CS463 – Natural Language Processing

Part of Speech Tagging

- **Parts of Speech**
- **Part-of-Speech Tagging**
- **Approaches to POS Tagging**
  - **Rule-based Tagging**
  - **Transformation-based Learning Tagging**
  - **Stochastic-based Tagging**
    - **Hidden Markov Model (HMM)**
    - **The Viterbi Algorithm**

Parts of Speech

- Thrax's set of eight **parts-of-speech**: noun, verb, pronoun, preposition, adverb, conjunction, participle, and article.
  - These categories are based on morphological and distributional properties of speech (syntactic not semantic).
  - Recent tagsets have even more categories.

- **Parts-of-speech** (also known as **POS**, **word classes**, or **syntactic categories**) are useful because they reveal a lot about a word and its neighbors.
  - Knowing whether a word is a noun or a verb tells us about likely neighboring words
    - nouns are preceded by determiners and adjectives, verbs by nouns
  - Useful in many applications: machine translation, question answering parsing, information extraction, speech recognition, …

Parts of Speech

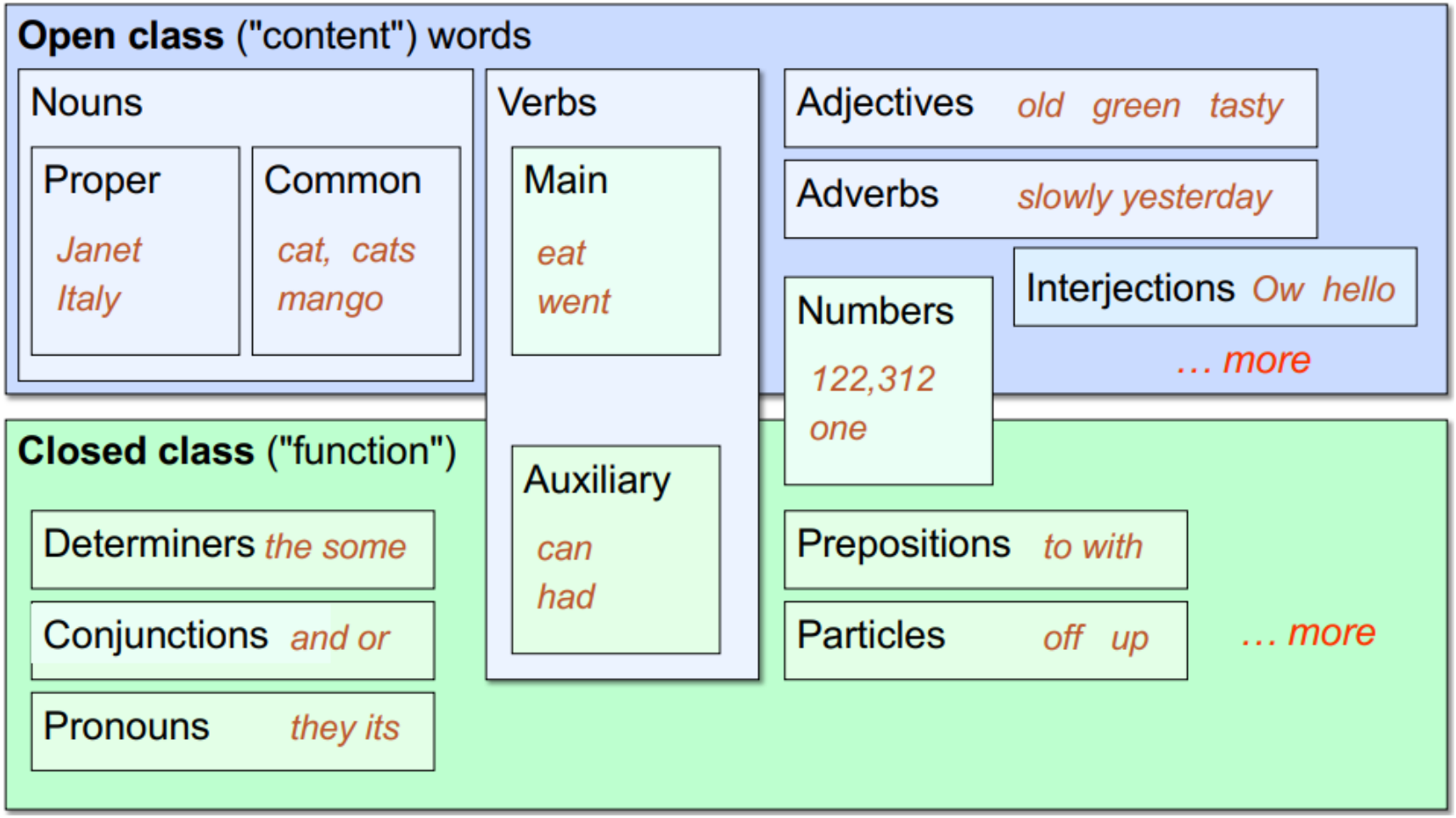
- Parts-of-speech can be divided into two broad categories: **closed class** types and **open class** types.
- Any given speaker or corpus may have different open class words, but all speakers of a language, and sufficiently large corpora, likely share the set of closed class words.
- **Closed class** words – are generally **function** words, which tend to be short, occur frequently, and often have structuring uses in grammar.
  - Prepositions, particles, determiners (articles), conjunctions, pronouns, auxiliary verbs, and numerals
- **Open class** – nouns, verbs, adjectives, and adverbs

Parts of Speech

- **Closed classes**
  - Prepositions: on, under, over, near, by, at, from, to, with, etc.
  - Determiners: a, an, the, etc.
  - Pronouns: she, who, I, others, etc.
  - Conjunctions: and, but, or, as, if, when, etc.
  - Auxiliary verbs: can, may, should, are, etc.
  - Particles: up, down, on, off, in, out, at, by, etc.

- **Open classes**
  - Nouns: student, teacher, boy, girl, dog, cat, etc.
  - Verbs: eat, drink, sleep, walk, run, drink, etc.
  - Adjectives: warm, slim, lovely, cloudy, etc.
  - Adverbs: really, surprisingly, now, later, afterwards, etc.

Parts of Speech



Part-of-Speech Tagging

- **Part-of-speech tagging** is the process of assigning a part-of-speech marker to each word in an input text.
  - The input to a tagging algorithm is a sequence of (tokenized) words and a tagset.
  - The output is a sequence of tags, one per token.
- **Part-of-speech (POS) tagging** is a popular Natural Language Processing process which refers to categorizing words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.
- **Tagging** is the task of labeling (or tagging) each word in a sentence with its appropriate part of speech.
  - Example: The representative put chairs on the table.

    The[AT] representative[NN] put[VBD]

    chairs[NNS] on[IN] the[AT] table[NN].

Part-of-Speech Tagging

- POS Tagging **Process**:
  - Looks at each word in a sentence
  - And assigns tag to each word
    - For example: The man saw the boy.

      the-DET man-NN saw-VBD- the-DET boy-NN

- Tagging has limited scope: we just fix the syntactic categories of words and do not do a complete parse.
  - Associate with each word a lexical tag
    - 45 classes from Penn Treebank
    - 87 classes from Brown Corpus
    - 146 classes from C7 tagset (CLAWS system)

Part-of-Speech Tagging

- **Penn Treebank** - Large corpora of 4.5 million words of American English. It is a 45-tag tagset, which can be used to label many corpora.

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	"	left quote	' or "
LS	list item marker	*1, 2, One*	TO	"to"	*to*	"	right quote	' or "
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Part-of-Speech Tagging

- **Tagging** is a case of limited syntactic **disambiguation**. Many words have more than one syntactic category.
  - words are **ambiguous -** have more than one possible part-of-speech
  - the goal is to find the correct tag for the situation and **resolve** such **ambiguities**.
    - The word "book" can be a verb (book that flight) or a noun (hand me that book).
- How common is tag ambiguity?
  - Tag ambiguity for word types in Brown and WSJ, using Treebank-3

		WSJ	Brown
Types:			
Unambiguous	(1 tag)	44,432 (**86%**)	45,799 (**85%**)
Ambiguous	(2+ tags)	7,025 (**14%**)	8,050 (**15%**)
Tokens:			
Unambiguous	(1 tag)	577,421 (**45%**)	384,349 (**33%**)
Ambiguous	(2+ tags)	711,780 (**55%**)	786,646 (**67%**)

Part-of-Speech Tagging

- Parts of speech follow the usual frequency distribution behavior
  - Most words have one part of speech (1 tag)

The	students	went	to	class
DT	NN	VBP	IN	NN

  - Of the rest, most have two (2 tags)

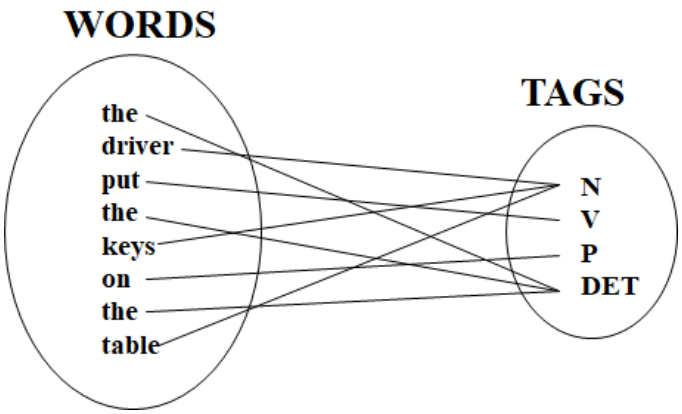
flies	well	others	like
VBZ	ADV	DT	VBP
NN	NN	NN	IN

  - The rest
    - A small number of words have lots of parts of speech
      - Some of the most ambiguous frequent words are that, set, down, put and back;
    - Unfortunately, the words with lots of parts of speech occur with high frequency

earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** about debt
I was twenty-one **back/RB** then

Part-of-Speech Tagging

- There are various **standard tagsets** to choose from; some have a lot more tags than others
- The choice of tagset is based on the application
- Accurate tagging can be done with even large tagsets
- Part of speech **tagging** is the process of assigning parts of speech to each word in a sentence.



- Assume we have
  - A tagset
  - A dictionary that gives you the possible set of tags for each entry
  - A text to be tagged
  - A reason?

Part-of-Speech Tagging

- Most words are unambiguous.
- Many of the most common English words are ambiguous.
- 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”)

Approaches to POS Tagging

- **Rule-based Tagging** approach – uses handcrafted sets of rules to tag input sentences based on lexical and other linguistic knowledge.
- **Stochastic-based Tagging** approach - statistical approaches applied on training corpus to compute probabilities of tags in contexts.
- **Transformation-based Tagging** approach (or Brill tagging) – hybrid of both rule-based and stochastic approaches.

Rule-based Tagging

- A two stage architecture
  - Use dictionary (lexicon) to assign each word a list of potential POS.
  - Use large lists of hand-written disambiguation rules to identify a single POS for each word.
- ENGTWOL tagger (Voutilainen,'95)
  - 56000 English word stems.
- Advantage: high precision (99%).
- Disadvantage: needs a lot of rules.

Rule-based Tagging

- Hand-crafted rules for ambiguous words that test the context to make appropriate choices
  - Relies on rules e.g. NP → Det (Adj*) N
    - For example: *the clever student*
  - Morphological Analysis to aid disambiguation
    - E.g. X–ing preceded by Verb – label it a verb
  - 'Supervised method' I.e. using a pre-tagged corpus
    - Advantage: Corpus of same genre
    - Problem: not always available
  - Extra Rules
    - indicative of nouns
    - Punctuation
  - Extremely labor-intensive

Rule-based Tagging

- Start with a dictionary of words and possible tags.
- Assign all possible tags to words using the dictionary.
- Write rules by hand to selectively remove tags
  - Typically >1k rules
- Stop when each word has exactly one (presumably correct) tag.

Rule-based Tagging - Example

- Start with a dictionary of words and possible tags.

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

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

Rule-based Tagging - Example

- Assign all possible tags to words using the dictionary.

She	Promised	To	Back	The	Bill
PRP	VBN VBD	TO	VB NN JJ RB	DT	NN VB

Rule-based Tagging - Example

- Write rules by hand to selectively remove tags
  - Eliminate VBN if VBD is an option when VBN | VBD follows PRP

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB NN JJ RB	DT	NN VB

Rule-based Tagging - Example

- Write rules by hand to selectively remove tags
  - Eliminate NN/JJ/RB if VB is an option following a TO

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB ~~NN~~ ~~JJ~~ ~~RB~~	DT	NN VB

Rule-based Tagging - Example

- Write rules by hand to selectively remove tags
  - Eliminate VB if NN is an option following a DT

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB ~~NN~~ ~~JJ~~ ~~RB~~	DT	NN ~~VB~~

- Stop when each word has exactly one (presumably correct) tag.

Transformation-based Learning Tagging

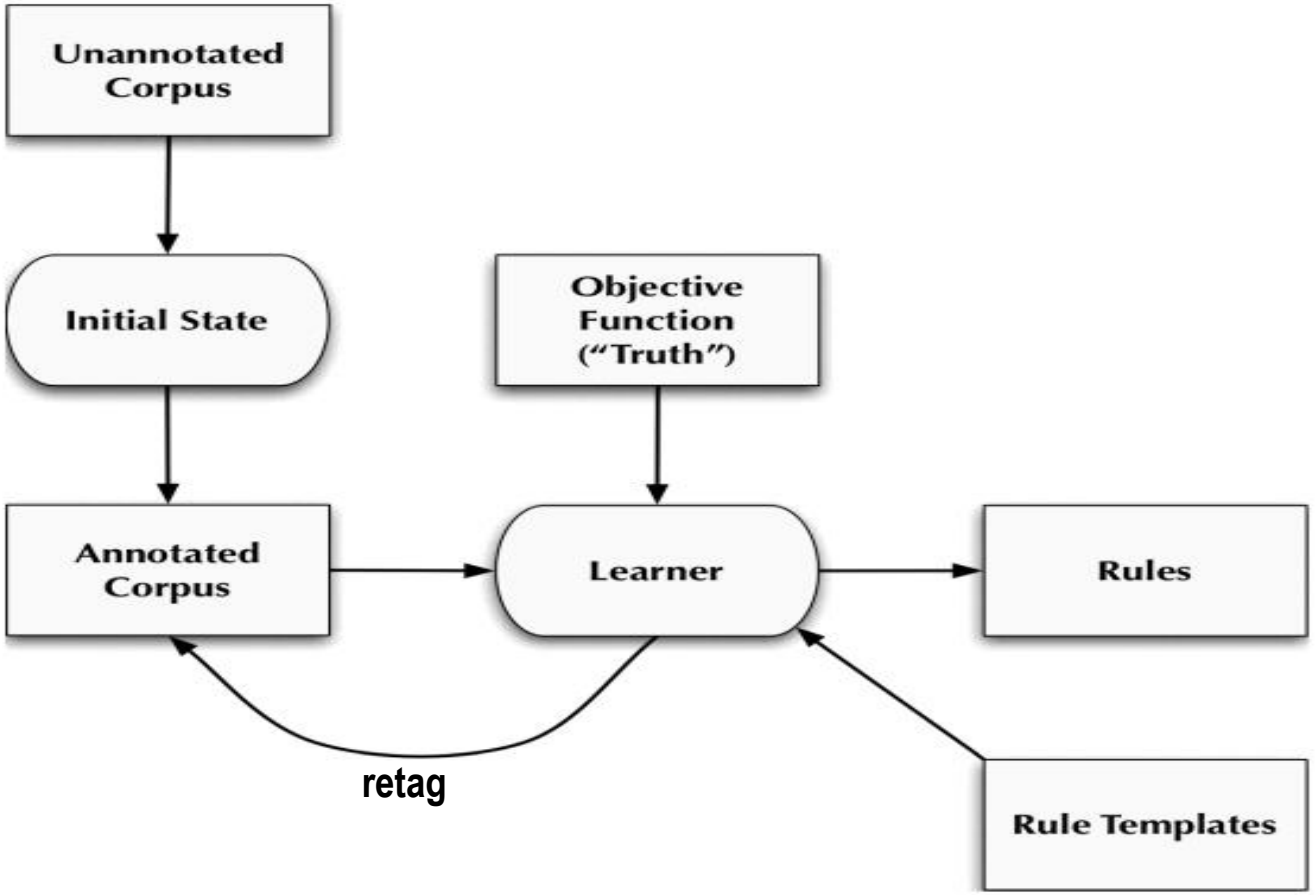
- **Transformation-based Learning (TBL) Tagging –** is an example of rule-based machine learning tagging that combines **Rule-based** and **Stochastic** tagging.
  - It is rule-based because it specifies tags in a certain environment.
  - It is stochastic (probabilistic) as it uses tagged corpus to find the best performing rules.
- Rules are learnt from data input:
  - Tagged corpus.
  - Dictionary (with most frequent tags for each word).
- TBL tagging has been applied to a large number of NLP tasks:
  - POS tagging
  - NP chunking
  - Word sense disambiguation
  - etc.

Transformation-based Learning Tagging

- How does Transformation-based Learning tagging work?
- It consists of two phases:
  - Training phase (typically applied once) – where rules are learnt.
  - Application phase (typically applied many times) – the rules are applied in the order they were learnt.

Transformation-based Learning Tagging

- **Training phase** of Transformation-based Learning:

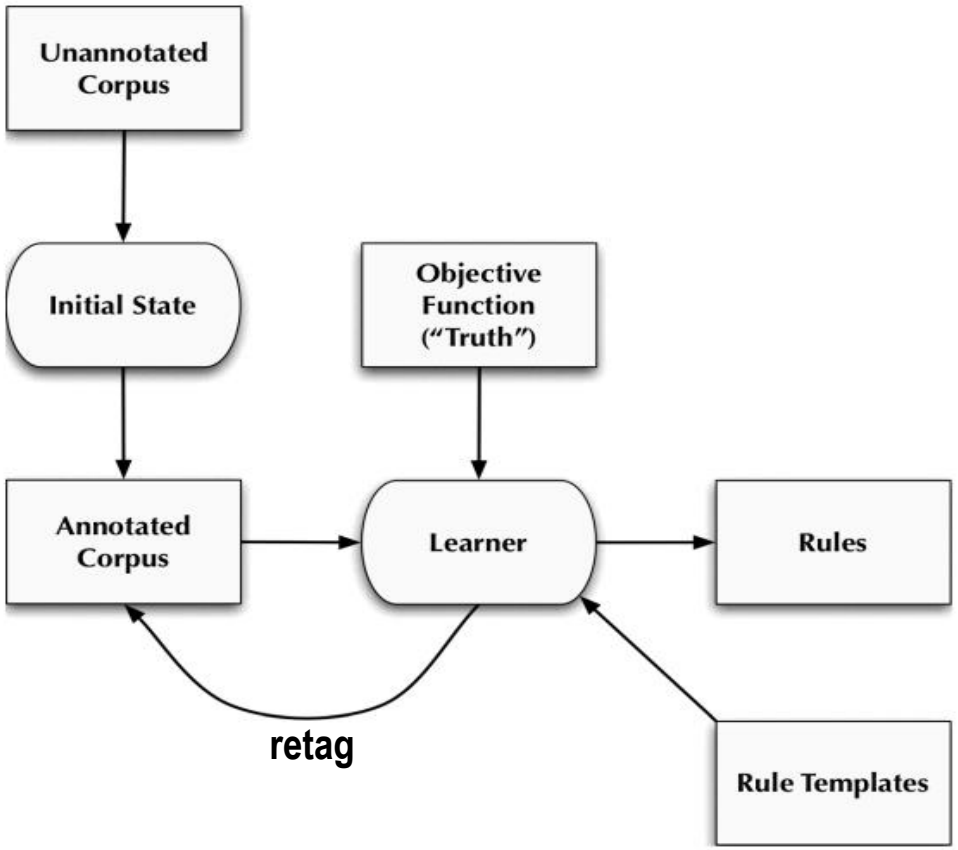


Transformation-based Learning Tagging

- TBL algorithm:
  - ***Step 1:*** Label every word with most likely tag (from a dictionary)
  - ***Step 2:*** Check every possible transformation & select the one that results in the most improved tagging accuracy
  - ***Step 3:*** Re-tag corpus applying this rule, and add rule to end of rule set
  - Repeat 2-3 until some stopping criterion is reached,
    - e.g., X% correct with respect to training corpus
  - OUTPUT → an ordered set of transformation rules.

Transformation-based Learning Tagging

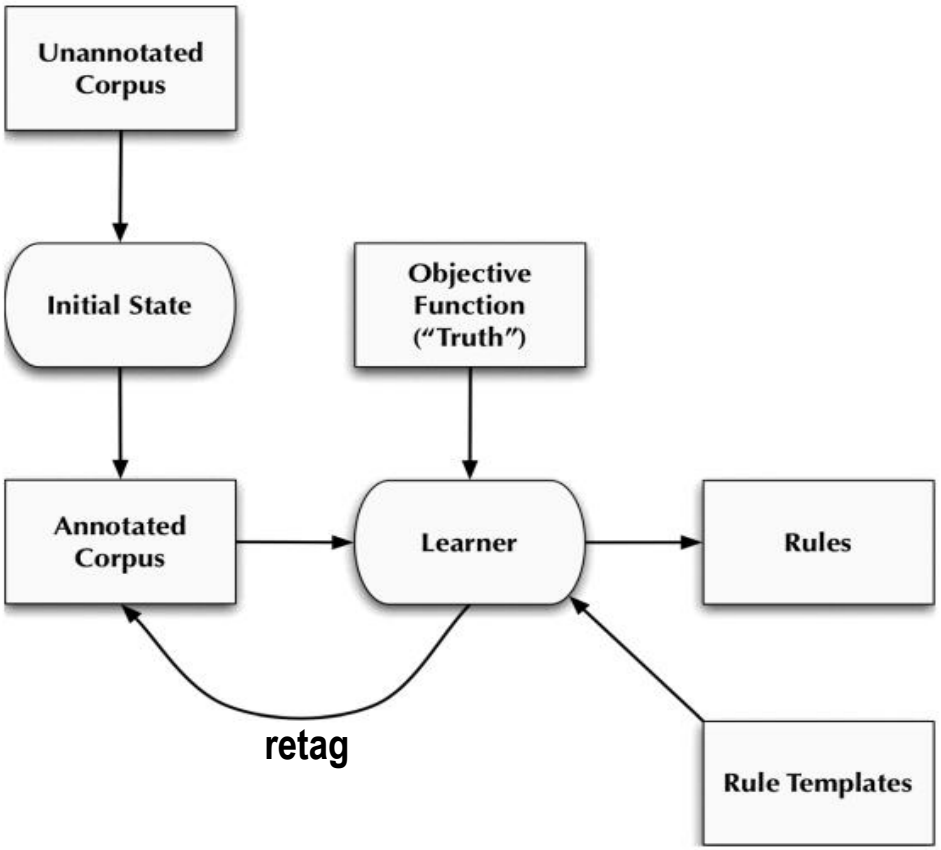
- POS tagging with transformation-based learning
  1. Initial state
     - known words (i.e. words found in the lexicon) - are tagged with their most frequent tag.
     - Unknown words are tagged with the most frequent tag in the training corpus depending on the first letter (capital or not) of the word in question.



Transformation-based Learning Tagging

- POS tagging with transformation-based learning

2. Lexical Tagging
   - The unknown words are tagged in isolation, based on their morphology and their immediate neighbor.
3. Contextual Tagging
   - All words are tagged in context



Transformation-based Learning Tagging

- Popular implementations (examples) of TBL tagging:
  - *Original algorithm by **Brill** (1992)*
    - ***Basic idea -*** do a quick job first (using frequency), then revise it using contextual rules.
    - A supervised method – requires a tagged corpus.
    - Very lengthy training times.
  - *Algorithm* by Ramshaw and Marcus (1994)
    - Faster but extremely memory-consuming
  - Algorithm by Ngai and Florian (2001)
    - Very fast (twice fast the implementation of Brill).
    - Support multidimensional learning (multiple task classification).

Transformation-based Learning Tagging - Example

... is expected to race tomorrow

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

Transformation-based Learning Tagging - Example

- Tagging these two sentences:
  - *It is expected to race tomorrow*
  - *The race for outer space*

- Tagging algorithm:
  1. Tag all uses of "race" as NN (most likely tag in the Brown corpus)
     - It is expected to race/NN tomorrow
     - the race/NN for outer space
  2. Use a contextual rule to replace the tag NN with VB for all uses of "race" preceded by the tag TO:
     - It is expected to race/VB tomorrow
     - the race/NN for outer space

Stochastic-based Tagging

- **Simple approach -** disambiguate words based on the probability that a word occurs with a particular single tag.
- **N-gram approach -** the best tag for given words is determined by the probability that it occurs with the *n* previous tags.
- **Viterbi Algorithm -** trim the search for the most probable tag using the best *N Maximum Likelihood Estimates* (*N* is the number of tags of the following word)**.**
- **Hidden Markov Model (HMM) -** combines the above two approaches.

Stochastic-based Tagging

- We want the best set of tags for a sequence of words (or a sentence).
  - *W* is a sequence of words
  - *T* is a sequence of tags
- Using Bayes theorem:

$$\arg\max P(T \mid W) = \frac{P(W \mid T)P(T)}{P(W)}$$

  - P(w) is common, so it can be ignored

$$\arg\max P(T \mid W) = P(W \mid T)P(T)$$

Stochastic-based Tagging – Hidden Markov Model (HMM)

- The goal of HMM decoding is to choose the **tag sequence** t_1^n (hidden) that is most probable, given the **observation sequence** of n words w_1^n

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i|t_i)}^{\text{emission}} \overbrace{P(t_i|t_{i-1})}^{\text{transition}}$$

  - $P(t_i|t_{i-1})$ Tag **transition** probability
  - $P(w_i|t_i)$ Word **likelihood** (or **emission**)

- Tag transition probability – represents probability of a tag given a previous tag.
  - Example: Determiners are very likely to precede adjectives and nouns, as in sequences like:
    - that/DT flight/NN
    - The/DD black/JJ hat/NN
  - P(NN|DT) and P(JJ|DT) are expected to be high
  - P(DT|JJ) is expected to be low

Stochastic-based Tagging – Hidden Markov Model (HMM)

- How do we get the probability of a specific tag sequence P(T)?
  - Count the number of times a sequence occurs and divide by the number of sequences of that length. Not likely to work.
  - Make a Markov assumption and use N-grams over tags...
    - P(T) is a product of the probability of N-grams that make it up.
  - Bigram Example:
    - <s> Det Adj Adj Noun </s>
    - P(Det|<s>)P(Adj|Det)P(Adj|Adj)P(Noun|Adj)

- Where do you get the N-gram counts?
  - From a large hand-tagged corpus
    - For Bigrams, count all the Tag_i Tag_{i+1} pairs
    - And smooth them to get rid of the zeroes
  - Alternatively, you can learn them from an untagged corpus.

Stochastic-based Tagging – Hidden Markov Model (HMM)

- How do we get the probability of a specific word sequence P(W|T)?
  - It is asking the probability of seeing "The big red dog" given "Det Adj Adj Noun" !
  - Collect up all the times you see that tag sequence and see how often "The big red dog" shows up. Again not likely to work.
- We'll make the following assumption:
  - Each word in the sequence only depends on its corresponding tag. So…

$$P(W \mid T) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

- How do we get the statistics for that?
  - See next slide.

Stochastic-based Tagging – Hidden Markov Model (HMM)

- Start with Bigram-HMM tagger:

$$\text{argmax}_T\ P(T|W)$$
$$\text{argmax}_T P(T)P(W|T)$$
$$\text{argmax}_t P(t_1 \ldots t_n)P(w_1 \ldots w_n|t_1 \ldots t_n)$$
$$\text{argmax}_t [P(t_1)P(t_2|t_1)\ldots P(t_n|t_{n-1})][P(w_1|t_1)P(w_2|t_2)\ldots P(w_n|t_n)]$$

- To tag a single word: $t_i = \text{argmax}_j\ P(t_i|t_{i-1})P(w_i|t_i)$
- How do we compute $P(t_i|t_{i-1})$?

  $c(t_{i-1}t_i)/c(t_{i-1})$
- How do we compute $P(w_i|t_i)$?

  $c(w_i\ t_i)/c(t_i)$

Stochastic-based Tagging – Hidden Markov Model (HMM)

- Computing the transition probability $P(t_i|t_{i-1})$ of VB (verb in the base form) given MD (modal verb) as its previous occurring tag:
  - Based on WSJ corpus, MD occurs 13124 times, of which it is followed by VB 10471 times.

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1},t_i)}{C(t_{i-1})} \Rightarrow P(VB|MD) = \frac{C(MD,VB)}{C(MD)} = \frac{10471}{13124} = .80$$

- Computing the emission probability $P(w_i|t_i)$ of the word "will" given MD as its tag:
  - Of the 13124 occurrences of MD in the WSJ corpus, it is associated with the word "will" 4046 times.

$$P(w_i|t_i) = \frac{C(t_i,w_i)}{C(t_i)} \Rightarrow P(will|MD) = \frac{C(MD,will)}{C(MD)} = \frac{4046}{13124} = .31$$

Stochastic-based Tagging – Hidden Markov Model (HMM)

- Example: suppose w_i = race, a verb (VB) or a noun (NN)?
- Assume that other mechanism has already done the best tagging to the surrounding words, leaving only race untagged.
  - Sami/NNP is/VBZ expected/VBN to/TO race/? tomorrow/NN
  - People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/? for/IN outer/JJ space/NN

$$t_i = \text{argmax}_j \, P(t_i|t_{i-1})P(w_i|t_i)$$

Simplify the problem:
to/TO race/???
the/DT race/???

→

P(VB|TO) P(race|VB)
P(NN|TO) P(race|NN)

Stochastic-based Tagging – Hidden Markov Model (HMM)

- Look at the Brown and Switchboard corpora

  P(NN | TO) = 0.021
  P(VB | TO) = 0.34

- If we are expecting a verb, how likely it would be "race"?

  P(race | NN) = 0.00041
  P(race | VB) = 0.00003

- Finally $t_i = \text{argmax}_j \, P(t_i|t_{i-1})P(w_i|t_i)$

  P(NN | TO) P(race | NN) = 0.021 * 0.00041 = 0.000009
  P(VB | TO) P(race | VB) = 0.34 * 0.00003 = 0.00001

- So "race" is likely to be tagged as VB, as 0.00001 > 0.000009

Stochastic-based Tagging – The Viterbi Algorithm

- How do we compute the most probable tag sequence for a sequence of words (a complete sentence)?
  - We use the **Viterbi algorithm** as a decoding algorithm for **HMMs**
- Viterbi resembles the dynamic programming **Minimum Edit Distance** algorithm.
- The Viterbi algorithm first sets up a **probability matrix** or **lattice**
  - where we have **columns as our observables** (words of a sentence in the same sequence as in sentence);
  - and **rows as hidden states** (all known possible POS Tags).
- Each cell of the lattice is represented by $\mathbf{V_t(j)}$ (*t* represent column and *j* represents the row), called **Viterbi path probability**.

Stochastic-based Tagging – The Viterbi Algorithm

- **Viterbi path probability $V_t(j)$** is computed as:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

- We get *a* (transition matrix) and *b* (emission matrix) from the HMM calculations discussed earlier.

Stochastic-based Tagging – The Viterbi Algorithm - Example

- Consider this sentence: Janet will back the bill
- Our aim is to get something like this:

  Janet/NNP will/MD back/VB the/DT bill/NN

- Before beginning, let's get our required matrices calculated using **WSJ** corpus
  - with the help of **HMM** calculations for **transition** and **emission** probabilities.

Stochastic-based Tagging – The Viterbi Algorithm - Example

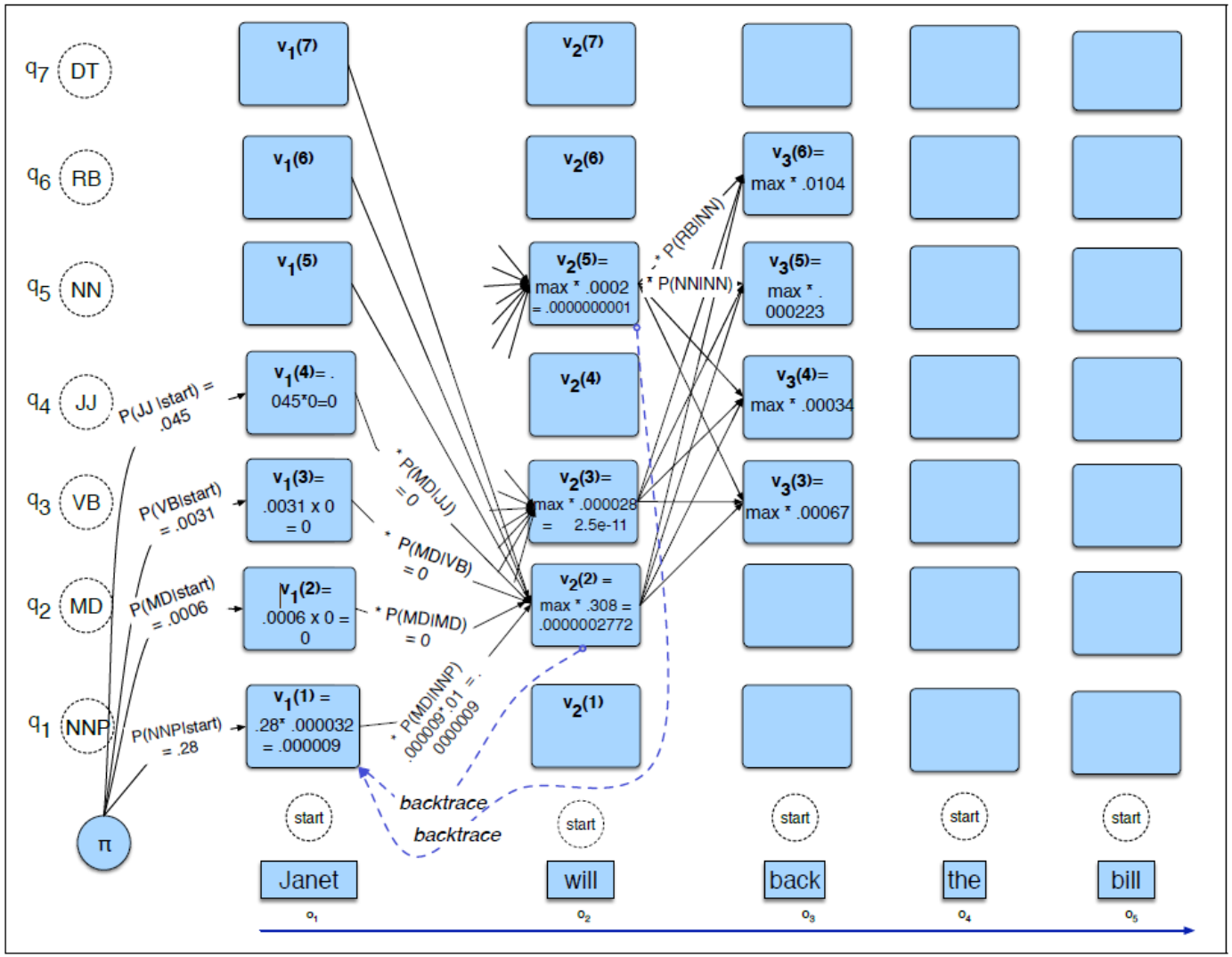
- **A. Transition** probabilities:

	NNP	MD	VB	JJ	NN	RB	DT
<*s*>	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

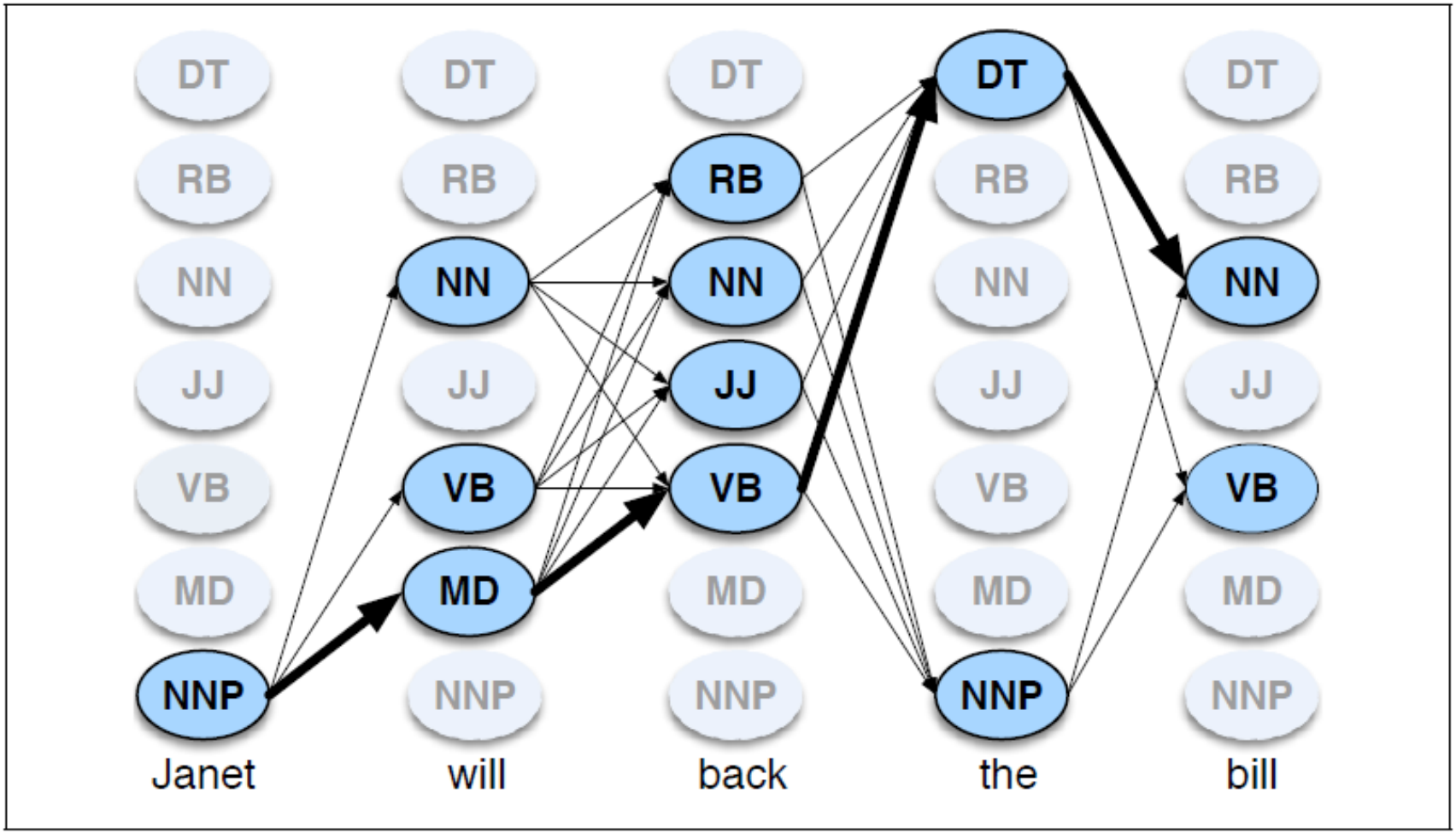
- **B. Emission** probabilities (Observation likelihoods):

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

Stochastic-based Tagging – The Viterbi Algorithm - Example



Stochastic-based Tagging – The Viterbi Algorithm - Example



Stochastic-based Tagging

- This method has achieved 95-96% correct with reasonably complex English tagsets and reasonable amounts of hand-tagged training data.
- POS Taggers accuracy rates are in the range of 95-99%
  - Vary according to text/type/genre
    - Of pre-tagged corpus
    - Of text to be tagged
- Worst case scenario: assume success rate of 95%
    - Prob(one-word sentence) = .95
    - Prob(two-word sentence) = .95 * .95 = 90.25%
    - Prob(ten-word sentence) = 59% approx