ILLINOIS INSTITUTE OF TECHNOLOGY

Parts of Speech and Sequence Tagging

CS-585

Natural Language Processing

Derrick Higgins

Sequence labeling

- We can treat a text as a bunch of lexical units (a bag of words)
- …Or we can also treat it as a sequence
  - Typically for shorter texts - e.g., one sentence at a time

- Similarly, we can make predictions at the text level, or we can make predictions associated with each word in the text, incorporating information from the surrounding sequence
- These are *sequence labeling* tasks
  - Very common in NLP, but also in bioinformatics

Sequence labeling

- Sequence labeling involves associating a predicted label or tag with each word

```
AT0 AJ0   AJ0 NN1  VVD   PRP AT0 AJ0 NN1
```
The quick brown fox jumped over the lazy dog

```
ES    ES   ES ES   ES EN EN    EN
```
bueno porque le gusta decir hi good morning

Sequences and Text Spans

- Sequence labeling methods can also be used to label spans of text, by specifying word-level tags that encode whether the word is inside or outside of the span

Place

Thank you for visiting the [Chicago Public Library] .

O O O O O I I I

- Problem – this is ambiguous

Place Place

Thank you for visiting the [Chicago] [Public Library] .

IOB Representations (and friends)

Tag type	Use
`I`	The word is *inside* the span (and more specific tags do not apply)
`O`	The word is *outside* the span
`B`	The word is at the *beginning* of the span
`E/L`	The word is at the *end* of the span (the *last* word)
`S/U`	The word is a *singleton* / *unit* span

BIOES / BILUO Tagging Scheme

IOB / IOB2 Tagging Scheme

IOB Tagging Schemes

IOB

O	O	O	O	O	B-Place	I-Place	I-Place	O
Thank	you	for	visiting	the	Chicago	Public	Library	.

I-Place	O	O	O	O	B-Place	I-Place	O
Albany	is	the	capital	of	New	York	.

IOB2

B-Place	O	O	O	O	B-Place	I-Place	O
Albany	is	the	capital	of	New	York	.

IOB Tagging Schemes

BIOES

O	O	O	O	O	B-Place	I-Place	E-Place	O
Thank	you	for	visiting	the	Chicago	Public	Library	.

S-Place	O	O	O	O	B-Place	E-Place	O	
Albany	is	the	capital	of	New	York	.	

BILUO

U-Place	O	O	O	O	B-Place	L-Place	O	
Albany	is	the	capital	of	New	York	.	

PART OF SPEECH TAGGING

Parts of Speech

- Generally speaking, the "grammatical type" of word:
  - Verb, Noun, Adjective, Adverb, Article, …
- We also include inflections:
  - Verbs: Tense, number, …
  - Nouns: Number, proper/common, …
  - Adjectives: comparative, superlative, …
  - …

POS tagging key for NLP

- Ambiguity: *book can be a noun or a verb*
- Useful input for parsing
  - *"book a ticket" vs. "book of the century"*
- Disambiguate meaning even without parsing
  - *Is a text about literature or about travel?*

Tagsets

- *Tagsets*: canonical definitions of parts of speech
  - May be more or less granular
  - Typically language-specific (although there are universals – nouns, verbs, …)

Tagset	Language	Number of Tags
CLAWS8	English	170
Penn Treebank	English	36
BNC/C5	English	57
Universal POS tags	"universal"	17
Stuttgart Tübingen Tagset	German	54
Chinese PTB	Chinese	34
…		

BNC Parts of Speech

Nouns:

`NN0` Common noun, neutral for number (e.g. *aircraft)*

`NN1` Singular common noun (e.g. *pencil, goose, time)*

`NN2` Plural common noun (e.g. *pencils, geese, times)*

`NP0` Proper noun (e.g. *London, Michael, Mars, IBM)*

Pronouns:

`PNI` Indefinite pronoun (e.g. *none, everything, one)*

`PNP` Personal pronoun (e.g. *I, you, them, ours)*

`PNQ` *Wh*-pronoun (e.g. *who, whoever, whom)*

`PNX` Reflexive pronoun (e.g. *myself, itself, ourselves)*

Verbs:

`VVB` finite base form of lexical verbs (e.g. *forget, send, live, return)*

`VVD` past tense form of lexical verbs (e.g. *forgot, sent, lived)*

`VVG` *-ing* form of lexical verbs (e.g. *forgetting, sending, living)*

`VVI` infinitive form of lexical verbs (e.g. *forget, send, live, return)*

`VVN` past participle form of lexical verbs (e.g. *forgotten, sent, lived*

`VVZ` *-s* form of lexical verbs (e.g. *forgets, sends, lives, returns)*

`VBB` present tense of BE, except for *is*

…and so on: `VBD` `VBG` `VBI` `VBN` `VBZ`

`VDB` finite base form of DO: *do*

…and so on: `VDD` `VDG` `VDI` `VDN` `VDZ`

`VHB` finite base form of HAVE: *have, 've*

…and so on: `VHD` `VHG` `VHI` `VHN` `VHZ`

`VM0` Modal auxiliary verb (e.g. *will, would, can, could, 'll, 'd*)

Articles

`AT0` Article (e.g. *the*, *a*, *an*, *no*)

`DPS` Possessive determiner (e.g. *your*, *their*, *his*)

`DT0` General determiner (*this*, *that*)

`DTQ` *Wh*-determiner (e.g. *which*, *what*, *whose*, *whichever*)

`EX0` Existential *there*, i.e. occurring in "***there*** *is…*" or "***there*** *are…*"

Adjectives

`AJ0` Adjective (general or positive) (e.g. *good*, *old*, *beautiful*)

`AJC` Comparative adjective (e.g. *better*, *older*)

`AJS` Superlative adjective (e.g. *best*, *oldest*)

Adverbs

`AV0` General adverb (e.g. *often*, *well*, *longer* (adv.), *furthest*.

`AVP` Adverb particle (e.g. *up*, *off*, *out*)

`AVQ` *Wh*-adverb (e.g. *when*, *where*, *how*, *why*, *wherever*)

Miscellaneous:

`CJC` Coordinating conjunction (e.g. *and*, *or*, *but*)

`CJS` Subordinating conjunction (e.g. *although*, *when*)

`CJT` The subordinating conjunction *that*

`CRD` Cardinal number (e.g. *one*, *3*, *fifty-five*, *3609*)

`ORD` Ordinal numeral (e.g. *first*, *sixth*, *77th*, *last*)

`ITJ` Interjection or other isolate (e.g. *oh*, *yes*, *mhm*, *wow*)

`POS` The possessive or genitive marker *'s* or *'*

`TO0` Infinitive marker *to*

`PUL` Punctuation: left bracket - i.e. (or [

`PUN` Punctuation: general separating mark - i.e. . , ! , : ; - or ?

`PUQ` Punctuation: quotation mark - i.e. ' or "

`PUR` Punctuation: right bracket - i.e.) or]

`XX0` The negative particle *not* or *n't*

`ZZ0` Alphabetical symbols (e.g. *A*, *a*, *B*, *b*, *c*, *d*)

Task: Part-Of-Speech Tagging

- **Goal**: Assign the correct part-of-speech to each word (and punctuation) in a text.
- Example:

Two	old	men	bet	on	the	game	.
CRD	AJ0	NN2	VVD	PP0	AT0	NN1	PUN

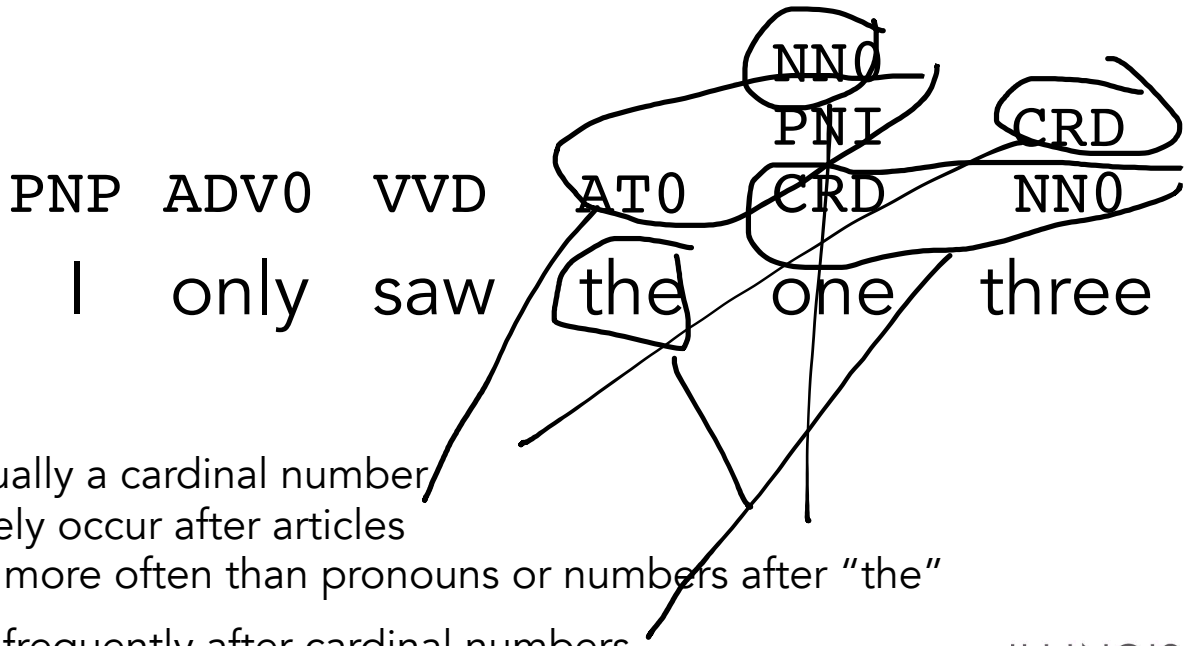
- Learn a *local* model of POS dependencies, usually from pretagged data
- **No** parsing!

Modeling considerations for POS tagging

- POS tagging is very similar to word-sense disambiguation, in that it involves making predictions at the word level, rather than the text level
  - One difference is that we need to assign a tag to *every* word
  - Another is that the predictive information tends to be much more "local"
- Not a bag-of-words problem!
  - Contextual features rather than document-level lexical features

Modeling considerations for POS tagging

- POS tagging (and sequence labeling in general) involves using
  - Bottom-up evidence related to the likelihood of tags
  - Top-down information about what tag sequences are valid/likely



"three" is usually a cardinal number
Pronouns rarely occur after articles
Nouns occur more often than pronouns or numbers after "the"

Nouns occur frequently after cardinal numbers

Global constraints for sequence labeling

O	O	O	O	O	B-Place	I-Place	I-Place	O
Thank	you	for	visiting	the	Chicago	Public	Library	.

O	O	O	O	O	I-Place	I-Place	B-Place	O
Thank	you	for	visiting	the	Chicago	Public	Library	.

O	O	O	O	O	B-Place	I-Company	I-Place	O
Thank	you	for	visiting	the	Chicago	Public	Library	.

Evaluation

- Part-of-speech tagging is "easy"
  - Just guessing the most frequent tag for each word (and "noun" for unknown words) gets over 90% accuracy*
  - As always, it's important to evaluate accuracy measures relative to some baseline
- POS tagging accuracy can suffer when models are applied to a different domain or genre than that used for training. E.g., performance for a model trained on Wall Street Journal text (Stanford MEMM)**:

Domain	WSJ	news-groups	reviews	blogs	answers	emails
Accuracy	96.8%	89.1%	91.4%	94.1%	88.9%	88.7%

*Jurafsky & Martin. *Speech and Language Processing.*
**Schnabel & Schutze (2013). *FLORS: Fast and Simple Domain Adaptation for Part-of-Speech Tagging.*

Evaluation

- *Accuracy* is a high-level measure of model performance for classification tasks, aligned with the loss function used for optimization (generally, cross-entropy)
  - *But it doesn't tell us how the model is doing on specific categories*
  - *And as we have seen, it can be misleading in cases with very unbalanced class distributions or simple baselines – e.g., a mediocre model gives us 90%, a great model give us 92%*
- Applies for a range of tasks (sequence labeling, text categorization, etc.)

Evaluation: confusion matrix

A tool to provide more insight into model performance is the confusion matrix: a table comparing the model's predictions to the true labels class by class

E.g., for sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Total correct = 100+150+90=340

Total items = 100+50+5+40+150+30+10+60+90=535

Accuracy = 340/535=63.6%

Evaluation: true positives

True positives: Items in a given class that are also predicted to be in that class.

E.g., for Neutral sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Evaluation: false positives

False positives: Items **not** in a given class that **are** predicted to be in that class.

E.g., for Neutral sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Evaluation: true negatives

True negatives: Items **not** in a given class that are **not** predicted to be in that class.

E.g., for Neutral sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Evaluation: false negatives

False negatives: Items in a given class that are **not** predicted to be in that class.

E.g., for Neutral sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Evaluation: precision

Precision: The proportion of predictions for a given class that are correct

Precision = True Positives ÷ (True Positives + False Positives)

For Neutral sentiment:

Predicted class

True class	Positive	Neutral	Negative
Positive	100	50	5
Neutral	40	150	30
Negative	10	60	90

Precision = 150 ÷ (150+50+60) = 57.7%

Evaluation: recall

Recall: The proportion of true labels for a given class that are correctly predicted

Recall = True Positives ÷ (True Positives + False Negatives)

For Neutral sentiment:

		Predicted class: Positive	Predicted class: Neutral	Predicted class: Negative
True class	Positive	100	50	5
	Neutral	40	150	30
	Negative	10	60	90

Recall = 150 ÷ (150+40+30) = 68.2%

Evaluation: F measure

F-measure is a balanced weighting of precision and recall, combining the two using the harmonic mean:

$$F_1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

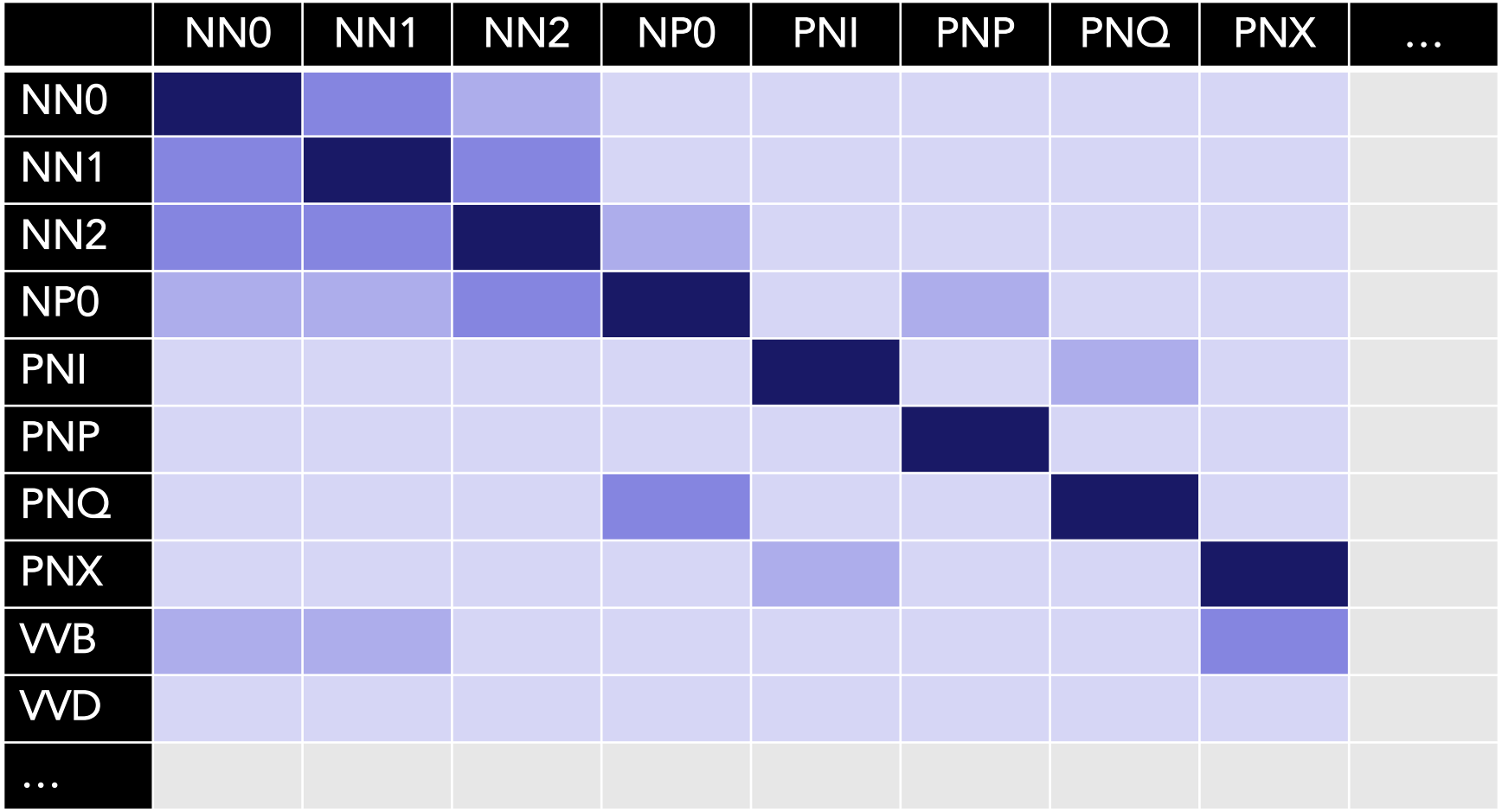
Technically, we refer to it as F_1, because there are a family of F measures with different weighting of precision vs. recall:

$$F_\beta = (1 + \beta^2)\frac{PR}{\beta^2 P + R}$$

$\beta > 1$ gives more importance to recall; $0 < \beta < 1$ gives more importance to precision

So for P=57.7%, R=68.2%: $F_1 = \frac{2 \times .577 \times .682}{.577 + .682} = .625$

Confusion matrix for POS tagging



Inference and search for sequence modeling

- Note that our search space has blown up, relative to text categorization tasks
  - For text categorization tasks, we might have to choose one of 2 classes for a text, or one of 200
  - For POS tagging, we might have to select a tag from a list of ~80, for each word in a sentence $\rightarrow 80^N$ possible tag assignments for a sentence of length *N*!
  - Generally for sequence labeling tasks, the search space is exponential in the length of the sequence
- Of course, most of these tag combinations are very unlikely.

Inference and search for sequence modeling

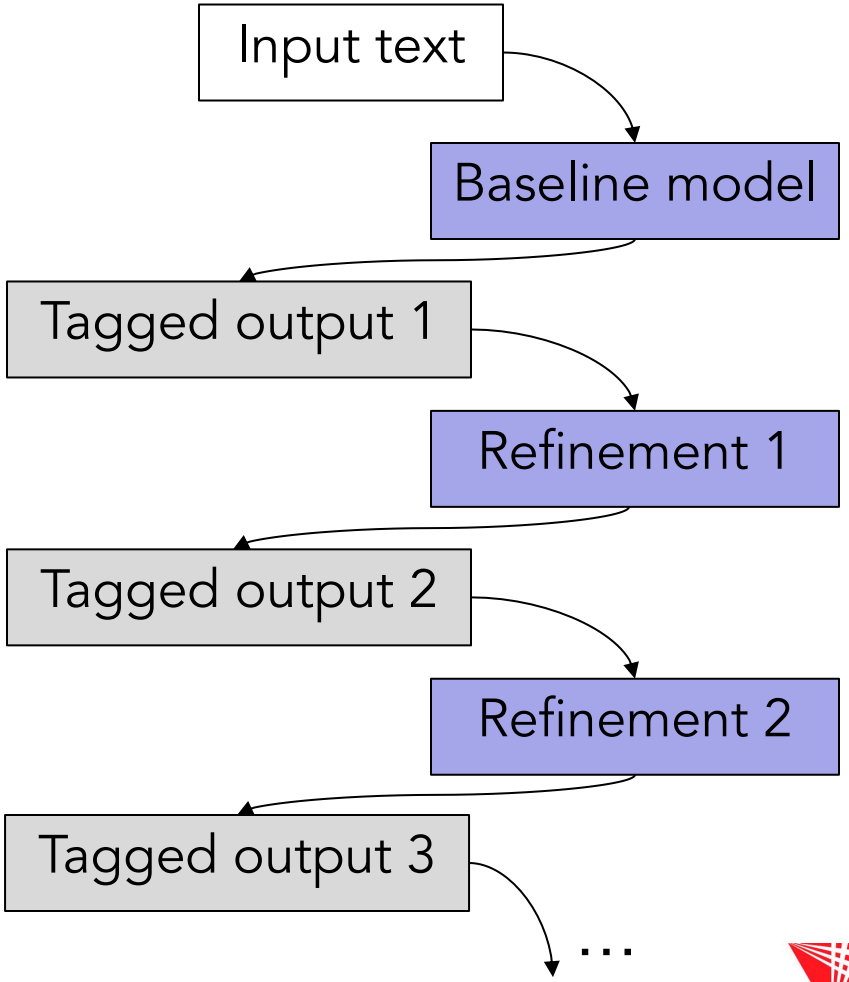
- We can make the search tractable in a few ways
  - Greedy search: commit to tag assignments one by one, and use them as context for the remaining assignments
  - Beam search: consider only a limited number of hypotheses for partial tag assignments, and discard the rest
  - Dynamic programming: store intermediate results in a data structure to reduce backtracking and transform the exponential search into a quadratic one

Transformation-based learning and the Brill tagger

- We'll get to probabilistic approaches to POS tagging next lecture
- But today we'll look at a very powerful tagging framework that is easy to understand and demonstrates a greedy approach to reducing the search space
- The Brill tagger (after Eric Brill)
  - *Transformation-based learning*

Transformation-based learning

- Idea: to get a good model
  1. Start with a mediocre model
  2. Identify the most common errors it makes
  3. Fix them
  4. Go to 2



Brill tagging using TBL

Initialize: Rule set *R* = ()

1. Label each word in the training data with its most frequent tag (from tag dictionary)
2. Calculate the **Δ**Accuracy for every transformation rule that could be applied
3. Select the rule with the highest **Δ**Accuracy and
   A. Append it to *R*
   B. Apply it to update tags previously assigned to words in the training data
4. If stopping criterion (target accuracy threshold, lack of improvement on development set for N iterations) not reached, go to step 2
5. Return *R* – a sequence of transformations that can be applied to label unseen data

Rule templates for the Brill tagger

- Transformation rules are if-then rules of the form

```
Replace tag A with tag B
    if condition C holds
```

- Rule templates are meta-patterns defining rule types, e.g.,

```
Replace tag A with tag B for w_i
    if w_{i-1}=X
Replace tag A with tag B for w_i
    if tag(w_{i-1})=X and tag(w_{i+1})=Y
```

Rule templates for the Brill tagger

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	

Rule templates for the Brill tagger

Change the tag of an unknown word (from X) to Y if:

1. Deleting the prefix (suffix) x, $|x| \leq 4$, results in a word (x is any string of length 1 to 4).
2. The first (last) (1,2,3,4) characters of the word are x.
3. Adding the character string x as a prefix (suffix) results in a word ($|x| \leq 4$).
4. Word w ever appears immediately to the left (right) of the word.
5. Character z appears in the word.

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.

Brill tagger: tradeoffs

- Pro
  - Simple way of incorporating both local context features and top-down information about consistency of tag sequences
  - Pretty good accuracy, especially on unknown words
- Con
  - Fairly slow at prediction time
  - Also fairly slow to learn
    - First 100 rules achieve 96.8% accuracy
      First 200 rules achieve 97.0% accuracy