

# CS674 Natural Language Processing

- **Last classes**
  - N-gram models
    - Smoothing
    - Backoff
- **Today**
  - Part-of-speech tagging
    - Transformation-based learning

## Part of speech tagging

“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

## Part of speech tagging

- **Assign the correct part of speech (word class) to each word/token in a document**

“The/DT planet/NN Jupiter/NNP and/CC its/PRP moons/NNS are/VBP in/IN effect/NN a/DT mini-solar/JJ system/NN ./, and/CC Jupiter/NNP itself/PRP is/VBZ often/RB called/VBN a/DT star/NN that/IN never/RB caught/VBN fire/NN ./.”
- **Needed as an initial processing step for a number of language technology applications**
  - Answer extraction in QA
  - Base step in identifying syntactic phrases for IR systems
  - Critical for word-sense disambiguation (WordNet apps)
  - Information extraction
  - ...

## Why is p-o-s tagging hard?

- **Ambiguity**
  - He will **race**/VB the car.
  - When will the **race**/NOUN end?
  - The boat **floated**/ VBN down the river sank.
- **Average of ~2 parts of speech for each word**
- **The number of tags used by different systems varies a lot. Some systems use < 20 tags, while others use > 400.**

## Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+,%, &amp;</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WPS	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>(' or ")</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>(' or ")</i>
PP	Personal pronoun	<i>I, you, he</i>	(	Left parenthesis	<i>( [ , { , &lt;</i>
PPS	Possessive pronoun	<i>your, one's</i>	)	Right parenthesis	<i>( [ , { , &gt;</i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>( ! ?)</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>( ; ... -&gt;)</i>
RP	Particle	<i>up, off</i>			

## Among easiest of NLP problems

- **State-of-the-art methods achieve ~97% accuracy.**
- **Simple heuristics can go a long way.**
  - ~90% accuracy just by choosing the most frequent tag for a word (MLE)
  - To improve reliability: *need to use some of the local context.*
- **But defining the rules for special cases can be time-consuming, difficult, and prone to errors and omissions**

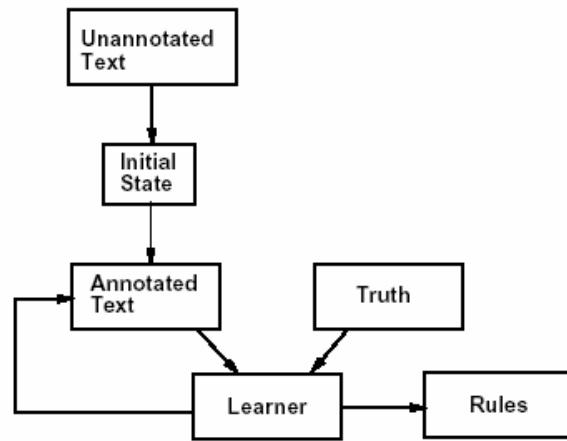
## Approaches

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. **probabilistic:** 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
3. **hybrid corpus-/rule-based:** E.g. transformation-based tagger (Brill tagger); learns symbolic rules based on a corpus.
4. **ensemble methods:** combine the results of multiple taggers.

## Transformation-based learning

- **Machine learning technique**
  - For acquiring simple default heuristics and rules for special cases
  - Rules are learned by iteratively collecting errors and generating rules to correct them.
- **Requires a large (training) corpus of manually tagged text**

## TBL: high-level algorithm



Learns an ordered list of transformations (i.e. rewrite rules)

## Rewrite rules

- **Rule**
  - Change *modal* to *noun*, if preceding word is a *determiner*,
- **Example**
  - Determiner: the, a, an, this, that ...
  - Modals: can, will, would, may, might...followed by the main verb
  - The/*det* can/*modal* rusted/*verb* ./.
  - The/*det* can/*noun* rusted/*verb* ./.

## Transformation-based learning

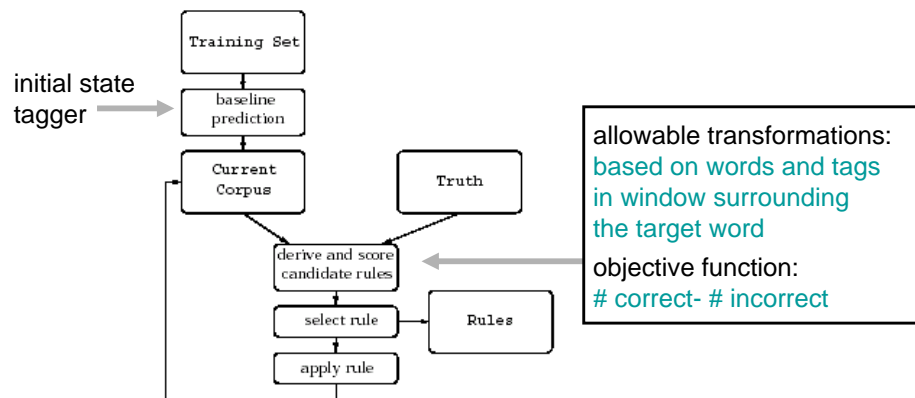


Figure 1: Transformation-based Learning

[Brill 1993]

## Learning algorithm: greedy search

- **Specify**
  - An initial state annotator
  - Space of allowable transformations
  - Objective function for comparing corpus to truth
- **Algorithm**
  - Iterate
    - Try each possible transformation
    - Choose the one with the best score
    - Add to list of transformations
    - Update the training corpus
  - Until no transformation improves performance

## Transformation templates

- **Change tag A to B when:**
  - preceding/following word is tagged Z
  - 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
  - preceding word is tagged Z and following word is tagged W
  - preceding/following word is tagged Z and word two before/after is tagged W

## Generating transformations

- Apply the initial tagger and compile types of tagging errors. Each type of error is of the form:
  - <incorrect tag, desired tag, # of occurrences>
- For each error type, instantiate all templates to generate candidate transformations.
- Apply each candidate transformation to the corpus and count the number of corrections and errors that it produces. Save the transformation that yields the greatest improvement.
- Stop when no transformation can reduce the error rate by a predetermined threshold.

## Example

- Suppose that the initial tagger mistags 159 words as verbs when they should have been nouns.
- Produces the error triple:  
< verb, noun, 159 >
- Suppose template #3 is instantiated as the rule:  
*Change the tag from verb to noun if one of the two preceding words is tagged as a determiner.*
- When this template is applied to the corpus, it corrects 98 of the 159 errors. But it also creates 18 new errors. Error reduction is  $98 - 18 = 80$ .

## Learned rules

1. **NN→VB** if the previous tag is **TO**  
I wanted to/TO win/NN→VB a Subaru WRX...
2. **VBP→VB** if one of the prev-3 tags is **MD**  
The food might/MD vanish/VBP→VB from sight.
3. **NN→VB** if one of prev-2 tags is **MD**  
I might/MD not reply/NN→VB
4. **VB→NN** if one of the prev-2 tags is **DT**
5. **VBD→VBN** if one of the prev-3 tags is **VBZ**
6. **VBN→VBD** if one of the previous tag is **PRP**

## Tagging new text

- **The resulting tagger consists of two phases:**
  - Use the initial tagger to tag all the text
  - Apply each transformation, in order, to the corpus to correct some of the errors.
- **The order of the transformations is very important!**
  - For example, it is possible for a word's tag to change several times as different transformations are applied. In fact, a word's tag could thrash back and forth between the same two tags.

## Evaluation

- **Training: 600,000 words from the Penn Treebank WSJ corpus**
- **Testing: separate 150,000 words from PTB**
- **Assumes all possible tags for all test set words are known.**
- **97.0% accuracy**
- **Tagger learned 378 rules.**

## Problems?

- **Not lexicalized**
  - Transformations are entirely tag-based; no specific words were used in the rules.
  - But certain phrases and lexicalized expressions can yield idiosyncratic tag sequences, so allowing the rules to look for specific words should help...
  - Add additional templates
    - E.g. when the preceding/following word is *w...*
  - Tagger achieves 97.2% accuracy
    - First 200 rules achieved 97.0%
    - First 100 rules achieved 96.8%
  - Learns 447 rules
- **Unknown words**

## Transformation-based learning

- **Part-of-speech tagging**  
[Brill 1995; Ramshaw & Marcus 1994]
- **Prepositional phrase attachment**  
[Brill & Resnik 1995]
- **Syntactic parsing**  
[Brill 1994]
- **Noun phrase chunking**  
[Ramshaw & Marcus 1995, 1999]
- **Context-sensitive spelling correction**  
[Mangu & Brill 1997]
- **Dialogue act tagging**  
[Samuel et al. 1998]