Last Class:

1. EM

Today: Part-of-Speech Tagging

1. Background
2. HMM Tagger

Slide CS674–2

POS tags

"There are 10 parts of speech, and they are all troublesome."
-*Mark Twain*

- POS tags are also known as word classes, morphological classes, or lexical tags.
- Typically much larger than Twain's 10:
  - Penn Treebank: 45
  - Brown corpus: 87
  - C7 tagset: 146

Slide CS674–3

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	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	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*			

Slide CS674–4

Why do POS tagging?

1. Provides a lot of information about the word and its neighbors. Useful for speech recognition.
2. Can tell us something about how the word is pronounced. Useful for speech synthesis systems.
3. Can be used in IR systems...to aid stemming algorithms, to select nouns.
4. Can aid WSD algorithms.
5. Used in ASR language models, e.g. in class-based N-gram language models.
6. Critical for partial parsing algorithms.

Slide CS674–5

Part-of-Speech Tagging Baseline

How hard is p-o-s tagging?

Given word w, find the most likely tag t, i.e. find the tag that maximizes: $P(t|w)$

Maximum Likelihood Estimator: 90% accuracy rate.

To improve reliability: *need to use some of the local context.*

Approaches

1. **rule-based**: involve a large database of hand-written disambiguation rules, e.g. that specify that an ambiguous word is a noun rather than a verb if it follows a determiner.
2. **stochastic**: resolve tagging ambiguities by using a training corpus to compute the probability of a given word having a given tag in a given context.

   HMM tagger, Maximum Likelihood Tagger, Markov model tagger
3. **hybrid**: E.g. transformation-based tagger (Brill tagger); learns symbolic rules based on a corpus.
4. **ensemble methods**: combine the results of multiple taggers.

HMM Tagger

Given $W = w_1, \ldots, w_n$, find $T = t_1, \ldots, t_n$ that maximizes

$$P(t_1, \ldots, t_n | w_1, \ldots, w_n)$$

Restate using Bayes' rule:

$$(P(t_1, \ldots, t_n) * P(w_1, \ldots, w_n | t_1, \ldots, t_n))/P(w_1, \ldots, w_n)$$

Ignore denominator...
Make independence assumptions...

Independence Assumptions (factor 1)

$P(t_1, \ldots, t_n)$: approximate using **n-gram model**

bigram $\prod_{i=1,n} P(t_i \,|\, t_{i-1})$

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

Independence Assumptions (factor 2)

$P(w_1, \ldots, w_n \,|\, t_1, \ldots, t_n)$: approximate by assuming that a word appears in a category independent of its neighbors

$$\prod_{i=1,n} P(w_i \,|\, t_i)$$

Assuming bigram model:

$$P(t_1, \ldots, t_n) * P(w_1, \ldots, w_n | t_1, \ldots, t_n) \approx$$

$$\prod_{i=1,n} P(t_i | t_{i-1}) * P(w_i | t_i)$$

Hidden Markov Models

Equation can be modeled by an HMM.

- **states**: represent a possible lexical category
- **transition probabilities**: bigram probabilities
- **observation probabilities**, **lexical generation probabilities**: indicate, for each word, how likely that word is to be selected if we randomly select the category associated with the node.

Viterbi Algorithm

c: number of lexical categories

$P(w_t | t_i)$: lexical generation probabilities

$P(t_i | t_j)$: bigram probabilities

Find most likely sequence of lexical categories $T_1, \ldots, T_n$ for word sequence.

Initialization

For i = 1 to c do
SCORE(i,1) = $P(t_i | \phi) * P(w_1 | t_i)$
BPTR(i,1) = 0

Iteration

For t = 2 to n
For i = 1 to c
SCORE(i,t) =
$MAX_{j=1,c}(SCORE(j, t-1) * P(t_i | t_j)) * P(w_t | t_i)$
BPTR(i,t) = index of j that gave max

Identify Sequence

T(n) = i that maximizes SCORE(i,n)
For i = n-1 to 1 do
T(i) = BPTR(T(i+1), i+1)

Results

- Effective if probability estmates are computed from a large corpus
- Effective if corpus is of the same style as the input to be classified
- Consistently achieve accuracies of 96% or better using trigram model
- Cuts error rate in half vs. naive algorithm (90% accuracy rate)
- Can be smoothed using backoff or deleted interpolation...

Extensions

- Can train HMM tagger on unlabeled data using the EM algorithm, starting with a dictionary that lists which tags can be assigned to which words.
- EM then learns the word likelihood function for each tag, and the tag transition probabilities.
- Merialdo (1994) showed, however, that a tagger trained on even a small amount hand-tagged data works better than one trained via EM.