CSC 594 Topics in AI – Applied Natural Language Processing

Fall 2009/2010

6. Part-Of-Speech (POS) Tagging

Grammatical Categories: Parts-of-Speech

- 8 (ish) traditional parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction, etc.
- Nouns: people, animals, concepts, things (e.g. "birds")
- Verbs: express action in the sentence (e.g. "sing")
- Adjectives: describe properties of nouns (e.g. "yellow")
- etc.

POS examples

• N	noun	*chair, bandwidth, pacing*
• V	verb	*study, debate, munch*
• ADJ	adjective	*purple, tall, ridiculous*
• ADV	adverb	*unfortunately, slowly*
• P	preposition	*of, by, to*
• PRO	pronoun	*I, me, mine*
• DET	determiner	*the, a, that, those*

Source: Jurafsky & Martin "Speech and Language Processing"

POS Tagging

- The process of assigning a part-of-speech or lexical class marker to each word in a sentence (and all sentences in a collection).

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

Source: Jurafsky & Martin "Speech and Language Processing" 4

Why is POS Tagging Useful?

- First step of a vast number of practical tasks
- Helps in stemming
- Parsing
  - Need to know if a word is an N or V before you can parse
  - Parsers can build trees directly on the POS tags instead of maintaining a lexicon
- Information Extraction
  - Finding names, relations, etc.
- Machine Translation
- Selecting words of specific Parts of Speech (e.g. nouns) in pre-processing documents (for IR etc.)

Source: Jurafsky & Martin "Speech and Language Processing" 5

POS Tagging
Choosing a Tagset

- To do POS tagging, we need to choose a standard set of tags to work with
- Could pick very coarse tagsets
  - N, V, Adj, Adv.
- More commonly used set is finer grained, the "Penn TreeBank tagset", 45 tags
  - PRP$, WRB, WP$, VBG
- Even more fine-grained tagsets exist

Source: Jurafsky & Martin "Speech and Language Processing" 6

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*			

Using the Penn Tagset

- Example:
  ```
  The/DT grand/JJ jury/NN commented/VBD on/IN
    a/DT number/NN of/IN other/JJ topics/NNS ./.
  ```
- Prepositions and subordinating conjunctions marked IN ("although/IN I/PRP..")
- Except the preposition/complementizer "to" is just marked "TO".

Source: Jurafsky & Martin "Speech and Language Processing"

Tagged Data Sets

- Brown Corpus
  - An early digital corpus (1961)
  - Contents: 500 texts, each 2000 words long
  - From American books, newspapers, magazines
  - Representing genres:
    - Science fiction, romance fiction, press reportage scientific writing, popular lore
  - 87 different tags
- Penn Treebank
  - First large syntactically annotated corpus
  - Contents: 1 million words from Wall Street Journal
  - Part-of-speech tags and syntax trees
  - 45 different tags
  - Most widely used currently

Source: Andrew McCallum, UMass Amherst

POS Tagging

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

Another example of Part-of-speech ambiguities

```
NNP   NNS    NNS       NNS   CD  NN
      VBZ    VBZ       VBZ
             VB
“Fed raises interest rates 0.5 %  in effect to
                                    control inflation”
```


Current Performance

Input: **`the lead paint is unsafe`**
Output: **`the/Det lead/N paint/N is/V unsafe/Adj`**

- Using state-of-the-art automated method, how many tags are correct?
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of simplest possible method:
      Tag every word with its most frequent tag, and
      Tag unknown words as nouns

How Hard is POS Tagging? Measuring Ambiguity

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

Three Methods for POS Tagging

1. Rule-based
   - Hand-coded rules
2. Probabilistic/Stochastic
   - **Sequence** (n-gram) models; machine learning
     - HMM (Hidden Markov Model)
     - MEMMs (Maximum Entropy Markov Models)
3. Transformation-based
   - Rules + n-gram machine learning
     - Brill tagger

Source: Jurafsky & Martin “Speech and Language Processing” 13

Rule-Based POS Tagging (1)

- Make up some regexp rules that make use of morphology

```
>>> patterns = [
    (r'^-?[0-9]+(.[0-9]+)?', 'cd'),      # cardinal numbers
    (r'(The|the|A|a|An|an|)$', 'at'),    # articles / determiners
    (r'un.*', 'jj'),                     # adjectives
    (r'.*\'s$', 'nn$'),                  # possesive nouns
    (r'.*s$', 'nns'),                    # plural nouns
    (r'.*ing$', 'vbg'),                  # gerunds
    (r'.*ed$', 'vbd'),                   # past tense verbs
    (r',', ','),                         # comma
    (r'^(!|\?|\.|\"|\"\'|)$', '.'),        # other punctuation
    (r'.*', 'nn')                        # nouns (default)
    ]

>>> regexp_tagger = tag.Regexp(patterns)
```

Source: Marti Hearst, i256, at UC Berkeley 14

Rule-Based POS Tagging (2)

- “Two-level morphology” scheme (used in ENGTWOL)
  - Start with a dictionary
  - [Stage 1] Assign all possible tags to words from the dictionary
  - [Stage 2] Write rules by hand to selectively remove tags
  - Leaving the correct tag for each word.

Source: Jurafsky & Martin “Speech and Language Processing” 15

Stage 1 of ENGTWOL Tagging

- First Stage: Run words through FST 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**

Source: Jurafsky & Martin "Speech and Language Processing" 16

Stage 2 of ENGTWOL Tagging

- Second Stage: Apply NEGATIVE constraints.
- Example: Adverbial "that" rule
  - Eliminates all readings of "that" except the one in
    - "It isn't *that* odd"

Given input: "that"
If
(+1 A/ADV/QUANT) **; if next word is adj/adv/quantifier**
(+2 SENT-LIM) **; following which is E-O-S**
(NOT -1 SVOC/A) **; and the previous word is not a**
; verb like "consider" which
; allows adjective complements
; in "I consider that odd"
Then eliminate non-ADV tags
Else eliminate ADV

Source: Jurafsky & Martin "Speech and Language Processing" 17

Probabilistic POS Tagging (1)

- N-grams
  - The N stands for how many terms are used/looked at
    - Unigram: 1 term (0th order)
    - Bigram: 2 terms (1st order)
    - Trigrams: 3 terms (2nd order)
      - Usually don't go beyond this
  - You can use different kinds of terms, e.g.:
    - Character, Word, POS
  - Ordering
    - Often adjacent, but not required
  - We use n-grams to help determine the *context* in which some linguistic phenomenon happens.
    - e.g., Look at the words before and after the period to see if it is the end of a sentence or not.

Source: Marti Hearst, i256, at UC Berkeley 18

Probabilistic POS Tagging (2)

- Tagging with lexical frequencies

```
Secretariat/NNP is/VBZ expected/VBN to/TO race/VB
  tomorrow/NN
People/NNS continue/VBP to/TO inquire/VB the/DT
  reason/NN for/IN the/DT race/NN for/IN outer/JJ
  space/NN
```

  - Problem: assign a tag to “race” given its lexical frequency
  - Solution: we choose the tag that has the greater conditional probability -> a probability of the word in a given POS
    - P(**race**|VB)
    - P(**race**|NN)

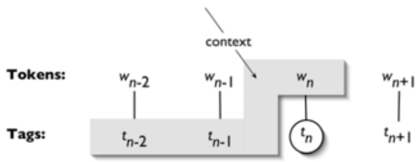
Unigram Tagger

- Train on a set of sentences
- Keep track of how many times each word is seen with each tag.
- After training, associate with each word its most likely tag.
  - Problem: many words never seen in the training data.
  - Solution: have a default tag to “backoff” to.

More **problems**…
- Most frequent tag isn’t always right!
- Need to take the context into account
  - Which sense of “to” is being used?
  - Which sense of “like” is being used?

N-gram Tagger

- Uses the preceding N-1 predicted tags
- Also uses the unigram estimate for the current word



How N-gram Tagger Works

- Constructs a frequency distribution describing the frequencies each word is tagged with in different contexts.
  - The context considered consists of the word to be tagged and the n-1 previous words' tags.
- After training, tag words by assigning each word the tag with the maximum frequency given its context.
  - Assigns “None” tag if it sees a word in a context for which it has no data (which it has not seen).
- Tuning parameters
  - “cutoff” is the minimal number of times that the context must have been seen in training in order to be incorporated into the statistics
  - Default cutoff is 1

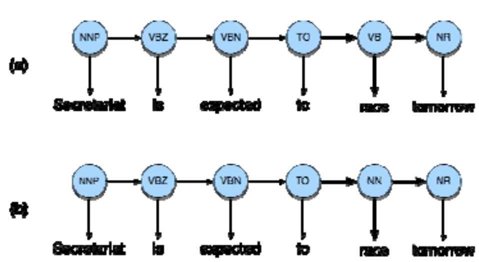
Source: Marti Hearst, i256, at UC Berkeley

POS Tagging as Sequence Classification

- We are given a sentence (an “observation” or “sequence of observations”)
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Source: Jurafsky & Martin “Speech and Language Processing”

Disambiguating “race”



Source: Jurafsky & Martin “Speech and Language Processing”

Example

Using the maximum likelihood and conditional independence assumptions, we have:

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

- 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 reading (when n = 2, **bi-gram**)

Source: Jurafsky & Martin "Speech and Language Processing"

25

Transformation-Based Tagger

- The Brill tagger (by E. Brill)
  - Basic idea: do a quick job first (using frequency), then revise it using contextual rules.
  - Painting metaphor from the readings
  - Very popular (freely available, works fairly well)
  - A supervised method: requires a tagged corpus

Source: Marti Hearst, i256, at UC Berkeley

26

Brill Tagger: In more detail

- Start with simple (less accurate) rules…learn better ones from tagged corpus
  - Tag each word initially with most likely POS
  - Examine set of transformations to see which improves tagging decisions compared to tagged corpus
  - Re-tag corpus using best transformation
  - Repeat until, e.g., performance doesn't improve
  - Result: tagging procedure (ordered list of transformations) which can be applied to new, untagged text

Source: Marti Hearst, i256, at UC Berkeley

27

Examples

- Examples:
  - They are expected to race tomorrow.
  - The race for outer space.
- Tagging algorithm:
  1. Tag all uses of "race" as NN (most likely tag in the Brown corpus)
     - They are expected to race/NN tomorrow
     - the race/NN for outer space
  2. Use a transformation rule to replace the tag NN with VB for all uses of "race" preceded by the tag TO:
     - They are expected to race/VB tomorrow
     - the race/NN for outer space

Source: Marti Hearst, i256, at UC Berkeley

28

Sample Transformation Rules

```
Rules:
NN -> NNP if the tag of words i+1...i+2 is 'NNP'
NN -> VB if the tag of the preceding word is 'TO'
NN -> VBD if the tag of the following word is 'DT'
NN -> VBD if the tag of the preceding word is 'NNS'
NN -> JJ if the tag of the preceding word is 'DT', and the tag of the followi
ng word is 'NN'
NN -> NNP if the tag of the preceding word is 'NN', and the tag of the follow
ing word is ','
NN -> NNP if the tag of words i+1...i+2 is 'NNP'
NN -> IN if the tag of the preceding word is '.'
NNP -> NN if the tag of words i-3...i-1 is 'JJ'
NN -> JJ if the tag of the following word is 'JJ'
NN -> VBP if the tag of the preceding word is 'PRP'
WDT -> IN if the tag of the following word is 'DT'
NN -> JJ if the tag of the preceding word is 'IN', and the tag of the followi
ng word is 'NN'
NN -> VBN if the tag of the preceding word is 'VBP'
VBD -> VB if the tag of the preceding word is 'MD'
NN -> JJ if the tag of the preceding word is 'CC', and the tag of the followi
ng word is 'NN'
```

Source: Marti Hearst, i256, at UC Berkeley

29

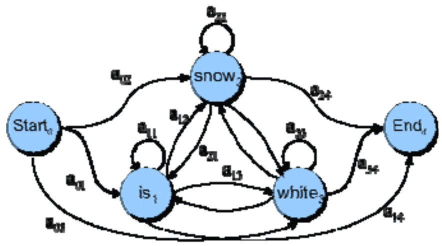
Hidden Markov Models (HMM)

- The n-gram example shown earlier is essentially a Hidden Markov Model (HMM)
- Definitions:
  - A weighted finite-state automaton adds probabilities to the arcs
    - The sum of the probabilities leaving any arc must sum to one
  - A Markov chain is a special case of a WFST in which the input sequence uniquely determines which states the automaton will go through
  - Markov chains can't represent inherently ambiguous problems
    - Useful for assigning probabilities to unambiguous sequences

Source: Jurafsky & Martin "Speech and Language Processing"

30

Markov Chain for Words



Markov Chain: "First-order observable Markov Model"

- A set of states
  - $Q = q_1, q_2 \ldots q_N;$ the state at time t is q_t
- Transition probabilities:
  - a set of probabilities $A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$.
  - Each a_{ij} represents the probability of transitioning from state i to state j
  - The set of these is the transition probability matrix A
- Current state only depends on previous state

Hidden Markov Model (1)

- In part-of-speech tagging (and other things)
  - The output symbols are **words**
  - But the hidden states are **part-of-speech tags**
- So we need an extension!
- A Hidden Markov Model is an extension of a Markov chain in which the input symbols are not the same as the states.
- This means we don't know which state we are in.

Hidden Markov Model (2)

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

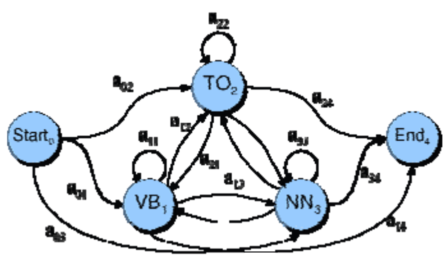
    $a_{ij} = P(q_t = j \mid q_{t-1} = i) \quad 1 \le i, j \le N$
- Observation likelihoods
  - Output probability matrix $B = \{b_i(k)\}$

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

    $\pi_i = P(q_1 = i) \quad 1 \le i \le N$
- Special initial probability vector π

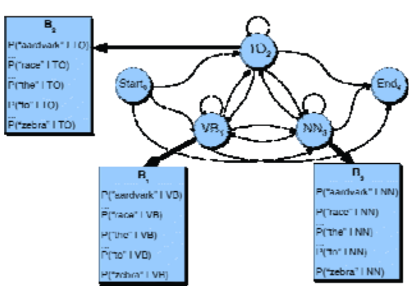
Source: Jurafsky & Martin "Speech and Language Processing" 34

Transition Probabilities



Source: Jurafsky & Martin "Speech and Language Processing" 35

Observation Likelihoods



Source: Jurafsky & Martin "Speech and Language Processing" 36

Decoding

- 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.
  - Not a good idea.
  - Luckily dynamic programming helps us here

Source: Jurafsky & Martin "Speech and Language Processing" 37

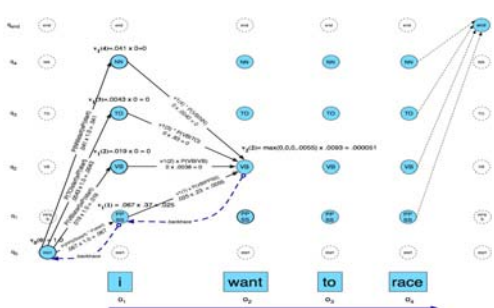
The Viterbi Algorithm

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**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; 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 $backpointer[q_F,T]$

Source: Jurafsky & Martin "Speech and Language Processing" 38

Viterbi Example



Source: Jurafsky & Martin "Speech and Language Processing" 39

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass filling the columns left to right using our transition probabilities and observations probabilities
- Dynamic programming key is that we need only store the MAX probability path to each cell, (not all paths).

Source: Jurafsky & Martin "Speech and Language Processing"

40

POS-tagging Evaluation

- 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.

Source: Jurafsky & Martin "Speech and Language Processing"

41