Statistical Natural Language Processing

Lecture 8: Part of Speech Tagging

Dr. Momtazi
Amirkabir University of Technology

Outline

Outline

1. Part of Speech Tagging

2. Sequential Modeling

3. Evaluation

Parts Of Speech (POS)

- 8 Parts of speech are traditionally used to summarize the linguistic knowledge
  - Noun, Verb, Preposition, Adverb, Article, Interjection, Pronoun, Conjunction

- The modified list is currently used
  - Noun, Verb, Auxiliary, Preposition, Adjective, Adverb, Number, Determiner, Interjection, Pronoun, Conjunction, Particle

- Known as:
  - Parts of speech
  - Lexical categories
  - Word classes
  - Morphological classes
  - Lexical tags

POS Examples

Noun	book/books, sugar, Germany, Sony
Verb	eat, wrote
Auxiliary	can, should, have
Adjective	new, newer, newest
Adverb	well, urgently
Numbers	872, two, first
Determiner	the, some
Conjunction	and, or
Pronoun	he, my
Preposition	to, in
Particle	off, up
Interjection	Ow, Eh

Open vs. Closed Classes

- Closed (limited number of words, do not grow usually)
  - Determiners: the, some, a, an, ...
  - Pronouns: she, he, I, ...
  - Prepositions: to, in, on, under, over, by, ...
  - Auxiliaries: can, should, have, had, ...
  - Conjunctions: and, or
  - Particles: off, up
  - Interjections: Ow, Eh

- Open (unlimited number of words)
  - Nouns
  - Verbs
  - Adjectives
  - Adverbs

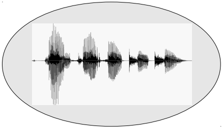
Applications

- Speech Synthesis
- Parsing
- Machine Translation
- Information Extraction

Applications

- Speech Synthesis

How to pronounce *“lead”* ?



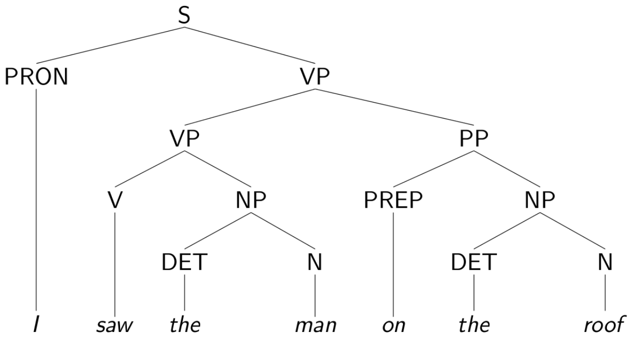
Applications

- Machine Translation

"I like ..." $\Rightarrow$

Applications

- Parsing



POS Tagset

- There are so many parts of speech tagsets we can draw
- Choosing a standard tagset is essential
- Tag types
  - Coarse-grained
    - noun
    - verb
    - adjective
    - ...
  - Fine-grained
    - noun-proper-singular, noun-proper-plural, noun-common-mass, ..
    - verb-past, verb-present-3rd, verb-base, ...
    - adjective-simple, adjective-comparative, ...
    - ...

Penn TreeBank

A large annotated corpus of English tagset: 45 tags

Penn TreeBank 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*			

Ambiguity

- Definition
  - The process of assigning a part of speech to each word in a text

- Challenge
  - Words often have more than one POS

On my back

The back door

Pay the money back

Promised to back the bill

Ambiguity

- Definition
  - The process of assigning a part of speech to each word in a text
- Challenge
  - Words often have more than one POS

On my back$_{[NN]}$

The back$_{[JJ]}$ *door*

Pay the money back$_{[RB]}$

Promised to back$_{[VB]}$ *the bill*

Distribution of Ambiguities

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

Distribution of Ambiguities

- The frequency of ambiguous words are relatively high
  - 11.5% of word types
  - 40% of word tokens

Goal

- Using a set of labeled data to train a model
- Using the trained model to predict the POS tag of the unseen words

POS Tagging

Plays well with others

Plays	NNS/VBZ
well	UH/JJ/NN/RB
with	IN
others	NNS

Plays$_{[VBZ]}$ *well*$_{[RB]}$ *with*$_{[IN]}$ *others*$_{[NNS]}$

Performance

- Baseline model
  - Tagging unambiguous words with the correct label
  - Tagging ambiguous words with their most frequent label
  - Tagging unknown words as a noun

Already performs around 90%

Outline

Task

- Similar to a normal classification task
  - Feature Selection
  - Algorithm

POS Tagging

- Features

Word	the: the $\rightarrow$ DT
Prefixes	unbelievable: un- $\rightarrow$ JJ
Suffixes	slowly: -ly $\rightarrow$ RB
Lowercased word	Importantly: importantly $\rightarrow$ RB
Capitalization	Stefan: [CAP] $\rightarrow$ NNP
Word shapes	35-year: d-x $\rightarrow$ JJ

- Model
  - Maximum Entropy
    $P(t|w)$

Data	Performance
Overall	93.7
Unknown	82.6

POS Tagging

- More Features?

They$_{[PRP]}$ *left*$_{[VBD]}$ *as*$_{\mathbf{[IN]}}$ *soon*$_{[RB]}$ *as*$_{[IN]}$ *he*$_{[PRP]}$ *arrivied*$_{[VBD]}$

- Better Algorithm
  - Using Sequence Modeling

Sequence Modeling

- Many of the NLP techniques should deal with data represented as sequence of items
  - Characters, Words, Phrases, Lines, ...

警察枪杀了那个逃

B I BI B B B B I

$I_{[PRP]}$ $saw_{[VBP]}$ $the_{[DT]}$ $man_{[NN]}$ $on_{[IN]}$ $the_{[DT]}$ $roof_{[NN]}$.

Sequence Modeling

- Two types of information
  - Local
  - Contextual

Sequence Modeling

- Making a decision based on the
  - Current Observation
    - Word (W_0)
    - Prefix
    - Suffix
    - Lowercased word
    - Capitalization
    - Word shape
  - Surrounding observations
    - W_{+1}
    - W_{-1}
  - Previous decisions
    - T_{-1}
    - T_{-2}

Learning Model

Training

$T_1 \rightarrow C_1$
$T_2 \rightarrow C_2$
$\dots$
$T_n \rightarrow C_n$

$\longrightarrow$

F_1
F_2
$\dots$
F_n

$\longrightarrow$ Model(F,C)

Testing

$T_{n+1} \rightarrow ?$ $\longrightarrow$ F_{n+1} $\longrightarrow$ C_{n+1}

Sequence Modeling

- Greedy inference
  - Starting from the beginning of the sequence
  - Assigning a label to each item using the classifier in that position
  - Using previous decisions as well as the observed data

- Beam inference
  - Keeping the top *k* labels in each position
  - Extending each sequence in each local way
  - Finding the best *k* labels for the next position

Hidden Markov Model (HMM)

- Finding the best sequence of tags ($t_1 ... t_n$) that corresponds to the sequence of observations ($w_1 ... w_n$)

- Probabilistic View
  - Considering all possible sequences of tags
  - Choosing the tag sequence from this universe of sequences, which is most probable given the observation sequence

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

Using Bayes Rule

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

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$$P(t_1^n | w_1^n) = \frac{P(w_1^n | t_1^n) \cdot P(t_1^n)}{P(w_1^n)}$$

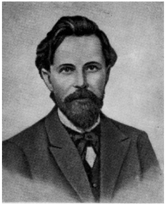
$$\hat{t}_1^n = \text{argmax}_{t_1^n} P(w_1^n | t_1^n) \cdot P(t_1^n)$$

Using Markov Assumption

$$\hat{t}_1^n = \text{argmax}_{t_1^n} P(w_1^n|t_1^n) \cdot P(t_1^n)$$

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

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



$$\hat{t}_1^n = \text{argmax}_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i) \cdot P(t_i|t_{i-1})$$

Two Probabilities

- The tag transition probabilities: $P(t_i|t_{i-1})$
  - Finding the likelihood of a tag to proceed by another tag
  - Similar to the normal bigram model

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

Two Probabilities

- The word likelihood probabilities: $P(w_i|t_i)$
  - Finding the likelihood of a word to appear given a tag

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

Two Probabilities

- Zero probability problem
  - Solution: similar to language modelling, use the smoothing method for both probabilities

Two Probabilities

$I_{[PRP]}$ $saw_{[VBP]}$ $the_{[DT]}$ $man_{[NN?]}$ $on_{[]}$ $the_{[]}$ $roof_{[]}$.

$$P([NN]|[DT]) = \frac{C([DT],[NN])}{C([DT])}$$

$$P(man|[NN]) = \frac{C([NN],man)}{C([NN])}$$

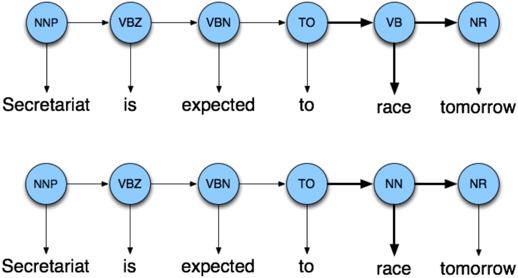
Ambiguity

Secretariat$_{[NNP]}$ *is*$_{[VBZ]}$ *expected*$_{[VBN]}$ *to*$_{[TO]}$ ***race***$_{[VB]}$ *tomorrow*$_{[NR]}$.

People$_{[NNS]}$ *inquire*$_{[VB]}$ *the*$_{[DT]}$ *reason*$_{[NN]}$ *for*$_{[IN]}$ *the*$_{[DT]}$ ***race***$_{[NN]}$.

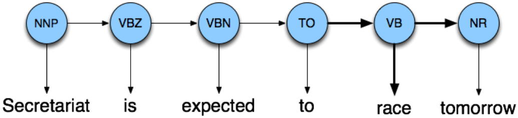
Ambiguity

Secretariat[NNP] *is*[VBZ] *expected*[VBN] *to*[TO] ***race***[VB] *tomorrow*[NR].



Ambiguity

Secretariat$_{[NNP]}$ *is*$_{[VBZ]}$ *expected*$_{[VBN]}$ *to*$_{[TO]}$ ***race***$_{[VB]}$ *tomorrow*$_{[NR]}$.



$$P(VB|TO) = 0.83$$

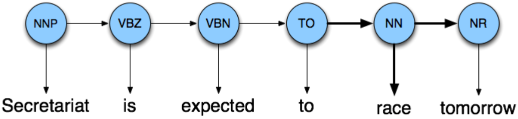
$$P(race|VB) = 0.00012$$

$$P(NR|VB) = 0.0027$$

$$P(VB|TO)P(NR|VB)P(race|VB) = 0.00000027$$

Ambiguity

Secretariat$_{[NNP]}$ *is*$_{[VBZ]}$ *expected*$_{[VBN]}$ *to*$_{[TO]}$ ***race***$_{[VB]}$ *tomorrow*$_{[NR]}$.



$$P(NN|TO) = 0.00047$$
$$P(race|NN) = 0.00057$$
$$P(NR|NN) = 0.0012$$

$$P(NN|TO)P(NR|NN)P(race|NN) = 0.00000000032$$

Performance

- Model
  - Maximum Entropy
    $P(t|w)$

Data	Performance
Overall	93.7
Unknown	82.6

  - HMM

Data	Performance
Overall	96.2
Unknown	86.0

- Upper bound (human agreement): $\sim$98%

Hidden Markov Model (HMM)

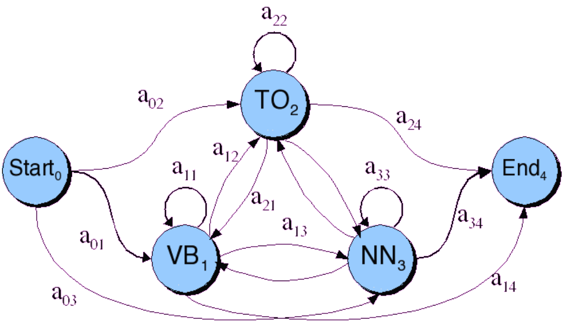
- A weighted finite-state automaton adds probabilities to the arcs
  - The probabilities leaving any arc must sum to one

- An HMM is an extension of a Markov chain in which the input symbols are not the same as the states

- We do not know which state we are in
  - The output symbols are words
  - The hidden states are POS tags

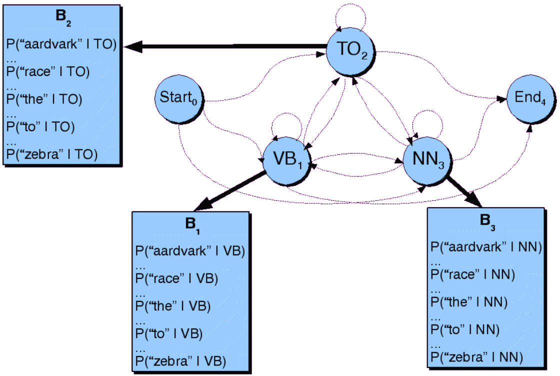
Hidden Markov Model (HMM)

- Transition probabilities



Hidden Markov Model (HMM)

- Word likelihood probabilities



The Viterbi Algorithm

- Viterbi inference
  - Memorizing the model using dynamic programming
  - Considering the small window of previous decisions

The Viterbi Algorithm

- Creating an array
  - Columns corresponding to inputs
  - Rows corresponding to possible states

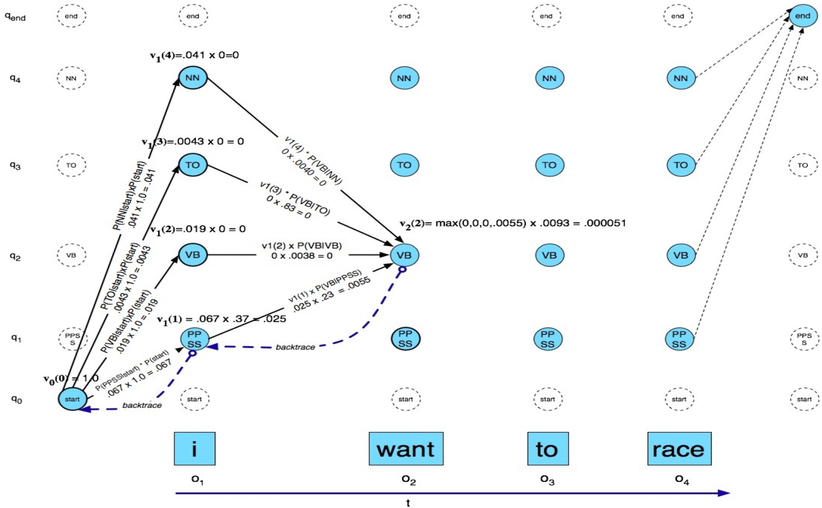
- Sweeping through the array in one pass filling the columns left to right using the transition probabilities and observation probabilities

- Storing the max probability path to each cell (not all paths) using dynamic programming

The Viterbi Algorithm

- Basic idea behind the algorithm:
  the recursive definition for finding the maximum probability

The Viterbi Algorithm



Hidden Markov Model (HMM)

- Proc and Cons
  - HMM taggers are very simple to train
    - Just need to compile counts from the training corpus
  - Perform relatively well
  - Main difficulty is modelling $P(word|tag)$ specially for complex words

Outline

1. Part of Speech Tagging

2. Sequential Modeling

3. Evaluation

Evaluating POS Taggers

- Comparing the output of a tagger with a human-labelled gold standard

- Accuracy:

$$\textit{Accuracy} = \frac{\#\text{currectly tagged words}}{\#\text{total word token}}$$

Evaluating POS Taggers

- Accuracy:

$$Accuracy = \frac{tp}{N}$$

$$Accuracy = \frac{\sum_c^C tp_c}{N}$$

Evaluating POS Taggers

- The accuracy score doesn't show everything
- It is useful to know what is misclassified as what

- Solution: providing a confusion matrix
  - A matrix (# tags x #tags): the rows correspond to the correct tags and the columns correspond to the tagger output
  - $Cell(i, j)$ gives the count of the number of times tag i was classified as tag j
  - The leading diagonal elements correspond to correct classifications
  - Off diagonal elements correspond to misclassifications

- A good approach for error analysis

Further Reading

- Speech and Language Processing
  - Chapter 5: POS Tagging
  - Chapter 6: MaxEnt & HMM