Part-of-Speech Tagging

Berlin Chen 2003

References:

1. Speech and Language Processing, chapter 8
2. Foundations of Statistical Natural Language Processing, chapter 10

Review

- Tagging (part-of-speech tagging)
  - The process of assigning (labeling) a part-of-speech or other lexical class marker to each word in a sentence (or a corpus)
    - Decide whether each word is a noun, verb, adjective, or whatever

  The/AT representative/NN put/VBD chairs/NNS on/IN the/AT table/NN
  - *An intermediate layer* of representation of syntactic structure
    - *When compared with syntactic parsing*
  - Above 96% accuracy for most successful approaches

Introduction

- Parts-of-speech
  - Known as POS, word classes, lexical tags, morphology classes
- Tag sets
  - Penn Treebank : 45 word classes used (Francis, 1979)
    - Penn Treebank is a parsed corpus
  - Brown corpus: 87 word classes used (Marcus et al., 1993)
  - ….

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

The Penn Treebank POS Tag Set

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*			

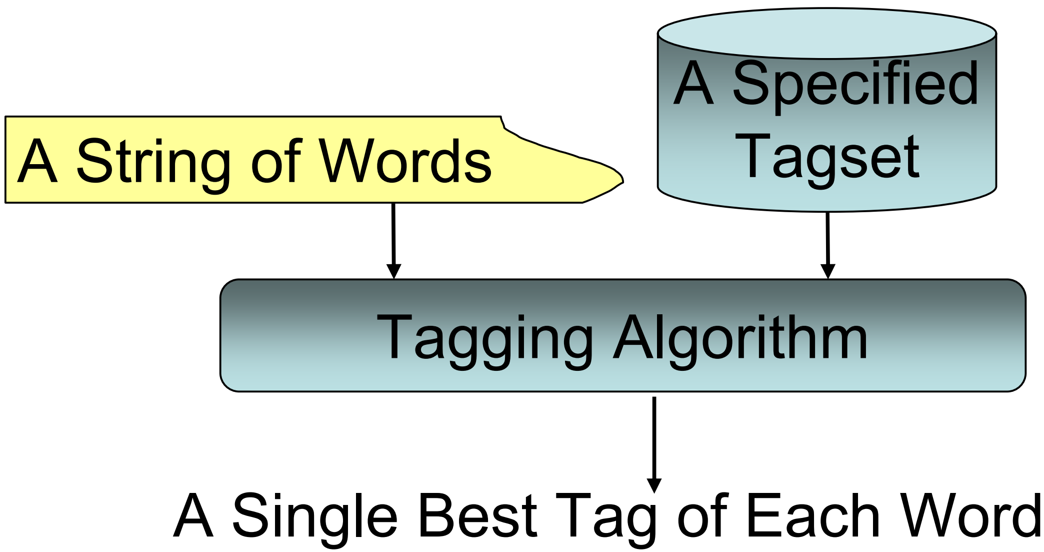
Disambiguation

- Resolve the ambiguities and chose the proper tag for the context
- Most English words are unambiguous (have only one tag) but many of the most common words are ambiguous
  - E.g.: "*can*" can be a (an auxiliary) verb or a noun
  - E.g.: statistics of Brown corpus

Unambiguous (1 tag)	**35,340**	
Ambiguous (2–7 tags)	**4,100**	
2 tags	3,760	
3 tags	264	
4 tags	61	
5 tags	12	
6 tags	2	
7 tags	1	("still")

- 11.5% word types are ambiguous
- But 40% tokens are ambiguous
(However, the probabilities of tags associated a word are not equal → many ambiguous tokens are easy to disambiguate)

Process of POS Tagging



VB DT NN .
Book that flight .

VBZ DT NN VB NN ?
Does that flight serve dinner ?

POS Tagging Algorithms

- Fall into One of Two Classes
- Rule-based Tagger
  - Involve a large database of hand-written disambiguation rules
    - E.g. a rule specifies that an ambiguous word is a noun rather than a verb if it follows a determiner
    - ENGTWOL: a simple rule-based tagger based on the **constraint grammar** architecture
- Stochastic/Probabilistic Tagger
  - Use a training corpus to compute the probability of a given word having a given context
  - E.g.: the HMM tagger chooses the best tag for a given word (maximize the product of **word likelihood** and **tag sequence probability**)

POS Tagging Algorithms

- Transformation-based/Brill Tagger
  - A hybrid approach
  - Like *rule-based approach*, determine the tag of an ambiguous word based on rules
  - Like *stochastic approach*, the rules are automatically included from previous tagged training corpus with the machine learning technique

Rule-based POS Tagging

- Two-stage architecture
  - **First stage**: Use a dictionary to assign each word a list of potential part-of-speech
  - **Second stage**: Use large lists of hand-written disambiguation rules to winnow down this list to a single part-of-speech for each word

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**

An example for
The ENGTOWL tagger

A set of 1,100 constraints can be applied to the input sentence

Rule-based POS Tagging

- Simple lexical entries in the ENGTWOL lexicon

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
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

past participle

Rule-based POS Tagging

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

Example:

one

It isn't **that** odd! A

ADV

I consider **that** odd. NUM

Compliment

HMM-based Tagging

- Also called *Maximum Likelihood Tagging*
  - Pick the most-likely tag for a word

- For a given sentence or words sequence , an HMM tagger chooses the tag sequence that maximizes the following probability

$$\text{tag}_i = \arg\max_i P(\text{word} \mid \text{tag}_i) \cdot P(\text{tag} \mid \text{previous } n-1 \text{ tags})$$

word/lexical likelihood

tag sequence probability

N-gram HMM tagger

HMM-based Tagging

- Assumptions made here
  - Words are independent of each other
    - A word's identity only depends on its tag
  - "Limited Horizon" and "Time Invariant" ("Stationary")
    - A word's tag only depends on the previous tag (limited horizon) and the dependency does not change over time (time invariance)
    - Time invariance means the tag dependency won't change as tag sequence appears different positions of a sentence

HMM-based Tagging

- Apply bigram-HMM tagger to choose the best tag for a given word
  - Choose the tag t_i for word w_i that is most probable given the previous tag t_{i-1} and current word w_i

$$t_i = \arg\max_j P\left(t_j \middle| t_{i-1}, w_i\right)$$

  - Through some simplifying Markov assumptions

$$t_i = \arg\max_j \underbrace{P\left(t_j \middle| t_{i-1}\right)}_{\text{tag sequence probability}} \underbrace{P\left(w_i \middle| t_j\right)}_{\text{word/lexical likelihood}}$$

HMM-based Tagging

- Apply bigram-HMM tagger to choose the best tag for a given word

$$t_i = \arg\max_j P\left(t_j \middle| t_{i-1}, w_i\right)$$

$$= \arg\max_j \frac{P\left(t_j, w_i \middle| t_{i-1}\right)}{P\left(w_i \middle| t_{i-1}\right)}$$

The same for all tags

$$= \arg\max_j P\left(t_j, w_i \middle| t_{i-1}\right)$$

$$= \arg\max_j P\left(w_i \middle| t_{i-1}, t_j\right) P\left(t_j \middle| t_{i-1}\right)$$

The probability of a word only depends on its tag

$$= \arg\max_j P\left(w_i \middle| t_j\right) P\left(t_j \middle| t_{i-1}\right) = \arg\max_j P\left(t_j \middle| t_{i-1}\right) P\left(w_i \middle| t_j\right)$$

HMM-based Tagging

- Example: Choose the best tag for a given word

Secretariat/NNP is /VBZ expected/VBN to/TO race/VB tomorrow/NN

to/TO race/???

$$\overset{0.34}{P(\text{VB}|\text{TO})}\ \overset{0.00003}{P(\text{race}|\text{VB})}=0.00001$$

$$\overset{0.021}{P(\text{NN}|\text{TO})}\ \overset{0.00041}{P(\text{race}|\text{NN})}=0.000007$$

Pretend that the previous
word has already tagged

HMM-based Tagging

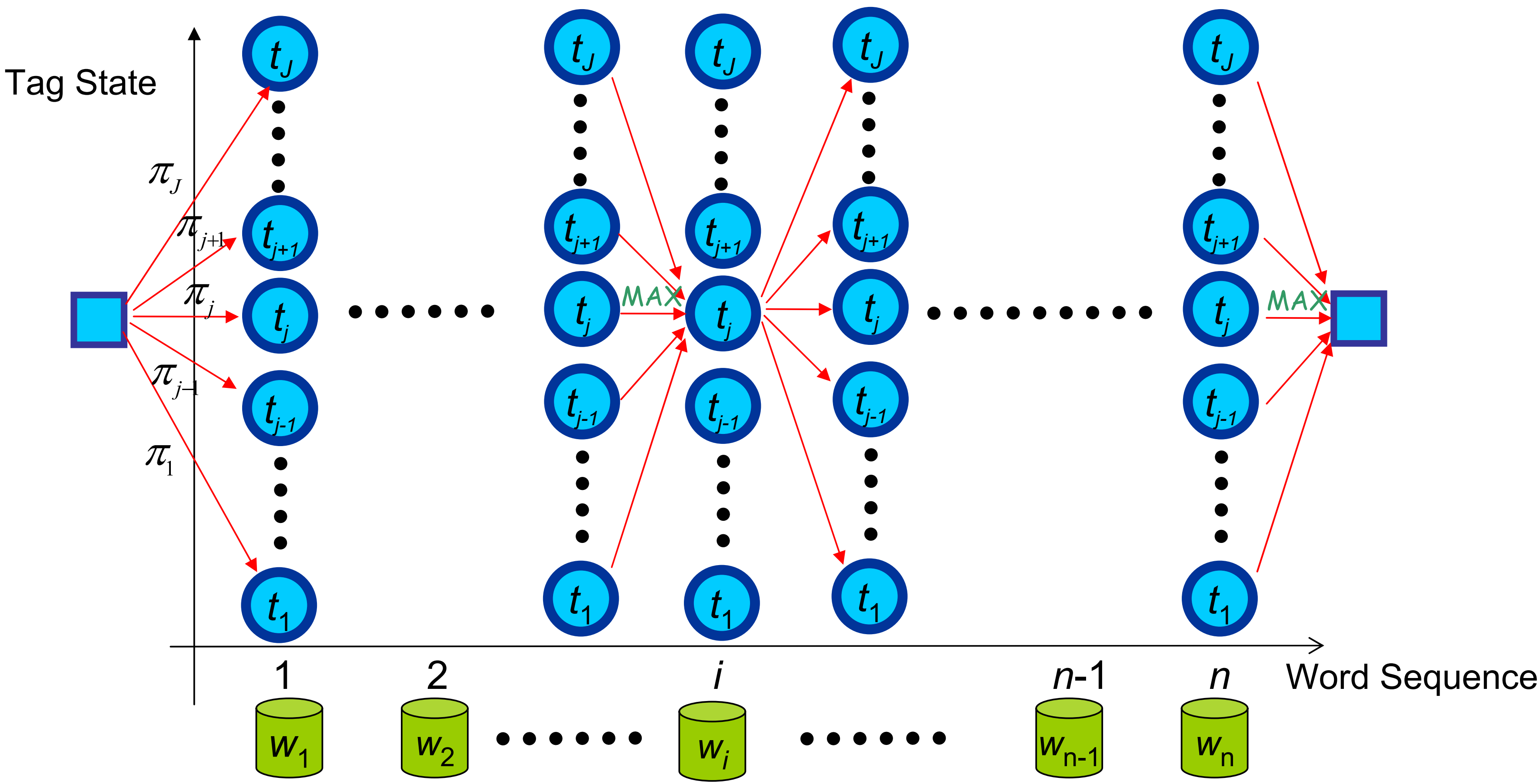
- Apply bigram-HMM tagger to choose the best sequence of tags for a given sentence

$$
\begin{aligned}
\hat{T} &= \arg\max_{T} P(T \mid W) \\
&= \arg\max_{T} \frac{P(T)P(W \mid T)}{\cancel{P(W)}} \\
&= \arg\max_{T} P(T)P(W \mid T) \\
&= \arg\max_{t_1, t_2, \ldots, t_n} P(t_1, t_2, \ldots, t_n)P(w_1, w_1, \ldots, w_n \mid t_1, t_2, \ldots, t_n) \\
&= \arg\max_{t_1, t_2, \ldots, t_n} \prod_{i=1}^{n} \left[P(t_i \mid t_1, t_2, \ldots, t_{i-1}) P(w_i \mid w_1, \ldots, w_{i-1}, t_1, t_2, \ldots, t_n) \right] \\
&= \arg\max_{t_1, t_2, \ldots, t_n} \prod_{i=1}^{n} \left[P(t_i \mid t_1, t_2, \ldots, t_{i-1}) P(w_i \mid t_i) \right]
\end{aligned}
$$

The probability of a word only depends on its tag

HMM-based Tagging

- The Viterbi algorithm for the bigram-HMM tagger



HMM-based Tagging

- The Viterbi algorithm for the bigram-HMM tagger

$$1.\ \text{Initialization}\ \delta_1(k) = \pi_k P(w_1 | t_k),\ 1 \le k \le J$$

$$2.\ \text{Induction}\ \delta_i(j) = \left[\max_i \delta_{i-1}(k) P(t_j | t_k)\right] P(w_i | t_j),\quad 2 \le i \le n,\ 1 \le k \le J$$

$$\psi_i(j) = \underset{1 \le j \le J}{\operatorname{argmax}} \left[\delta_{i-1}(k) P(t_j | t_k)\right]$$

$$3.\ \text{Termination}\quad X_n = \underset{1 \le j \le J}{\operatorname{argmax}}\, \delta_n(j)$$

$$\text{for i} := n\text{-1 to 1 step -1 do}$$

$$X_i = \psi_i(X_{i+1})$$

$$\text{end}$$

HMM-based Tagging

- Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence
  - **When trigram model is used**

$$\hat{T} = \arg\max_{t_1, t_2, \ldots, t_n} \left[P(t_1) P(t_2 | t_1) \prod_{i=3}^{n} P(t_i | t_{i-2}, t_{i-1}) \right] \left[\prod_{i=1}^{n} P(w_i | t_i) \right]$$

  - Maximum likelihood estimation based on the relative frequencies observed in the pre-tagged training corpus (labeled data)

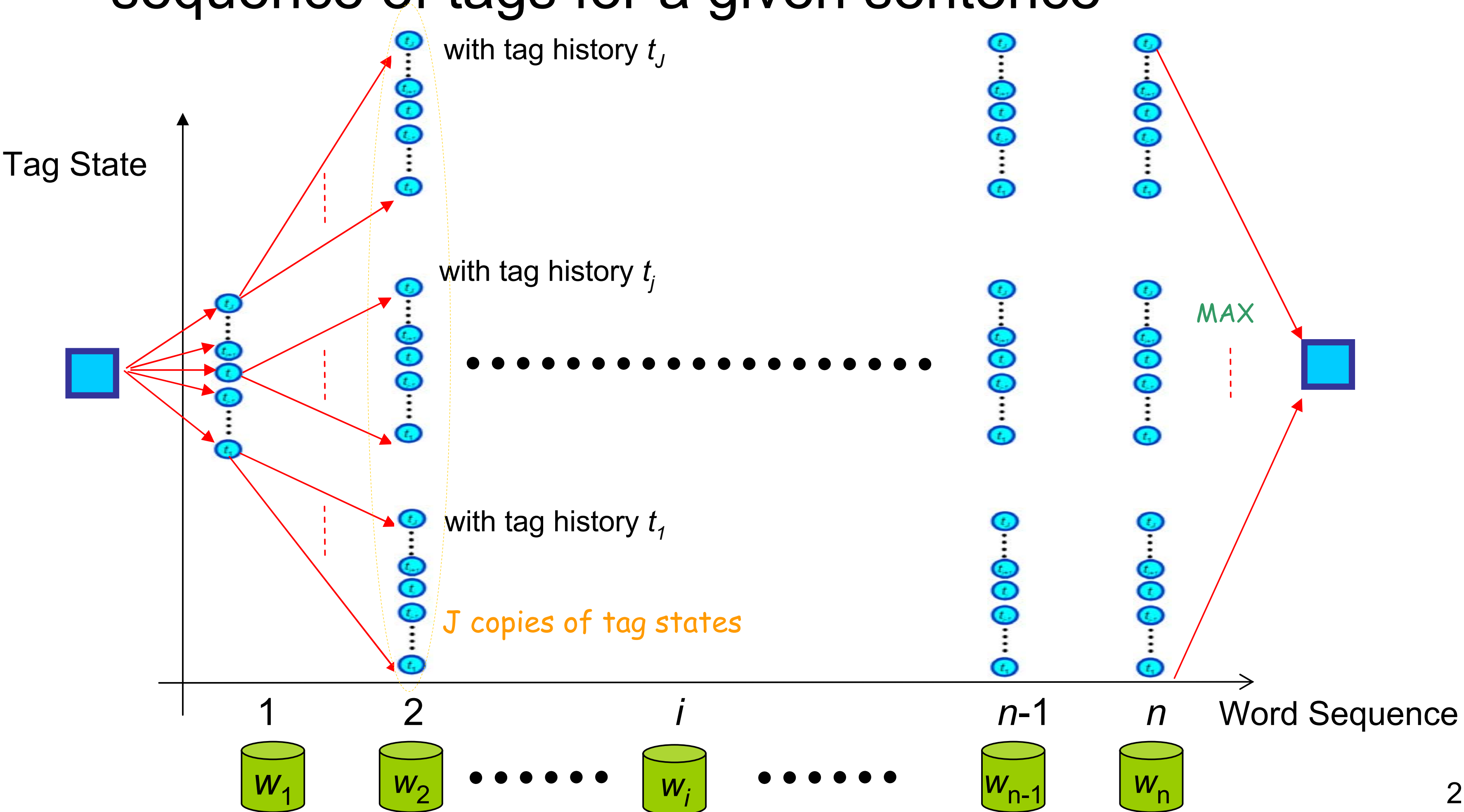
$$P(t_i | t_{i-2}, t_{i-1}) = \frac{c(t_{i-2} t_{i-1} t_i)}{c(t_{i-2} t_{i-1} t_i)}$$

Smoothing is needed !

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

HMM-based Tagging

- Apply trigram-HMM tagger to choose the best sequence of tags for a given sentence



HMM-based Tagging

- Probability re-estimation based on unlabeled data
  - EM (Expectation-Maximization) algorithm is applied
    - Start with a dictionary that lists which tags can be assigned to which words
      - » word likelihood function cab be estimated
      - » tag transition probabilities set to be equal
    - EM algorithm learns (re-estimates) the word likelihood function for each tag and the tag transition probabilities
  - However, a tagger trained on hand-tagged data worked better than one trained via EM

Transformation-based Tagging

- Also called Brill tagging
  - An instance of Transformation-Based Learning (TBL)
- **Spirits**
  - Like the **rule-based approach**, TBL is based on rules that specify what tags should be assigned to what word
  - Like the **stochastic approach**, rules are automatically induced from the data by the machine learning technique
- Note that TBL is a supervised learning technique
  - It assumes a pre-tagged training corpus

Transformation-based Tagging

- How the TBL rules are learned
  - Three major stages
    - Label every word with its most-likely tag using a set of tagging rules
    - Examine every possible transformation (rewrite rule), and select the one that results in the most improved tagging (supervised!)
    - Re-tag the data according this rule
  - The above three stages are repeated until some stopping criterion is reached
    - Such as insufficient improvement over the previous pass

Transformation-based Tagging

- Example

P(NN|race)=0.98
P(VB|race)=0.02

So, race will be initially coded as NN
(label every word with its most-likely tag)

1. is/VBZ expected/VBN to/To race/NN tomorrow/NN

2. the/DT race/NN for/IN outer/JJ space/NN

Refer to the correct tag Information of each word, and find the tag of *race* in "1" is wrong

Learn/pick a most suitable transformation rule: (by examining every possible transformation)

Change NN *to* VB *while the previous tag is* TO

Rewrite rule: expected/VBN to/To race/NN → expected/VBN to/To race/VB

Transformation-based Tagging

- Templates (abstracted transforms)
  - The set of possible transformation may be infinite
    - Should limit the set of transformations
    - The design of a small set of templates is needed

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

Brill's templates. Each begins with "Change tag a to tag b when ...."

Verb, 3sg, Present

Modal verbs (should, can,...)

#	Change tags From	To	Condition	Example
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

Verb, past participle

Rules learned by Brill's original tagger

Transformation-based Tagging

- Templates (abstracted transforms)

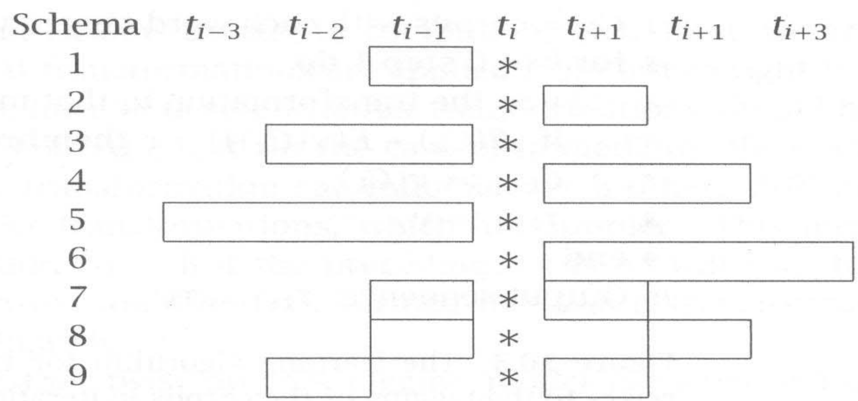


Table 10.7 Triggering environments in Brill's transformation-based tagger. Examples: Line 5 refers to the triggering environment "Tag t^j occurs in one of the three previous positions"; Line 9 refers to the triggering environment "Tag t^j occurs two positions earlier and tag t^k occurs in the following position."

Source tag	Target tag	Triggering environment
NN	VB	previous tag is TO
VBP	VB	one of the previous three tags is MD
JJR	RBR	next tag is JJ
VBP	VB	one of the previous two words is *n't*

Table 10.8 Examples of some transformations learned in transformation-based tagging.

Transformation-based Tagging

- Algorithm

```
function TBL(corpus) returns transforms-queue
  INTIALIZE-WITH-MOST-LIKELY-TAGS(corpus)
  until end condition is met do
    templates ← GENERATE-POTENTIAL-RELEVANT-TEMPLATES
    best-transform ← GET-BEST-TRANSFORM(corpus, templates)
    APPLY-TRANSFORM(best-transform, corpus)
    ENQUEUE(best-transform-rule, transforms-queue)
  end
  return(transforms-queue)
```

```
function GET-BEST-TRANSFORM(corpus, templates) returns transform
  for each template in templates
    (instance, score) ← GET-BEST-INSTANCE(corpus, template)
    if (score > best-transform.score) then best-transform ← (instance, score)
  return(best-transform)
```

Get best instance for each transformation

```
function GET-BEST-INSTANCE(corpus, template) returns transform
  for from-tag ← from tag−1 to tag−n do
   for to-tag ← from tag−1 to tag−n do
    for pos ← from 1 to corpus-size do
     if (correct-tag(pos) == to-tag && current-tag(pos) == from-tag)
        num-good-transforms(current-tag(pos−1))++
     elseif (correct-tag(pos)==from-tag && current-tag(pos)==from-tag)
        num-bad-transforms(current-tag(pos−1))++
    end
    best-Z ← ARGMAX_t(num-good-transforms(t) - num-bad-transforms(t))
    if(num-good-transforms(best-Z) - num-bad-transforms(best-Z)
               > best-instance.Z) then
      best-instance ← "Change tag from from-tag to to-tag
                          if previous tag is best-Z"
  return(best-instance)
```

for all combinations of tags

```
procedure APPLY-TRANSFORM(transform, corpus)
for pos ← from 1 to corpus-size do
  if (current-tag(pos)==best-rule-from)
      && (current-tag(pos−1)==best-rule-prev))
    current-tag(pos) = best-rule-to
```

The **GET_BEST_INSTANCE** procedure in the example algorithm is "Change tag from X to Y if the previous tag is Z".

Multiple Tags and Multi-part Words

- Multiple tags
  - A word is ambiguous between multiple tags and it is impossible or very difficult to disambiguate, so multiple tags is allowed, e.g.
    - adjective versus preterite versus past participle (JJ/VBD/VBN)
    - adjective versus noun as prenominal modifier (JJ/NN)
- Multi-part words
  - Certain words are split or some adjacent words are treated as a single word

would/MD n't/RB   Children/NNS 's/POS

in terms of (in/II31 terms/II32 of/II33)

Tagging of Unknown Words

- Simplest unknown-word algorithm
  - Pretend that each unknown word is ambiguous among all possible tags, with equal probability
  - Must rely solely on the contextual POS-trigram to suggest the proper tag
- Slightly more complex algorithm
  - Based on the idea that the probability distribution of tags over unknown words is very similar to the distribution of tags over words that occurred only once in a training set Nouns or Verbs
  - The likelihood for an unknown word is determined by the average of the distribution over all singleton in the training set (similar to Good-Turing?) $P(w_i|t_i)$?

Tagging of Unknown Words

- Most-powerful unknown-word algorithm
  - Hand-designed features
    - The information about how the word is spelled (inflectional and derivational features), e.g.:
      - Words end with s ($\rightarrow$plural nouns)
      - Words end with ed ($\rightarrow$past participles)
    - The information of word capitalization (initial or non-initial) and hyphenation

$$P\left(w_i \middle| t_i\right) = p\left(\text{unknown} - \text{word} \middle| t_i\right) \cdot p\left(\text{captial} \middle| t_i\right) \cdot p\left(\text{endings/hyph} \middle| t_i\right)$$

  - Features induced by machine learning
    - E.g.: TBL algorithm uses templates to induce useful English inflectional and derivational features and hyphenation

The first N letters of the word
The last N letters of the word

Evaluation of Taggers

- Compare the tagged results with a human labeled **Gold Standard** test set in percentages of correction
  - Most tagging algorithms have an accuracy of around 96~97% for the sample tagsets like the Penn Treebank set
  - Upper bound (ceiling) and lower bound (baseline)
    - *Ceiling*: is achieved by seeing how well humans do on the task
      - A 3~4% margin of error
    - *Baseline*: is achieved by using the unigram most-like tags for each word
      - 90~91% accuracy can be attained

Error Analysis

- Confusion matrix

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	-	.2			.7		
JJ	.2	-	**3.3**	2.1	1.7	.2	**2.7**
NN		**8.7**	-				.2
NNP	.2	**3.3**	**4.1**	-	.2		
RB	**2.2**	2.0	.5		-		
VBD		.3	.5			-	**4.4**
VBN		**2.8**				**2.6**	-

- Major problems facing current taggers
  - NN (noun) versus NNP (proper noun) and JJ (adjective)
  - RP (particle) versus RB (adverb) versus JJ
  - VBD (past tense verb) versus VBN (past participle verb) versus JJ

Applications of POS Tagging

- Tell what words are likely to occur in a word's vicinity
  - E.g. the vicinity of the possessive or person pronouns
- Tell the pronunciation of a word
  - DIScount (noun) and disCOUNT (verb) …
- Advanced ASR language models
  - Word-class N-grams
- Partial parsing
  - A simplest one: find the noun phrases (names) or other phrases in a sentence

Applications of POS Tagging

- Information retrieval
  - Word stemming
  - Help select out nouns or important words from a doc
  - Phrase-level information

    United, States, of, America → "United States of America"
    secondary, education → "secondary education"

    - Phrase normalization

      Book publishing, publishing of books

- Information extraction
  - Semantic tags or categories

Applications of POS Tagging

- Question Answering
  - Answer a user query that is formulated in the form of a question by return an appropriate noun phrase such as a location, a person, or a date
    - E.g. "Who killed President Kennedy?"

In summary, the role of taggers appears to be a fast lightweight component that gives sufficient information for many applications

- But not always a desirable preprocessing stage for all applications
- Many probabilistic parsers are now good enough !

Class-based N-grams

- Use the lexical tag/category/class information to augment the *N*-gram models

$$P\left(w_n \middle| w_{n-N+1}^{n-1}\right) = P\left(w_n \middle| c_n\right) P\left(c_n \middle| c_{n-N+1}^{n-1}\right)$$

prob. of a word given the tag

prob. of a word given the tag

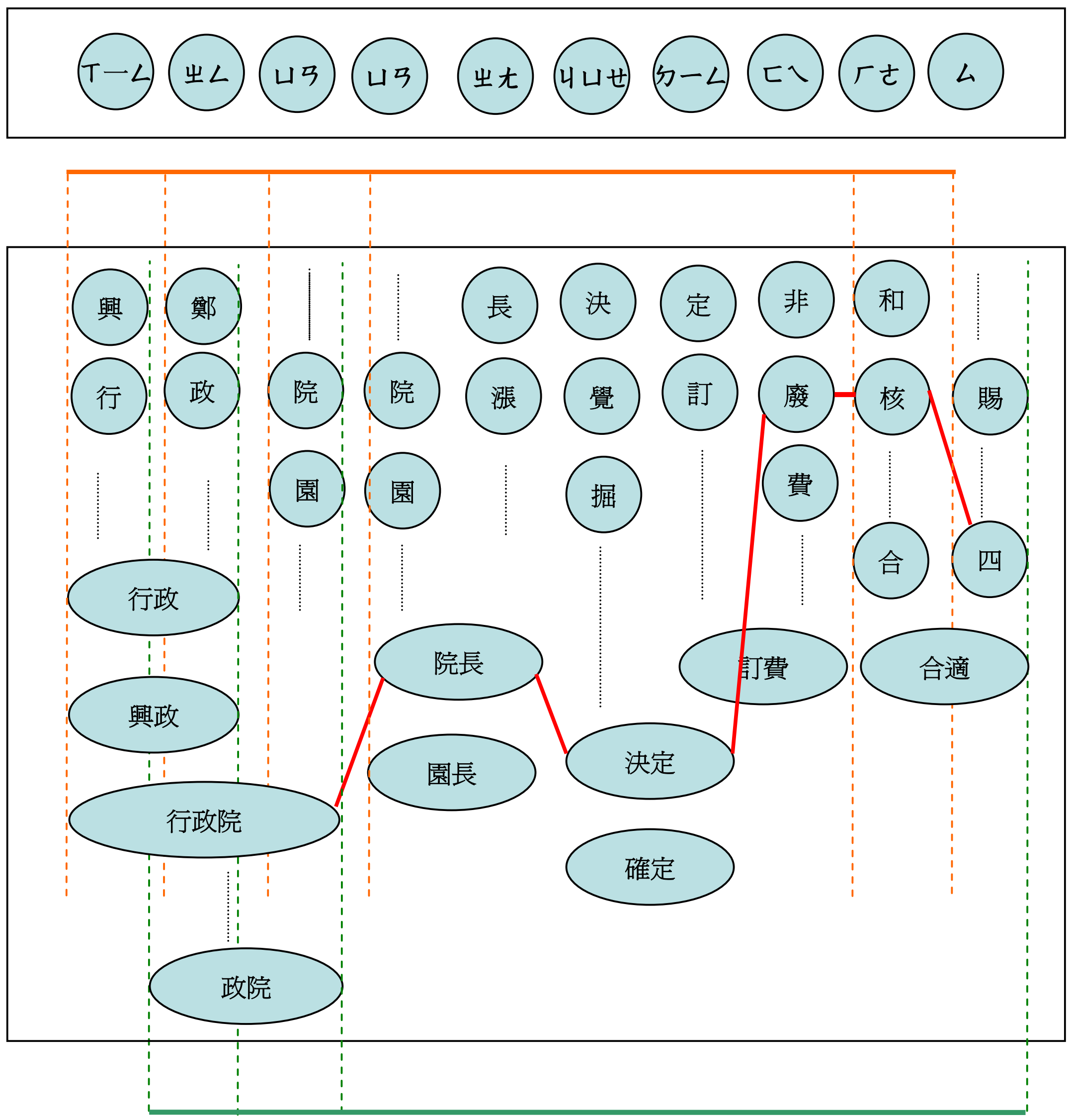
  - Maximum likelihood estimation

$$P\left(w_i \middle| c_j\right) = \frac{C(w)}{C(c)}$$

$$P\left(c_j \middle| c_k\right) = \frac{C\left(c_k c_j\right)}{\sum_l C\left(c_l c_j\right)}$$

Constraints: a word may only belong to one lexical category

行政院院長決定廢核四



ㄒㄧㄥ
ㄓㄥ
ㄩㄢ
ㄩㄢ
ㄓㄤ
ㄐㄩㄝ
ㄉㄧㄥ
ㄈㄟ
ㄏㄜ
ㄙ
興
鄭
長
決
定
非
和
行
政
院
院
漲
覺
訂
廢
核
賜
園
園
掘
費
合
四
行政
院長
訂費
合適
興政
決定
園長
行政院
確定
政院