Part of Speech Tagging

Rule-Based Approaches

Task description – more formal

- Given a sentence and its tokens t_i, assign a single label $l \in L$ with L being the tagset (e.g. Penn Treebank, STTS) to every t_i
- This is a **structural problem**, where our input is a sequence (“a list of tokens”) and the output is a sequence (“a list of labels”), and:
  - Both sequences have the same length
    - (This is not the case for OCR or speech recognition)

WORD	tag
the	**DET**
koala	**N**
put	**V**
the	**DET**
keys	**N**
on	**P**
the	**DET**
table	**N**

Rule-Based POS-Tagging

- The system “ENGTWOL”
  - Start with a dictionary
  - Assign all possible tags to words from the dictionary
  - Write rules by hand to selectively remove tags
  - Leaving the correct tag for each word.

Start With a Dictionary

• she:	PRP
• promised:	VBN,VBD
• to	TO
• back:	VB, JJ, RB, NN
• the:	DT
• bill:	NN, VB

- Etc… for the ~100,000 English words with more than 1 tag

Assign Every Possible Tag

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Write Rules to Eliminate Tags

Eliminate VBN if VBD is an option and when (VBN | VBD) follows "<start> PRP"

			NN		
			RB		
	~~VBN~~		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Stage 1 of ENGTWOL Tagging

- First Stage: Run words through FST morphological analyser 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**

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)             ;followed by End-Of-Sentence (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
```

Rule Learning

Brills Tagger

Rule-Learning: Transformation-Based (Brill) Tagging

- Developed by Brill in 1995 (“Brills Tagger”, accuracy over 96%)
- Error-Driven Approach to learn correction rules
- Was also used for different NLP areas, e.g.
  - Text chunking
  - Parsing
  - Named Entity Recognition

Transformation-Based (Brill) Tagging

- Combines Rule-based and Stochastic Tagging
  - Rule-based because rules are used to specify tags in a certain environment
  - Stochastic approach because we use a tagged corpus to find the best performing rules

    → *Rules are learned from data*
- Input:
  - Tagged corpus
  - Dictionary (*with most frequent tags*)
  - Rule Templates

Templates for TBL

- Example for a template:
  - „Preceding word is tagged y“ → change current tag to z
  - E.g.:
    - „Preceding word is tagged DET“ → change current tag to TO
    - „Preceding word is tagged VBN“ → change current tag to DET

  → Results in Tagset^2 rules of this template.
  - For each rule, measure:
    - How many errors does it correct: e_C
    - How many new errors does it introduce: e_N
    - Score a rule with: $s_{Rule} = e_C - e_N$

➔ After all rules of all templates are checked, use the one with the highest score s_{Rule}

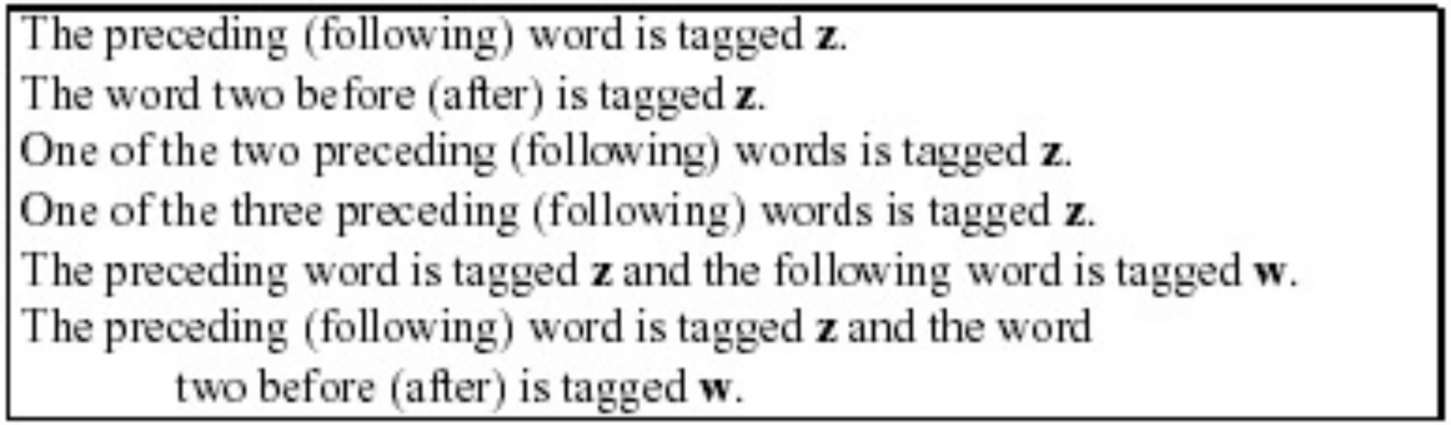
TBL Tagging Algorithm

- Step 1: Label every word with most likely tag (from dictionary)
- Step 2: Check every possible transformation & select one which most improves tag accuracy (cf Gold)
- Step 3: Re-tag corpus applying this rule, and add rule to the end of the rule set
- Repeat 2-3 until some stopping criterion is reached, e.g., X% correct with respect to training corpus
- RESULT: Ordered set of transformation rules to use on new data tagged only with most likely POS tags

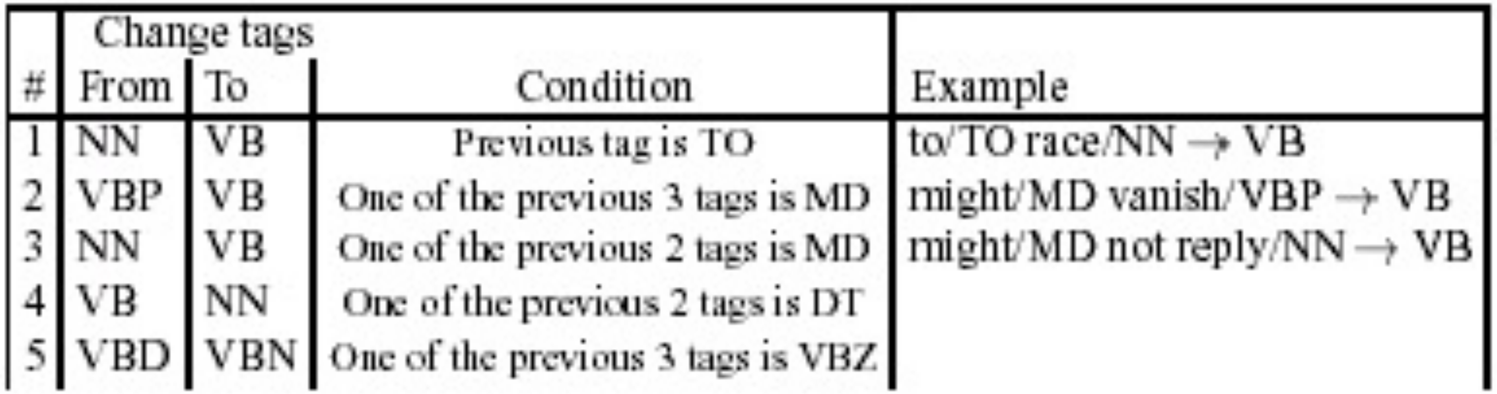
Sample TBL Rule Application

- Labels every word with its most-likely tag
  - E.g. *race* occurrences in the Brown corpus:
    - *P(NN|race) = .98*
    - *P(VB|race)= .02*
    - *is/VBZ expected/VBN to/TO race/NN tomorrow/NN*
- Then TBL applies the following rule
  - “Change NN to VB when previous tag is TO”
    … *is/VBZ expected/VBN to/TO race/NN tomorrow/NN*
    becomes
    … *is/VBZ expected/VBN to/TO race/VB tomorrow/NN*

Templates for TBL



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**.



	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	

TBL Unknown Words

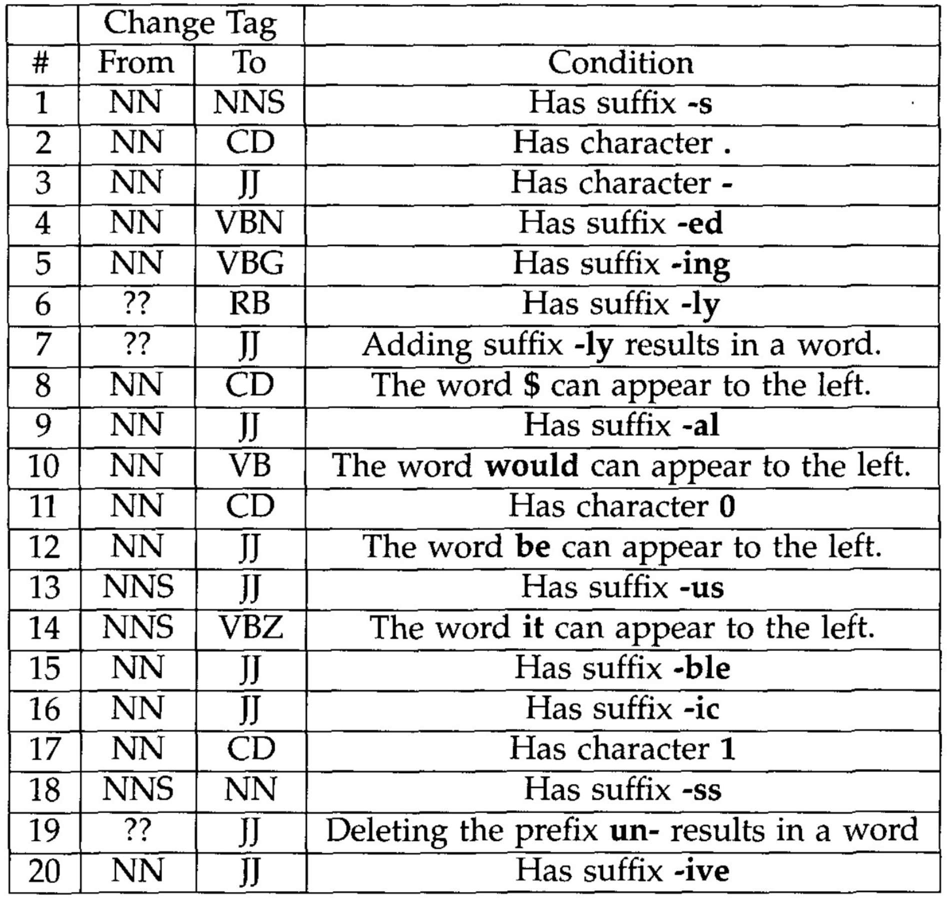
Naive assumption

- an unknown word is "**proper noun**" if the word is capitalized
- and "**common noun**" otherwise

Test possible transformations for unknown words to improve tag accuracy

- e.g. change the tag of an unknown word (from X) to Y if:
  - The first (last) (1,2,3,4) characters of the word are x.

The first 20 transformations for unknown words



	Change Tag		
#	From	To	Condition
1	NN	NNS	Has suffix **-s**
2	NN	CD	Has character **.**
3	NN	JJ	Has character **-**
4	NN	VBN	Has suffix **-ed**
5	NN	VBG	Has suffix **-ing**
6	??	RB	Has suffix **-ly**
7	??	JJ	Adding suffix **-ly** results in a word.
8	NN	CD	The word **$** can appear to the left.
9	NN	JJ	Has suffix **-al**
10	NN	VB	The word **would** can appear to the left.
11	NN	CD	Has character **0**
12	NN	JJ	The word **be** can appear to the left.
13	NNS	JJ	Has suffix **-us**
14	NNS	VBZ	The word **it** can appear to the left.
15	NN	JJ	Has suffix **-ble**
16	NN	JJ	Has suffix **-ic**
17	NN	CD	Has character **1**
18	NNS	NN	Has suffix **-ss**
19	??	JJ	Deleting the prefix **un-** results in a word
20	NN	JJ	Has suffix **-ive**

TBL Issues

- Problem: Could keep applying (new) transformations ad infinitum
- Problem: Rules are learned in ordered sequence
- Problem: Rules may interact
- But: Rules are compact and can be inspected by humans