POS Tagging

Chapter 5 & 6 of J&M'09
Chapter 9 & 10 of M&S'00

POS Tagging

Part-of-speech tagging

(1) a. Bill$_{NP}$ bought$_V$ a$_D$ record$_N$.

b. Bill$_{NP}$ will$_V$ record$_V$ a$_D$ song$_N$.

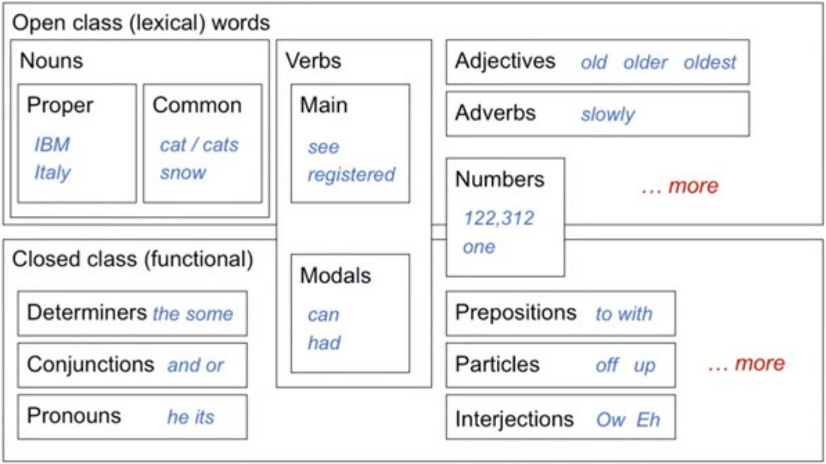
vs.

(2) a. Bill$_V$ bought$_V$ a$_D$ record$_V$.

b. Bill$_V$ will$_N$ record$_N$ a$_D$ song$_N$.

- **Closed class** (function words)
  - Pronouns: *you, she, her, me, I, they, ...*
    - Possessive: *his, her, my, our, their, its, mine, ours, ...*
    - wh-pronouns: *who, what, which, when, whom, whoever,...*
  - Prepositions: *in, under, to, by, from, about, ...*
  - Determiners: *the, a, an, each, every, some, ...*
  - Conjunctions
    - Coordinating: *and, or, but, as, ...*
    - Subordinating: *that, when, who, if, because, ...*
  - Particles: *up, down, off, over, on, ...*
  - Numerals: *one, two, three, ..., first, second, third,...*
  - Auxiliary verbs: *can, may, should, are, ...*
- **Open class** (content words)
  - Nouns
    - Proper nouns: *Tom, Mia, France, Jupiter, ...*
    - Common nouns
      - Count nouns: *cat, table, dream, emotion, height, sea, ...*
      - Non-count nouns: *milk, mail, music, heat, cash, fun*
  - Verbs: *read, eat, paint, think, say, fall, ...*
  - Adjectives: *green, good, false, painted, ...*
  - Adverbs: *quickly, yesterday, very, often, twice, never, not, ...*

POS Tagging


Open class (lexical) words
Nouns
Proper
IBM
Italy
Common
cat / cats
snow
Verbs
Main
see
registered
Modals
can
had
Adjectives old older oldest
Adverbs slowly
Numbers
122,312
one
… more
Closed class (functional)
Determiners the some
Conjunctions and or
Pronouns he its
Prepositions to with
Particles off up
Interjections Ow Eh
… more

Ambiguity abounds

(3) a. She knows you like the back of her hand.

b. This animal has four legs and flies.

c. I fed her baby carrots.

d. The dove dove.

e. I can't close the door, and the bear is getting too close.

f. I saw that gas can explode.

g. We ran out.

h. I said you should fire said person.

POS Tagging

Tagsets used in corpora (NLTK corpora)

- Brown corpus (tagset 87)
- Penn Treebank (tagset 36)
- BNC (tagset 65)
- COCA (159 tags)

– 11.5% of word types in the Brown corpus are ambiguous
i.e. only 11.5% of the word types can have more than one tag.

– 40% of Brown tokens are ambiguous.
i.e. 40% of occurring tokens can have more than one tag.

The word types in the 11.5% tend to be frequent:

(4) a. I know **that/IN** he is honest.
b. Yes, **that/DT** play was nice.
c. You can't go **that/RB** far.

POS Tagging

Some POS decisions are difficult even for humans:
(Penn Treebank tagset)

(5) a. Mrs/NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG

b. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN

c. Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CS

RP = particle
IN = preposition
RB = adverb

POS Tagging

Ambiguity

(6) Bill saw her father's bike yesterday.

In the Penn Treebank tagset, we get the following possibilities:

`Bill = NN / NNP / VP`
`saw = NN / VB / VBD`
`her = PRP$ / PRP`
`father = NN / VB`
`'s = POS / VBZ`
`bike = NN / VB`
`yesterday = NN / RB`

So, there is a total of $3^2 \times 2^5 = 288$ possible ways to tag (6), of which only one is correct: `Bill/NNP saw/VBD her/PRP$ father/NN 's/POS bike/NN yesterday/RB`

POS Tagging

Tagging methods & accuracies

- Rule-based POS Tagging (90% to 50%)
- Probability-based (Trigram HMM) (95% to 55%)
- Maxent $P(t|w)$ (93.7% to 82.6%)
- TnT (HMM++) (96.2% to 86.9%)
- MEMM tagger (a log-linear model) (96.9% to 86.9%)
- Bidirectional dependencies (Stanford parser) (97.2% to 90%)
- Upper bound: 98% (human agreement)

Note:

Current part-of-speech taggers work rapidly and reliably, with token accuracies of about 97%. But when it comes to sentence accuracy, current taggers have about 57% accuracy.
Source: Manning 2011 and J&M'17.

POS Tagging

The simplest method:

1. Assign the most frequent tag to ambiguous words.
2. Assign the 'noun' label to all unknown words.

90% accuracy on known words
50% accuracy on unknown words

POS Tagging

Rule-based tagging

1. create list of words with their most likely parts of speech (such lists of words are sometimes called a **lexicon**)
2. for each word of a sentence, tag it by look up in the lexicon the most frequent tag.

   E.g. `rat/NN` and not `rat/VB`
3. To correct errors, the tagger applies tag-changing rules.
   - Contextual Rules: revise the tag of a word based on the surrounding words or on the tags of the surrounding words.
   - Lexical Rules: use stemming to analyze words not in the lexicon to see if it can make a reasonable guess as to their classification.

POS Tagging

Examples of contextual rules (see here for a larger list)

- NN VB PREVTAG TO
  (common noun becomes verb base if previous is infinitive 'to')
  E.g. `to/TO run/NN` → `to/TO run/VB`
- VB NN PREV1OR2TAG DT
  (VB becomes NN if 1 or 2 of the 2 preceding words is a determiner)
  E.g. `the/DT run/VB` → `the/DT run/NN`
- JJ NN NEXTWD of
  (adjective becomes NN if the following word is 'of')
  E.g. `best/JJ of` → `best/NN of`
- IN DT NEXTTAG NN
  (preposition becomes DT if the next tag is NN)
  E.g. `that/IN cat/NN` → `that/DT cat/NN`
- NN VBP PREVWD who
  (NN becomes verb past tense if previous word is 'who')
  E.g. `who saw/NN` → `who saw/VBP`

POS Tagging

A Bayesian approach:

Find the most likely sequence of tags $t_1^n = t_1 ... t_n$ for the sequence of word $w_1^n = w_1 ... w_n$. More formally:

$$\hat{t}_1^n = arg\, max_{t_1^n} P(t_1^n | w_1^n)$$

(*arg max*$_x f(x)$ is the x that maximizes the value of $f(x)$)

The probability of a given word w being labeled with tag t is the conditional probability:

$$P(t|w) = \frac{P(w,t)}{\sum_t' P(w,t')} = \frac{P(w,t)}{P(w)}$$

Multiplying both sides of the equation by $P(w)$ we get:

$$P(t|w)P(w) = P(w,t)$$

An analogous step can be made if we conditionalize things the other way around:

$$P(w|t) = \frac{P(t,w)}{P(t)}$$

gets us $$P(w|t)P(t) = P(t,w)$$

But because $P(w,t) = P(t,w)$ it follows that:

$$P(t|w)P(w) = P(w|t)P(t)$$

We can now solve for, say $P(t|w)$ by dividing both sides by $P(w)$:

$$P(t|w) = \frac{P(w|t)P(t)}{P(w)}$$

which is equivalent to:

Likelihood$\downarrow$ $\swarrow$Prior

$$\boxed{P(t|w)} = \frac{\boxed{P(w|t)}\,\boxed{P(t)}}{\boxed{\sum_{t'} P(w|t')P(t')}}$$

Posterior$\nearrow$ $\nwarrow$Evidence

Sometimes (as is the case) the denominator can be dropped, therefore simplifying

$$P(t|w) = \frac{P(w|t)P(t)}{P(w)}$$

into:

$$P(t|w) \approx P(w|t)P(t)$$

So, the most likely tag sequence $\hat{t_1^n}$ for w_1^n is:

$$\hat{t_1^n} = arg\,max_{t_1^n} P(t_1^n|w_1^n) = arg\,max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$$

Unfortunately, $P(w_1^n|t_1^n)P(t_1^n)$ is still too hard to compute directly, and so the usual Markov independence assumptions are brought in:

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

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

So:

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

We can easily estimate both probabilities via Maximum Likelihood Estimation, exactly like we did for n-gram models:

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

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

POS Tagging

Hidden Markov Model:

t:	DT	JJ	NN	VBD	NNP
w:	the	big	cat	bit	Sam

... using the probability of word tags:

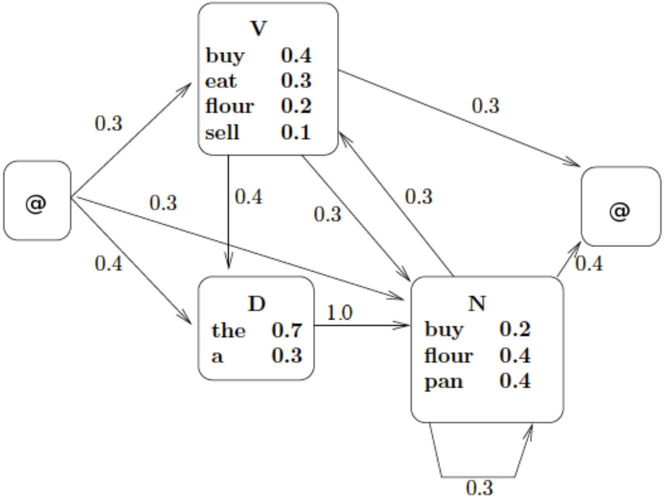
	I	**want**	**to**	**race**
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

... and the probability of tags sequences:

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

POS Tagging

A **HMM** is a directed probabilistic graph:



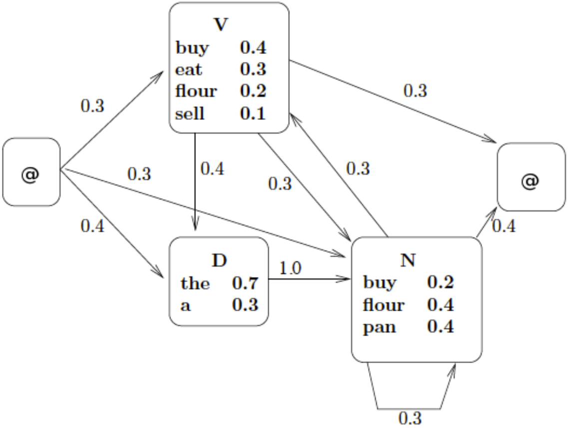
A (perhaps) more intuitive notation:

- $\sigma_{t,t'}$ = the probability $P(t'|t)$ that tag t is followed by t'
- $\tau_{t,w}$ = the probability $P(w|t)$ of t tagging the word w

The HMM Evaluation Problem:

$$P(t_1^n|w_1^n) = \prod_{i=1}^{n+1} \sigma_{t_{i-1},t_i} \tau_{t_i,w_i}$$

POS Tagging



$$P([flour, pan], [V, N]) = \sigma_{@,V} \times \tau_{V,flour} \times \sigma_{V,N} \times \tau_{N,pan} \times \sigma_{N,@} =$$

$$0.3 \times 0.2 \times 0.3 \times 0.4 \times 0.4$$

POS Tagging

The HMM Decoding Problem:
what is the most likely t_1^n for a given w_1^n?

$$\hat{t}_1^n = arg\,max_{t_1^n} P(t_1^n | w_1^n) = arg\,max_{t_1^n} \prod_{i=1}^{n+1} \sigma_{t_{i-1},t_i} \tau_{t_i,w_i}$$

Like the MED, too complex to compute exhaustively. Enter dynamic programming:

Viterbi algorithm:

1. $\mu_{@}(0) = 1.0$
2. $\mu_t(i) =$ for all $t' max(\mu_{t'}(i-1)\sigma_{t',t}\tau_{t,w_i})$

POS Tagging

'Un-MAXed' Trellis chart

	0	her1	dove2	dove3	4
DT	0	1.0 x 0.4 x 0.4 = 0.16	0	0	0
PRP	0	1.0 x 0.4 x 0.1 = 0.04	0	0	0
NN	0	0	0.16 x 0.6 x 0.03 = 0.00192	0.00192 x 0.07 x 0.02 = 0.000002 0.0006 x 0.3 x 0.02 = 0.000003	0
VB	0	0	0.04 x 0.5 x 0.03 = 0.0006	0.00192 x 0.4 x 0.03 = 0.00002	0
@	1.0	0	0	0	0.000003 x 0.6 = 0.000001 0.00002 x 0.3 = 0.000006

$\sigma(@,DT) = 0.4$ $\sigma(@,PRP) = 0.4$ $\sigma(DT,NN) = 0.6$
$\sigma(PRP,VB) = 0.5$ $\sigma(NN,NN) = 0.07$ $\sigma(NN,VB) = 0.4$
$\sigma(VB,NN) = 0.3$ $\sigma(NN,@) = 0.6$ $\sigma(VB,@) = 0.3$

$\tau(DT,her) = 0.4$ $\tau(PRP,her) = 0.1$
$\tau(NN,dove) = 0.02$ $\tau(VB,dove) = 0.03$

'MAXed' Trellis chart

	$_0$	her_1	dove_2	dove_3	$_4$
DT	0	**1.0 x 0.4 x 0.4 = 0.16**	0	0	0
PRP	0	1.0 x 0.4 x 0.1 = 0.04	0	0	0
NN	0	0	**0.16 x 0.6 x 0.02 = 0.00192**	0.00192 x 0.07 x 0.03 = 0.000002	0
VB	0	0	0.04 x 0.5 x 0.03 = 0.0006	**0.00192 x 0.4 x 0.03 = 0.00002**	0
@	1.0	0	0	0	**0.00002 x 0.3 = 0.000006**

$\sigma(@,DT) = 0.4$ $\sigma(@,PRP) = 0.4$ $\sigma(DT,NN) = 0.6$
$\sigma(PRP,VB) = 0.5$ $\sigma(NN,NN) = 0.07$ $\sigma(NN,VB) = 0.4$
$\sigma(VB,NN) = 0.3$ $\sigma(NN,@) = 0.6$ $\sigma(VB,@) = 0.3$

$\tau(DT,her) = 0.4$ $\tau(PRP,her) = 0.1$
$\tau(NN,dove) = 0.02$ $\tau(VB,dove) = 0.03$

POS Tagging

What is the most likely tagging for *Kids like sprouts* ?

$\sigma(\text{NN,VB}) = 0.6$	$\tau(\text{kids,NN}) = 0.4$
$\sigma(\text{@,VB}) = 0.4$	$\tau(\text{kids,VB}) = 0.1$
$\sigma(\text{NN,PREP}) = 0.3$	$\tau(\text{like,VB}) = 0.7$
$\sigma(\text{@,NN}) = 0.5$	$\tau(\text{like,ADJ}) = 0.005$
$\sigma(\text{VB,VB}) = 0.15$	$\tau(\text{sprouts,NN}) = 0.4$
$\sigma(\text{VB,PREP}) = 0.25$	$\tau(\text{sprouts,VB}) = 0.2$
$\sigma(\text{VB,ADJ}) = 0.3$	$\tau(\text{like,PREP}) = 0.02$
$\sigma(\text{VB,NN}) = 0.6$	
$\sigma(\text{NN,ADJ}) = 0.01$	
$\sigma(\text{VB,@}) = 0.4$	
$\sigma(\text{PREP,VB}) = 0.5$	
$\sigma(\text{PREP,NN}) = 0.3$	
$\sigma(\text{ADJ,VB}) = 0.008$	
$\sigma(\text{ADJ,NN}) = 0.4$	
$\sigma(\text{NN,@}) = 0.6$	

POS Tagging

State of the art HMM tagging uses trigrams, and a end-of-sequence marker factor:

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

Where tag trigrams are estimated with MLE:

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

See a problem with this?

POS Tagging

To deal with massive trigram sparseness: **deleted interpolation**

$$P(t_i|t_{i-1}, t_{i-2}) = \lambda_3 \frac{C(t_{i-2}, t_{i-1}, t_i)}{C(t_{i-2}, t_{i-1})} + \lambda_2 \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} + \lambda_1 \frac{C(t_i)}{N}$$

1. $\lambda_1 = \lambda_2 = \lambda_3 = 0$
2. For each trigram with non-zero counts, apply case that has maximum counts:

   Case 1: $\frac{C(t_1, t_2, t_3) - 1}{C(t_1, t_2) - 1}$ increment λ_3 by $C(t_1, t_2, t_3)$

   Case 2: $\frac{C(t_2, t_3) - 1}{C(t_2) - 1}$ increment λ_2 by $C(t_1, t_2, t_3)$

   Case 3: $\frac{C(t_3) - 1}{N - 1}$ increment λ_1 by $C(t_1, t_2, t_3)$
3. Normalize $\lambda_1, \lambda_2, \lambda_3$

POS Tagging

Unknown words: inspect the suffixes (up to some maximum i)

$$P(t_i|s_{n-i+1}...s_n)$$

By not normalizing capitalization, a state-of-the-art trigram HMM like Brants (2000) can achieves 96.7% accuracy on the Penn Treebank.

Read more about the TnT package here.

Maximum Entropy Markov models

$$t_1^n = \text{arg } \max_{t_1^n} \prod_i P(t_i|w_i, t_{i-1})$$

- HMMs compute the likelihood
  (observed word conditioned on tag)
- MEMMs compute the posterior
  (tag conditioned on observed word and more...)

... such as neighboring words, previous tags, and various combinations, using feature templates.

$$f_i(w, window, prev-tags) = \begin{cases} 1 \text{if } w \text{ ends in 'ing' and prev-tags} = \text{VBG NN} \\ 0 \text{o.w.} \end{cases}$$

POS Tagging

Given a large set of features, the most likely sequence of tags is computed by a maximum entropy model (soft max model, log-linear, conditional random field model):

$$t_1^n = \text{arg } \max_{t_1^n} \prod_i \frac{exp\left(\sum_i v_i f_i(t_i, w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}{\sum_{t' \in tagset} exp\left(\sum_i v_i f_i(t', w_{i-l}^{i+l}, t_{i-k}^{i-1})\right)}$$

Where:
v_i is a weight $\in \mathbb{R}$
w_i is a word in position i
w_{i-l}^{i+l} are its neighbors within a window of l words
k_{i-k}^{i-1} are the previous k tags

POS Tagging

A toy example:

Suppose we wanted to label every word with one of two tags ('ProperName' and 'Other'), and used the indicator functions:

$$f_1(t_i, \text{window}_i, \text{prevtag}_i) = \begin{cases} 1 \text{ if 1stCharCase}(w_i) = \text{upper}\& \\ \quad \text{prevtag}_i \neq @ \;\&\; t_i = \text{ProperName} \\ 0 \;\; \text{o.w.} \end{cases}$$

$$f_2(t_i, \text{window}_i, \text{prevtag}_i) = \begin{cases} 1 \text{ if prevtag}_i = @ \;\&\; t_i = \text{ProperName} \\ 0 \;\; \text{o.w.} \end{cases}$$

$$f_3(t_i, \text{window}_i, \text{prevtag}_i) = \begin{cases} 1 \text{ if 1stCharCase}(w_i) = \text{lower} \;\&\; t_i = \text{Other} \\ 0 \;\; \text{o.w.} \end{cases}$$

$$f_4(t_i, \text{window}_i, \text{prevtag}_i) = \begin{cases} 1 \text{ prevtag}_i = @ \;\&\; t_i = \text{Other} \\ 0 \;\; \text{o.w.} \end{cases}$$

POS Tagging

Each indicator function f_i has a respective weight $v_i \in \mathbb{R}$

$v_1 = 1.9, v_2 = 0.1, v_3 = 2.0, v_4 = 0.4$

Suppose we are tagging *@ Help Robin @*:

$$P(\text{ProperName}|\text{Help}) = \frac{exp(0.1)}{exp(1.9+0.1)+exp(2.0+0.4)} = 0.06$$

$$P(\text{Other}|\text{Help}) = \frac{exp(0.4)}{exp(1.9+0.1)+exp(2.0+0.4)} = 0.08$$

$$P(\text{ProperName}|\text{Robin}) = \frac{exp(1.9)}{exp(1.9+0.1)+exp(2.0+0.4)} = 0.36$$

$$P(\text{Other}|\text{Robin}) = \frac{exp(0)}{exp(1.9+0.1)+exp(2.0+0.4)} = 0.05$$

POS Tagging

Several ways to create/discover features:

- By hand
- Unsupervised learning
- Enumeratively

Example of automatic feature generation

Suppose we are learning how to tag the word ‘back’, and it appears in the training corpus as

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

Then we would generate features with the following constraints:

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

Training the model:

$$\hat{v} = \text{arg max}_v \left(\sum_i logP(t_i|w_i, v) - \frac{\lambda}{2} \sum_k v_k^2 \right)$$

Training set consists of all word history and tag sequences in the training data.

The values of the weights v are set by stochastic gradient descent or simulated annealing.

See an implementation here.