Part-of-Speech Tagging

Updated 5/09

Part-of-Speech Tagging

- Tagging is the task of labeling (or tagging) each word in a sentence with its appropriate part of speech.
- The[AT] representative[NN] put[VBD] chairs[NNS] on[IN] the[AT] table[NN].

Or

- The[AT] representative[JJ] put[NN] chairs[VBZ] on[IN] the[AT] table[NN].

- Tagging is a case of limited syntactic disambiguation. Many words have more than one syntactic category.
- Tagging has limited scope: we just fix the syntactic categories of words and do not do a complete parse.

Performance of POS taggers

- Most successful algorithms disambiguate about 96%-97% of the tokens!
- Information of taggers is quite useful for information extraction, question answering and shallow parsing.

Tag sets

- Brown Corpus: 87 Tags
- Penn Treebank: 45 tags
- C5 tagset (used by the Lancaster project) : 61 tags
- C7 tagset: 146 tags

Some Brown Corpus POS tags

AT	article
BEZ	the word *is*
IN	preposition
JJ	adjective
JJR	comparative adjective
MD	modal (*may, can, …*)
MN	singular or mass noun
NNP	singular proper noun
NNS	plural noun
PERIOD	. : ? !
PN	personal pronoun
RB	adverb
RBR	comparative adverb
TO	the word *to*
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle
VBN	verb, past participle
VBP	verb, non 3d person singular present
VBZ	verb, 3d person singular present
WDT	wh-determiner (*what, which …*)

Information Sources in Tagging

- **Syntagmatic Information**: Look at tags of other words in the context of the word we are interested in.
- Some part of speech sequences are common, such as AT JJ NN (The blue pen) while others are extremely unlikely, e.g. AT JJ VBP (the blue sing ?)

Information Sources in Tagging

- **Lexical Information**: Predicting a tag based on the word concerned.
- Although many words have several tags, their actual usage is usually strongly skewed towards one of the tags.
- For example, *flour* is much more likely to be a noun.

Baselines

- Using syntagmatic information is not very successful. For example, using rule based tagger Greene and Rubin (1971) correctly tagged only 77% of the words.
  - Many content words in English can have various POS. for example, there is a productive process that allows almost every noun to be turned into a verb: *I flour the pan* .
- A dumb tagger that simply assigns each word with its most common tag performs at a 90% accuracy! (Charniak 1993).

Markov Model Taggers

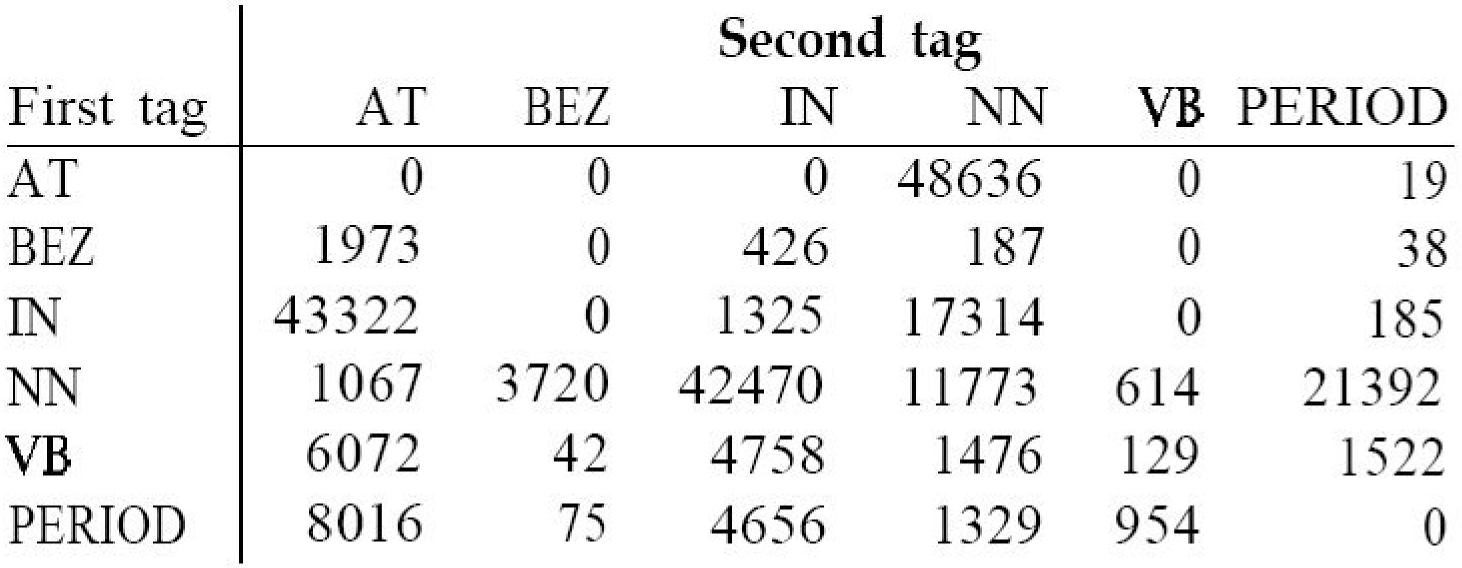
- We look at the sequence of tags in a text as a Markov chain.
  - Limited horizon.
    $P(X_{i+1}= t^j | X_1,\ldots,X_i) = P(X_{i+1}= t^j | X_i)$
  - Time invariant (stationary).
    $P(X_{i+1}= t^j | X_i) = P(X_2= t^j | X_1)$

Notations

w_i	the word at position i in the corpus
t_i	the tag of w_i
$w_{i,i+m}$	the words occurring at positions i through $i+m$ (alternative notations: $w_i \cdots w_{i+m}$, $w_i, \ldots, w_{i+m}$, $w_{i(i+m)}$)
$t_{i,i+m}$	the tags $t_i \cdots t_{i+m}$ for $w_i \cdots w_{i+m}$
w^l	the l^{th} word in the lexicon
t^j	the j^{th} tag in the tag set
$C(w^l)$	the number of occurrences of w^l in the training set
$C(t^j)$	the number of occurrences of t^j in the training set
$C(t^j, t^k)$	the number of occurrences of t^j followed by t^k
$C(w^l : t^j)$	the number of occurrences of w^l that are tagged as t^j
T	number of tags in tag set
W	number of words in the lexicon
n	sentence length

The Visible Markov Model Tagging Algorithm

- The MLE estimate of the probability of tag t^k following tag t^j is obtained from training corpora: $a_{kj}=P(t^k|t^j) = C(t^j, t^k)/C(t^j)$.

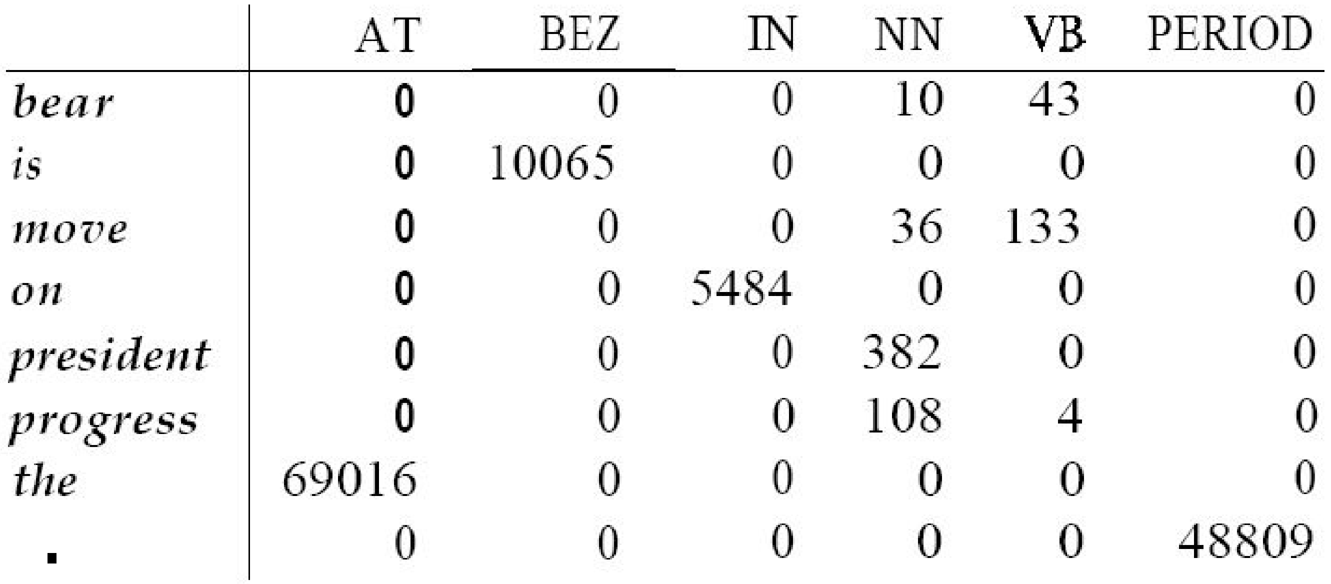


	Second tag					
First tag	AT	BEZ	IN	NN	VB	PERIOD
AT	0	0	0	48636	0	19
BEZ	1973	0	426	187	0	38
IN	43322	0	1325	17314	0	185
NN	1067	3720	42470	11773	614	21392
VB	6072	42	4758	1476	129	1522
PERIOD	8016	75	4656	1329	954	0

Tag counts from brown corpus

The Visible Markov Model Tagging Algorithm

- The MLE estimate of the probability of tag t^k following tag t^j is obtained from training corpora: $a_{kj}=P(t^k|t^j) = C(t^j, t^k)/C(t^j)$.
- The probability of a word being emitted by a tag: $b_{kjl} = P(w^l|t^j) = C(w^l, t^j)/C(t^j)$.



	AT	BEZ	IN	NN	VB	PERIOD
bear	**0**	0	0	10	43	0
is	**0**	10065	0	0	0	0
move	**0**	0	0	36	133	0
on	**0**	0	5484	0	0	0
president	**0**	0	0	382	0	0
progress	**0**	0	0	108	4	0
the	69016	0	0	0	0	0
.	0	0	0	0	0	48809

Best tagging sequence

- The best tagging sequence $t_{1,n}$ for a sentence $w_{1,n}$:

$$\arg\max_{t_{1,n}} P(t_{1,n} \mid w_{1,n}) = \arg\max_{t_{1,n}} \frac{P(w_{1,n} \mid t_{1,n})P(t_{1,n})}{P(w_{1,n})}$$

$$= \arg\max_{t_{1,n}} P(w_{1,n} \mid t_{1,n})P(t_{1,n})$$

$$= \arg\max \prod_{i=1}^{n} P(w_i \mid t_i)P(t_i \mid t_{i-1})$$

The Viterbi Algorithm

```
1.  comment: Given a sentence of length n
2.  $\delta_1(\text{PERIOD})=1.0$
3.  $\delta_1(t)=0.0$ for $t \neq$ PERIOD
4.  for i:=1 to n step 1 do
5.      for all tags $t^j$ do
6.          $\delta_{i+1}(t^j):=\max_{1\leq k\leq T}[\delta_i(t^k)\, P(w_{i+1}|t^j)\, P(t^j|\, t^k)]$
7.          $\Psi_{i+1}(t^j):=\text{argmax}_{1\leq k\leq T}[\delta_i(t^k)\, P(w_{i+1}|t^j)\, P(t^j|\, t^k)]$
8.      end
9.  end
10. $X_{n+1} = \text{argmax}_{1\leq k\leq T}[\delta_{n+1}(t^k)\,]$
11. for j:= n to 1 step -1 do
12.     $X_j := \Psi_{j+1}(X_{j+1})$
13. end
14. $P(X_1, \ldots X_n) = \max_{1\leq j\leq T}[\delta_{n+1}(t^j)]$
```

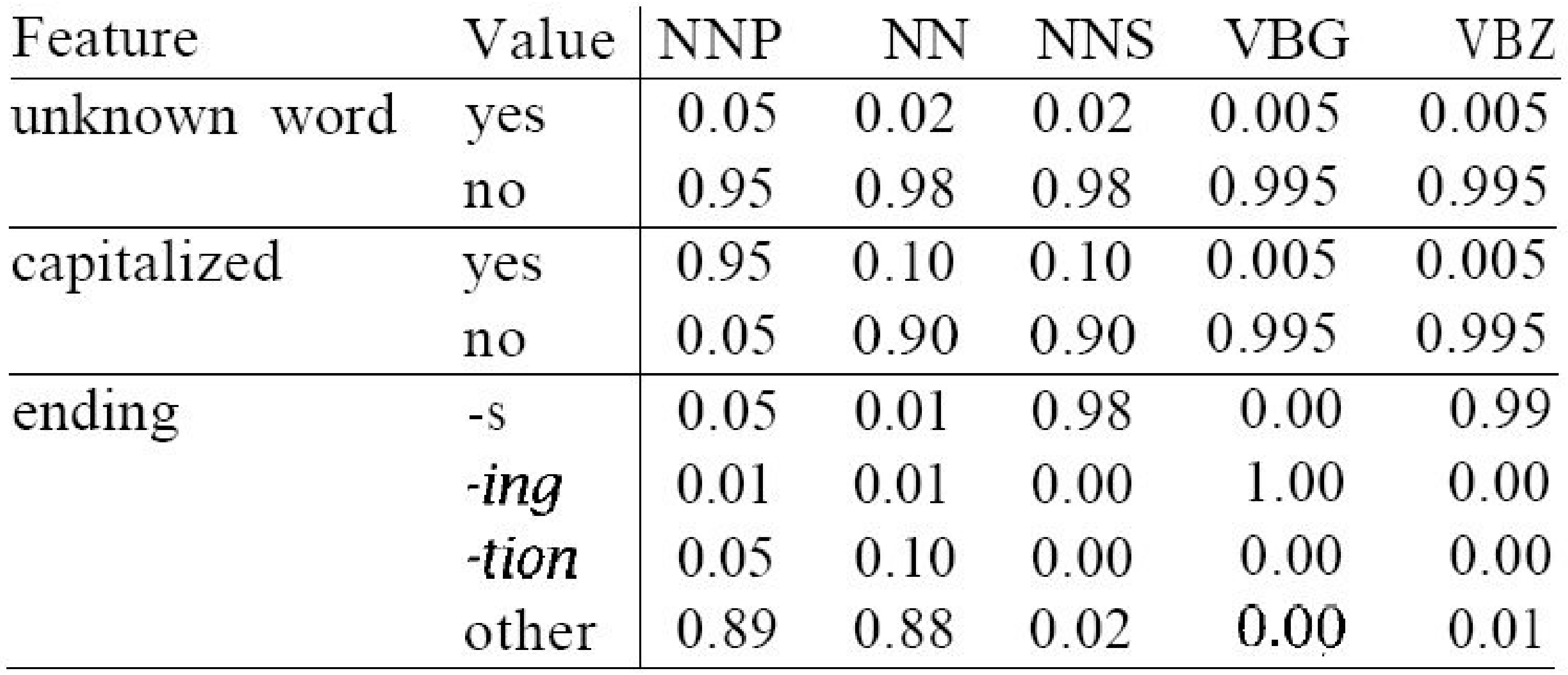
Terminological note

- For the purpose of tagging the Markov Models are treated as Hidden Markov Models.
- In other words, a mixed formalism is used: on training Visible Markov Models are constructed, but they are used as Hidden Markov Models on testing.

Unknown Words

- Simplest method: assume an unknown word could belong to any tag; unknown words are assigned the distribution over POS over the whole lexicon.
- Some tags are more common than others (for example a new word can be most likely one of the open classes such as verbs, nouns etc. but not prepositions or articles).
- Use features of the word (morphological and other cues, for example words ending in *–ed* are likely to be past tense forms or past participles).
- Use context.

Using features of unknown words



Feature	Value	NNP	NN	NNS	VBG	VBZ
unknown word	yes	0.05	0.02	0.02	0.005	0.005
	no	0.95	0.98	0.98	0.995	0.995
capitalized	yes	0.95	0.10	0.10	0.005	0.005
	no	0.05	0.90	0.90	0.995	0.995
ending	-s	0.05	0.01	0.98	0.00	0.99
	-ing	0.01	0.01	0.00	1.00	0.00
	-tion	0.05	0.10	0.00	0.00	0.00
	other	0.89	0.88	0.02	0.00	0.01

- Table of probabilities for dealing with unknown words in tagging. For example, P(unknown word = yes| NNP) = 0.05

Using features of unknown words

- Using Bayes Rule convert the above probabilities.
- Assuming feature independence (Naïve Bayes)

$$P(w^l|t^j) = \frac{1}{Z} P(\text{unknown word}|t^j) P(\text{capitalized}|t^j) P(\text{endings}|t^j)$$

Trigram taggers

- Better results are expected when more context is taken into account.
- A trigram tagger has a two tag memory, which may help tag disambiguation
  - E.g. *is clearly marked* and *he clearly marked* suggest VBN and VBD respectively, since **BEZ RB VBN** is more likely than **BEZ RB VBD** and **PN RB VBD** is more likely than **PN RB VBN**
- A bigram tagger will have hard time in tagging the word ‘marked’
- Trigram taggers need to be smoothed, interpolated etc. to cope with the problem of data sparsity.

Hidden Markov Model Taggers

- Often a large tagged training set is not available.
- We can use an HMM to learn the regularities of tag sequences in this case.
- The states of the HMM correspond to the tags and the output alphabet consists of words in dictionary or classes of words.
- Dictionary information is typically used for constraining model parameters.
- For POS tagging, emission probability for transition i -> j depend solely on i namely $b_{ijk} = b_{ik}$.

Initialization of HMM Tagger

- (Jelinek, 1985) Output alphabet consists of words. Emission probabilities are given by:

$$b_{j.l} = \frac{b^*_{j.l}\, C(w^l)}{\Sigma_{w^m} b^*_{j.m}\, C(w^m)}$$

the sum is over all words w^m in the dictionary

$$b^*_{j.l} = \begin{cases} 0 & \text{if } t^j \text{ is not a part of speech allowed for } w^l \\ \dfrac{1}{T(w^l)} & \text{Otherwise, where } T(w^l) \text{ is the number of tags allowed for } w^l \end{cases}$$

Initialization (Cont.)

- (Kupiec, 1992) Output alphabet consists of word equivalence classes, i.e., metawords u_L, where L is a subset of the integers from 1 to T, where T is the number of different tags in the tag set)

$$b_{j.L} = \frac{b_j{}^*{}_{.l}\, C(u_L)}{\Sigma_{u_{L'}}\, b_j{}^*{}_{.L'}\, C(u_{L'})}$$

the sum in the denom. is over all metawords $u_{L'}$

$$b_j{}^*{}_{.L} = \begin{cases} 0 & \text{if j is not in L} \\ \frac{1}{|L|} & \text{Otherwise, where |L| is the number of indices in L.} \end{cases}$$

Training the HMM

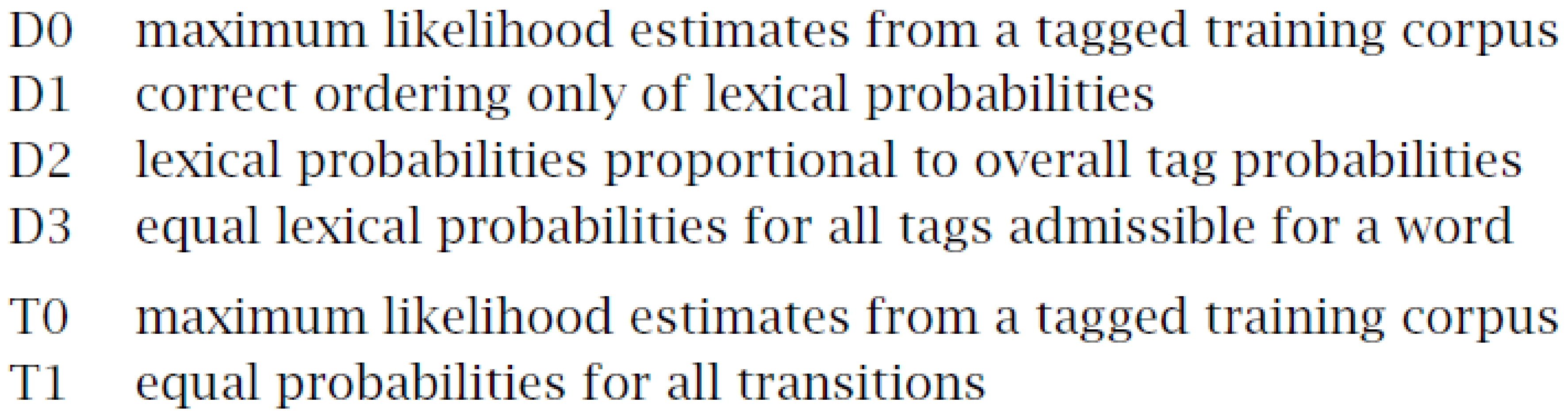
- Once the initialization is completed, the HMM is trained using the Forward-Backward algorithm.

Tagging using the HMM

- Use the Viterbi algorithm, just like in the VMM.

The effect of initialization and training on HMM tagger

- The theoretical stopping criterion for HMM training is not optimal for tagging and results in overtraining.
- Elworthy’s experiment:



D0	maximum likelihood estimates from a tagged training corpus
D1	correct ordering only of lexical probabilities
D2	lexical probabilities proportional to overall tag probabilities
D3	equal lexical probabilities for all tags admissible for a word
T0	maximum likelihood estimates from a tagged training corpus
T1	equal probabilities for all transitions

Transformation-Based Tagging

- Exploits wider range of lexical and syntactic regularities.
- Condition the tags on preceding words not just preceding tags.
- Use more context than bigram or trigram.
- Very famous Transformation-Based tagger is the Brill tagger (1995).

Key components

- Transformation-Based tagging has two key components:
  - A specification of which ‘error correcting’ transformations are admissible.
  - The learning algorithm
- Input data: dictionary and tagged corpus.
- Basic idea: tag each word by its most frequent tag using the dictionary. Then use the ranked list of transformations, for correcting the initial tagging. The ranked list is produced by the learning algorithm.

Transformations

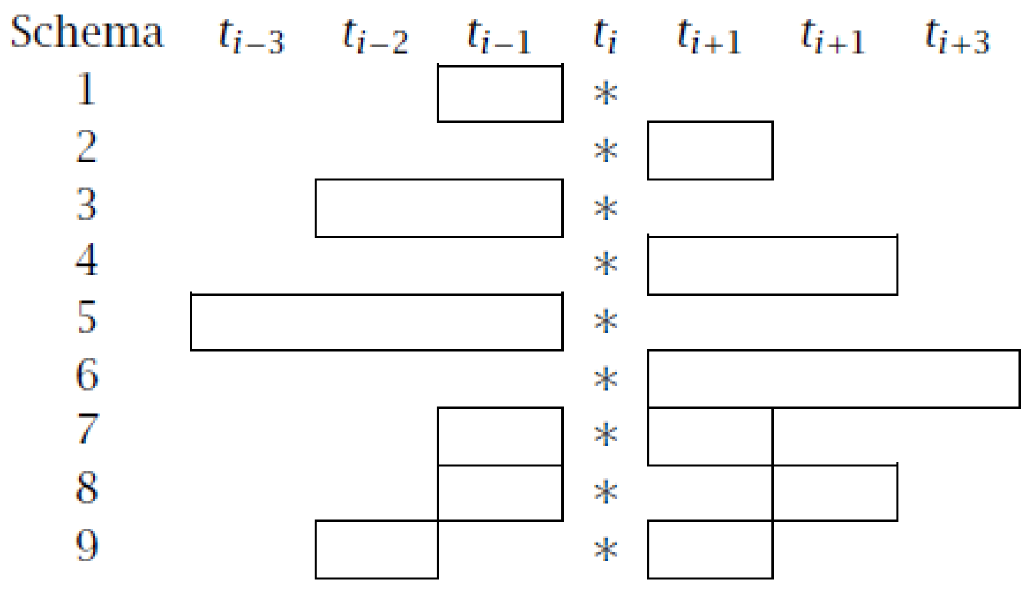
- A transformation consists of two parts: a triggering environment and a rewrite rule.
- Transformations may be triggered by either a tag or by a word.

Examples of some transformations learned in transformation-based tagging

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

Transformations

- In Brill tagger triggering environment consists of a small region around w



Morphology triggered transformations

- A second transformation is triggered by the appearance of specific words in the triggering environment
- A third type of transformation is Morphology triggered transformations are Morphology triggered transformations.
- They offer an elegant way of incorporating morphological knowledge into the general tagging formalism.
- Very helpful for unknown words.
    - Initially unknown words are tagged as NNP (proper noun) if capitalized, and NN (common noun) otherwise.
    - “replace NN by NNS if unknown word’s suffix is '-s’ "
    - etc.

Learning Algorithm

- The learning algorithm selects the best transformations and determines their order of application
- Initially tag each word with its most frequent tag.
- Iteratively we choose the transformation that reduces the error rate most.
- We stop when there is no transformation left that reduces the error rate by more than a pre-specified threshold.

Learning Algorithm

```
$C_0$ := corpus with each word tagged with its most frequent tag
**for** $k := 0$ **step** 1 **do**
    $\nu :=$ the transformation $u_i$ that minimizes $E(u_i(C_k))$
    **if** $(E(C_k) - E(\nu(C_k))) < \epsilon$ **then break fi**
    $C_{k+1} := \nu(C_k)$
    $\tau_{k+1} := \nu$
**end**
Output sequence: $\tau_1, \ldots, \tau_k$
```

- C_i refers to the tagging of the corpus in iteration i, E is the tagging error rate

Automata

- Once trained it is possible to convert the transformation-based tagger into an equivalent finite state transducer, a finite state automaton that has a pair of symbols on each arc, one input symbol and one output symbol. (Roche and Shabes, 1995)
- A finite state transducer passes over a chain of input symbols and converts it to a chain of output symbols, by consuming the input symbols on the arcs it traverses and outputting the output symbols.
- The great advantage of the deterministic finite state transducer is speed. (hundred thousands of words per second…)

Comparison to probabilistic models

- Transformation-Based Tagging does not have the wealth of standard methods available for probabilistic methods.
  - it cannot assign a probability to its prediction
  - It cannot reliably output 'k-best' taggings, namely a set of k most probable hypothesis.
- However, they encode prior knowledge. The transformations are biased towards good generalization.

Tagging Accuracy

- Ranges from 96%-97%
- Depends on:
  - Amount of training data available.
  - The tag set.
  - Difference between training corpus and dictionary and the corpus of application.
  - Unknown words in the corpus of application.
- A change in any of these factors can have a dramatic effect on tagging accuracy – often much more stronger than the choice of tagging method.

Applications of Tagging

- **Partial parsing**: syntactic analysis
- **Information Extraction**: tagging and partial parsing help identify useful terms and relationships between them.
- **Question Answering**: analyzing a query to understand what type of entity the user is looking for and how it is related to other noun phrases mentioned in the question.

Examples for frequent tagging errors

Correct tag	Tag assigned	example
NNS	JJ	An *executive* order
JJ	RB	*more* important issues
VBD	VBG	loan *needed* to meet
VBG	VBD	loan *needed* to meet