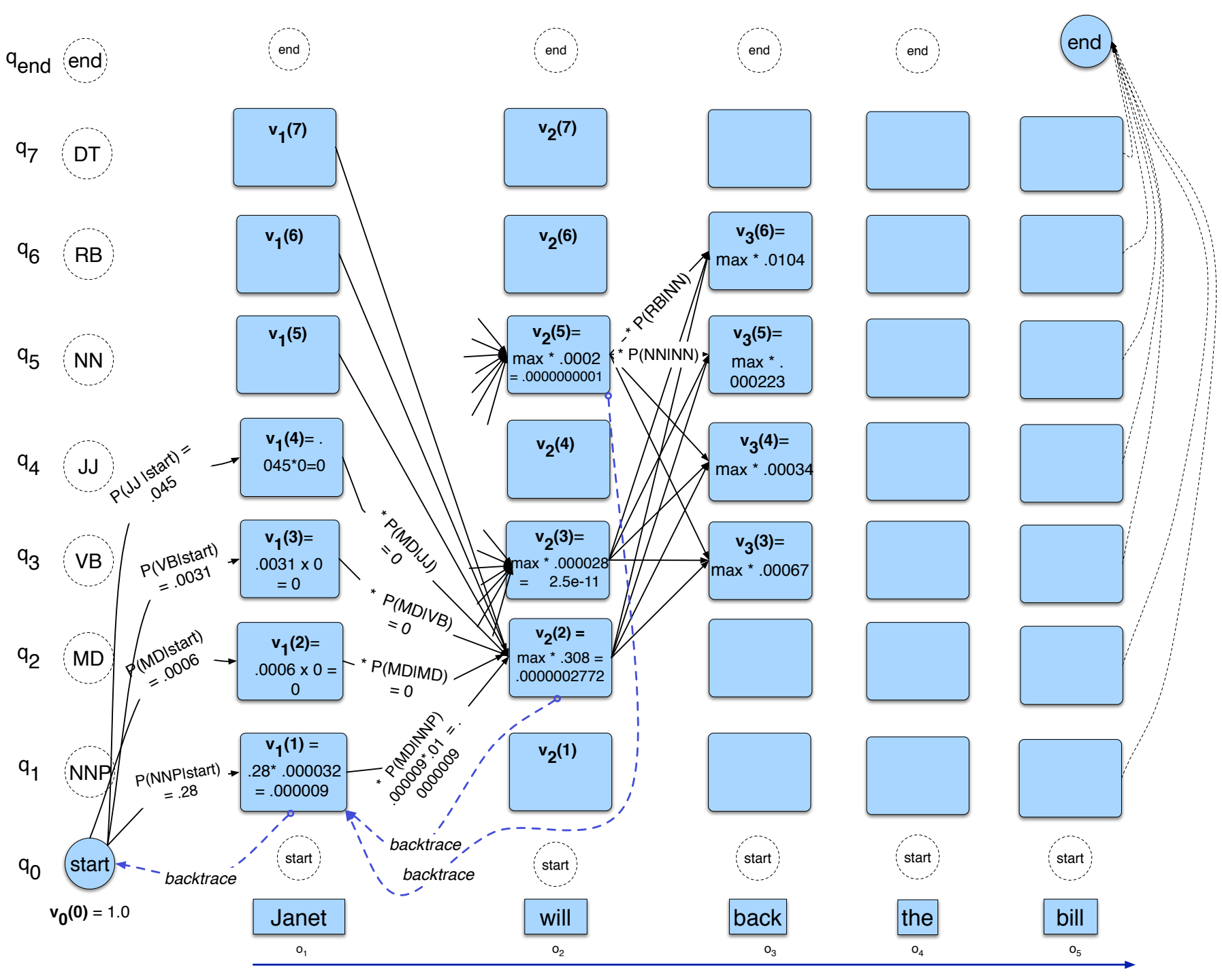
create a path probability matrix *viterbi[N+2,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$viterbi[s,1] \leftarrow a_{0,s} * b_s(o_1)$
$backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**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

Figure 9.8 Viterbi algorithm for finding optimal sequence of tags. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence. Note that states 0 and q_F are non-emitting.

Viterbi probability

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

$v_{t-1}(i)$	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


q_end end
q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
q0 start
v0(0) = 1.0
v1(7)
v1(6)
v1(5)
v1(4)= . 045*0=0
v1(3)= .0031 x 0 = 0
v1(2)= .0006 x 0 = 0
v1(1) = .28* .000032 = .000009
v2(7)
v2(6)
v2(5)= max * .0002 = .0000000001
v2(4)
v2(3)= max * .000028 = 2.5e-11
v2(2) = max * .308 = .0000002772
v2(1)
v3(6)= max * .0104
v3(5)= max * . 000223
v3(4)= max * .00034
v3(3)= max * .00067
P(JJ|start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) = .000009*.01 = .0000009
* P(RB|NN)
* P(NN|NN)
backtrace
backtrace
backtrace
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

Unknown words

Useful feature for distinguishing POS is word shape

Strongest source of information is morphology

- -s, -ed, -able, …
- Store for each suffix the statistics of which tag they were associated with in training:

$$P(t_i | l_{n-i+1} \ldots l_n)$$ = posterior estimate

Unknown words are unlikely to be closed-class words

- Compute suffix probabilities only for words whose frequency in the training set is < 10, or only for open-class words

POS tagging

- Rule-based tagging
- HMM tagging
- **MEMM tagging**

Maximum Entropy Markov Models

Sequence model adaptation of the MaxEnt (multinomial logistic regression) classifier

Discriminative sequence model

HMM

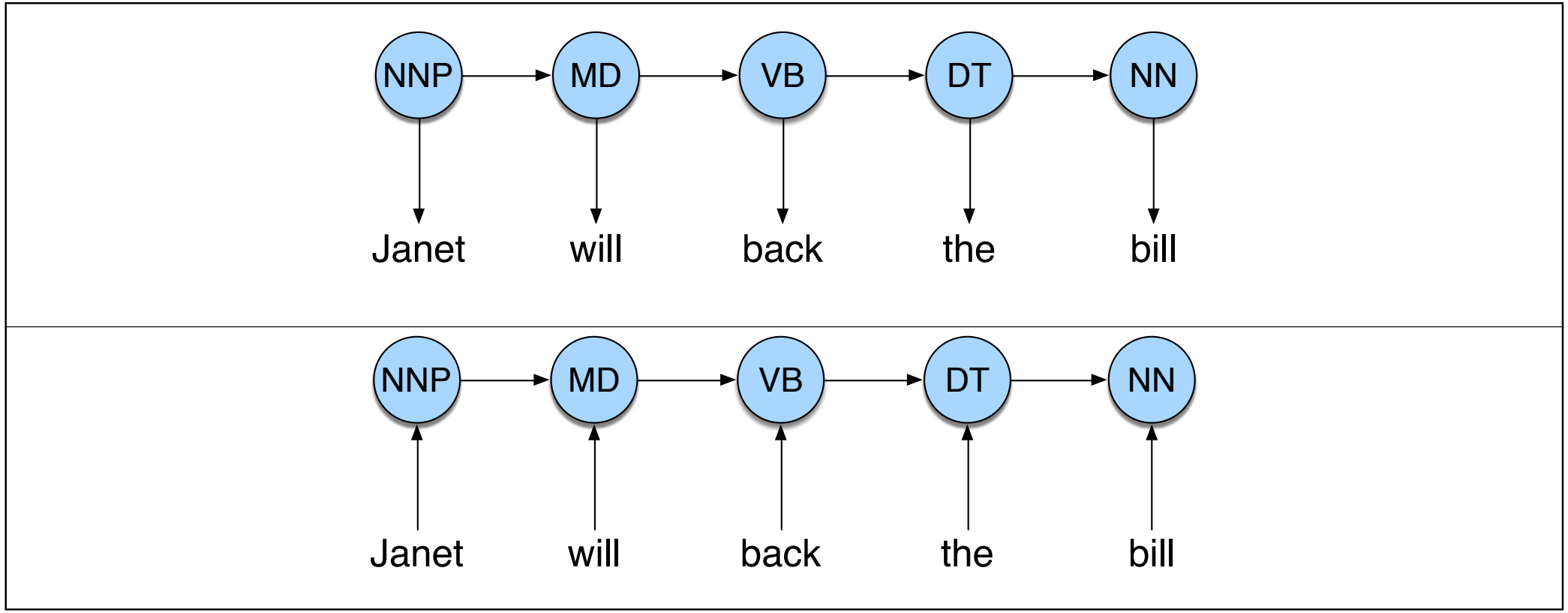
$$\begin{aligned}\hat{T} &= \operatorname*{argmax}_T P(T|W) \\ &= \operatorname*{argmax}_T P(W|T)P(T) \\ &= \operatorname*{argmax}_T \prod_i P(word_i|tag_i) \prod_i P(tag_i|tag_{i-1})\end{aligned}$$

MEMM

directly compute the posterior!

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

Comparing HMM (top) and MEMM (bottom)



Using more than two features!

Using just two features (observed word and previous tag) is not more accurate than generative HMM model

Discriminative sequence models make it easier to incorporate more features!

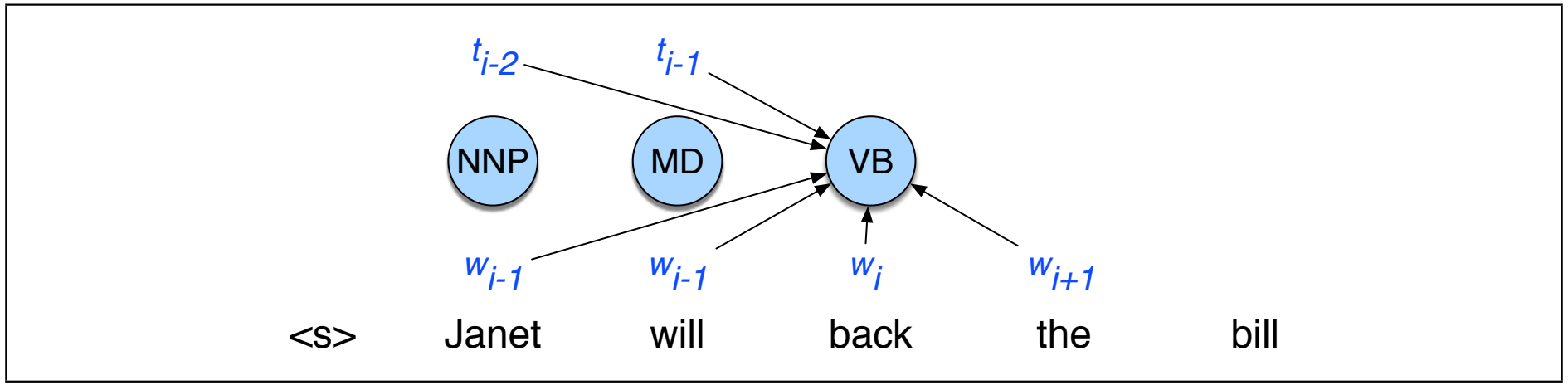


Figure 9.12 An MEMM for part-of-speech tagging showing the ability to condition on more features.

Feature templates for MEMMs

Used to automatically populate the set of features from every instance in the training and test set

$$\langle t_i, w_{i-2} \rangle, \langle t_i, w_{i1} \rangle, \langle t_i, w_i \rangle, \langle t_i, w_{i+1} \rangle, \langle t_i, w_{i+2} \rangle$$
$$\langle t_i, t_{i-1} \rangle, \langle t_i, t_{i-2}, t_{i-1} \rangle,$$
$$\langle t_i, t_{i1}, w_i \rangle, \langle t_i, w_{i1}, w_i \rangle \langle t_i, w_i, w_{i+1} \rangle,$$

Generating features for *back*

$t_i = \text{VB}$ and $w_{i-2} = \text{Janet}$
$t_i = \text{VB}$ and $w_{i-1} = \text{will}$
$t_i = \text{VB}$ and $w_i = \text{back}$
$t_i = \text{VB}$ and $w_{i+1} = \text{the}$
$t_i = \text{VB}$ and $w_{i+2} = \text{bill}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$
$t_i = \text{VB}$ and $t_{i-1} = \text{MD}$ and $t_{i-2} = \text{NNP}$
$t_i = \text{VB}$ and $w_i = \text{back}$ and $w_{i+1} = \text{the}$

Features to deal with unknown words

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
w_i 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.

Decoding and training MEMMs

Simplest way to turn MaxEnt classifier into a sequence model:

- Build a local classifier that classifies each word left to right
- = Greedy decoding algorithm

function GREEDY MEMM DECODING(words W, model P) **returns** tag sequence T

for $i = 1$ **to** *length(W)*

$$\hat{t}_i = \underset{t' \in T}{\text{argmax}}\ P(t' \mid w_{i-l}^{i+l}, t_{i-k}^{i-1})$$

Figure 9.13 In greedy decoding we make a hard decision to choose the best tag left to right.

Viterbi again

Greedy algorithm is fast but future decisions have no effect on current decision

Instead decode MEMM with the Viterbi algorithm:

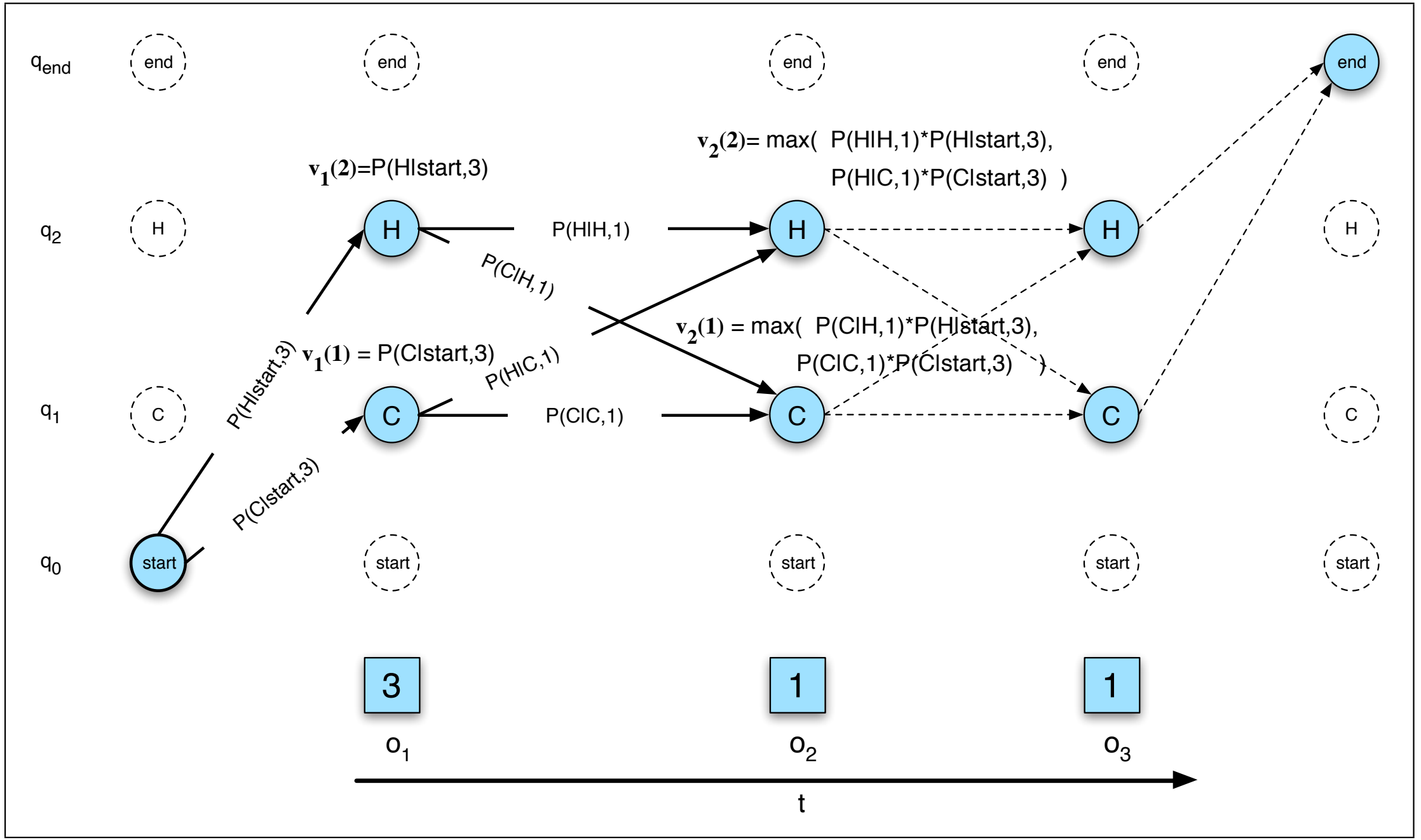
- Finding the sequence of POS tags that is optimal for the **whole sentence**

Only difference to HMM: how we fill each cell

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

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

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


q_{end}
q_2
q_1
q_0
end
H
C
start
$v_1(2)$=P(HIstart,3)
$v_1(1)$ = P(CIstart,3)
P(HIstart,3)
P(CIstart,3)
P(HIH,1)
P(CIH,1)
P(HIC,1)
P(CIC,1)
$v_2(2)$= max(P(HIH,1)*P(HIstart,3),
P(HIC,1)*P(CIstart,3))
$v_2(1)$ = max(P(CIH,1)*P(HIstart,3),
P(CIC,1)*P(CIstart,3))
3
1
1
o_1
o_2
o_3
t

Bidirectionality

Problem: Exclusively run left-to-right

- "Label bias": *will/**NN** to/TO fight/VB*
  - BUT! Previous transition prefers modal and emission probability of will being a modal is close to 1

Switch to a much more powerful model: Conditional Random Fields

Alternatively, use multiple passes or run tagger in two directions

- Greedy decoding: choose highest-scoring of the tag assigned by left-to-right and right-to-left classifier
- Viterbi decoding: choose higher scoring of two sequences

POS tagging

- **Evaluation**
- Advanced topics

Evaluation

So once you have you POS tagger running how do you evaluate it?

- Overall error rate with respect to a gold-standard test set.
- Error rates on particular tags
- Error rates on particular words
- Tag confusions...

The result is compared with a manually coded “Gold Standard”

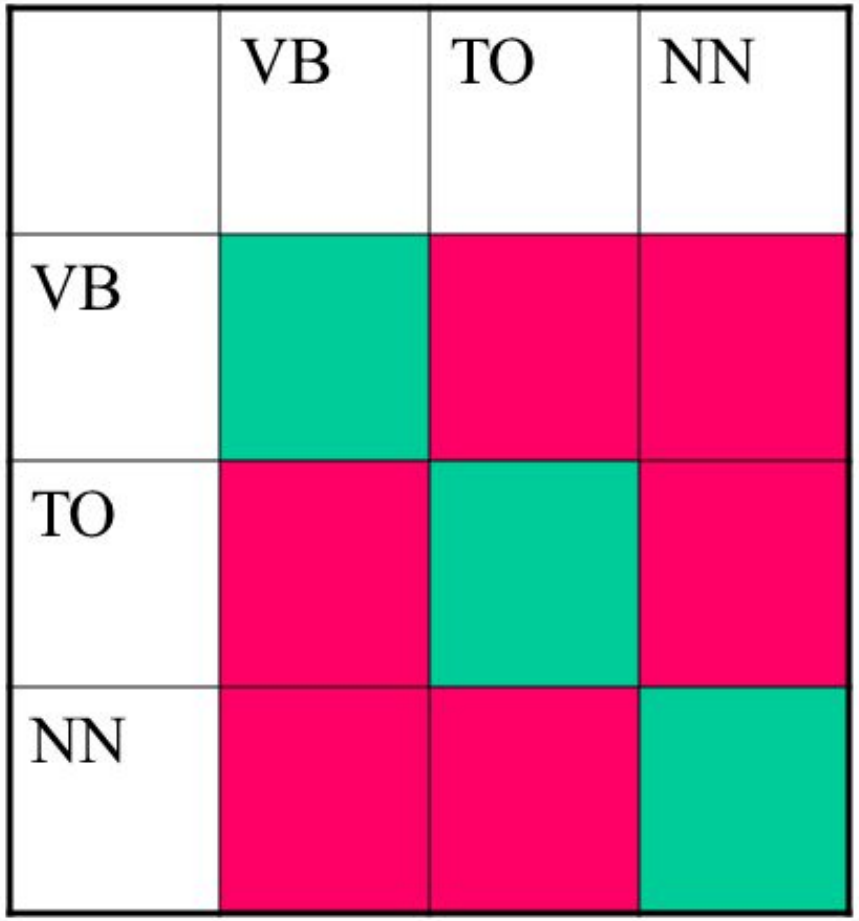
- Typically accuracy reaches 96-97%
- This may be compared with result for a baseline tagger (one that uses no context).

Important: 100% is impossible even for human annotators.

Confusion matrix

Which tags did we most often confuse with which other tags?

How much of the overall error does each confusion account for?



POS tagging

- Evaluation
- **Advanced topics**

POS tagging for other languages

Tagger accuracies for German are close to those for English

Augmentations become necessary when dealing with highly inflected or agglutinative languages such as Czech, Hungarian and Turkish

- Large vocabulary: therefore many unknown words
- More information coded in word morphology (needs to be labelled)
  - Tags are sequences morphological tags

Combination tagger

Each of the methods has its strong and weak points

Combining outputs from different taggers

- Meta-tagger: train machine learning tool on
  - Words
  - tags from first order taggers
- Voting
  - majority voting
  - probabilistic voting

Other tagging tasks

A number of problems can be framed as tagging problems

- Case restoration: If we just get lowercased text, we may want to restore proper casing: e.g. *the river Thames*
- Accent restoration: When keyboards lack the proper keys, it is common to not write the accents in Spanish or French.
- Named entity recognition: it may also be useful to find names of persons, organisations, etc. in the text, e.g. *Barack Obama*
- BaseNP chunking: for text processing purposes it is useful to detect base noun phrases that correspond to concepts, e.g. *department of defense*

BaseNP chunking

Task: find basic noun phrases
(facilitates parsing, information extraction)

Example: [*the student*] *said* [*the exam question*] *is hard*

Three tags

- B = beginning of baseNP
- I = continuing baseNP (internal)
- O = other word

Example: *the*/B *student*/I *said*/O *the*/B *exam*/I *question*/I *is*/O *hard*/O

Tagging task: assign tags (B, I, O) to each word

Summary

First step towards syntactic analysis

Hidden-Markov Models, Maximum Entropy Markov Models

Supervised accuracy

- In-domain: >97%
- Out-of-domain: <90%
- Low-resource languages
- Sentence accuracy: 55-57%

The Stanford Natural Language Processing Group

http://nlp.stanford.edu/software/tagger.shtml

people publications research blog software teaching local

Software > Stanford Log-linear Part-Of-Speech Tagger

Stanford Log-linear Part-Of-Speech Tagger

About | Questions | Mailing lists | Download | Extensions | Release history | FAQ

About

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. This software is a Java implementation of the log-linear part-of-speech taggers described in these papers (if citing just one paper, cite the 2003 one):

> Kristina Toutanova and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pp. 63-70.

Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?

Christopher D. Manning

Departments of Linguistics and Computer Science
Stanford University
353 Serra Mall, Stanford CA 94305-9010
manning@stanford.edu

Abstract. I examine what would be necessary to move part-of-speech tagging performance from its current level of about 97.3% token accuracy (56% sentence accuracy) to close to 100% accuracy. I suggest that it must still be possible to greatly increase tagging performance and examine some useful improvements that have recently been made to the Stanford Part-of-Speech Tagger. However, an error analysis of some of the remaining errors suggests that there is limited further mileage to be had either from better machine learning or better features in a discriminative sequence classifier. The prospects for further gains from semi-supervised learning also seem quite limited. Rather, I suggest and begin

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics? *Conference on Intelligent Text Processing and Computational Linguistics*

Homework:

- Experiment with NLTK POS taggers (NLT handbook ch.5: http://www.nltk.org/book/ch05.html)
- Section 4: try the default tagger and some alternatives on the Brown corpus
- Evaluate their performance

Next week:

- Dependency parsing (Jurafsky & Martin 3rd Ed. ch.14)