POS Tagger types

History of Tagging

100 BC Dionysis Thrax – document eight parts of speech

1959 Harris (U Penn) first tagger as part of TDAP parser project

1963 Klien and Simmons() Computational Grammar Coder CGC

- Small lexicon (1500 exceptional words), morphological analyzer and context disambiguator

1971 Green and Rubin TAGGIT expanded CGC

- More tags 87 and bigger dictionary
- Achieved 77% accuracy when applied to Brown Corpus

1983 Marshal/Garside – CLAWS tagger

- Probabilistic algorithm using tag bigram probabilities

History of Tagging

1988 Church () PARTS tagger

- Extended CLAWS idea
- Stored P(tag | word) * P(tag | previous n tags)
- instead of P(word | tag) * P(tag | previous n tags) used in HMM taggers

1992 Kupiec HMM tagger

1994 Schütze and Singer () variable length Markov Models

1994 Jelinek/Magerman – decision trees for the probabilities

1996 Ratnaparkhi () - the Maximum Entropy Algorithm

1997 Brill – unsupervised version of the TBL algorithm

Why POS Tagging

Why POS Tagging

- Named Entity Recognition (NER)
- Sentiment analysis
- Question answering
- Word sense disambiguation.

POS Tagging

• N	noun	*chair, bandwidth, pacing*
• V	verb	*study, debate, munch*
• ADJ	adjective	*purple, tall, ridiculous*
• ADV	adverb	*unfortunately, slowly*
• P	preposition	*of, by, to*
• PRO	pronoun	*I, me, mine*
• DET	determiner	*the, a, that, those*

POS Tagging

- Words have more than 1 POS
  - The back door = JJ
  - On my back = NN
  - Win the waters back = RB
  - Promised to back the bill = VB
- What is POS tagging problem?
- To determine the POS tag for a particular instance of a word

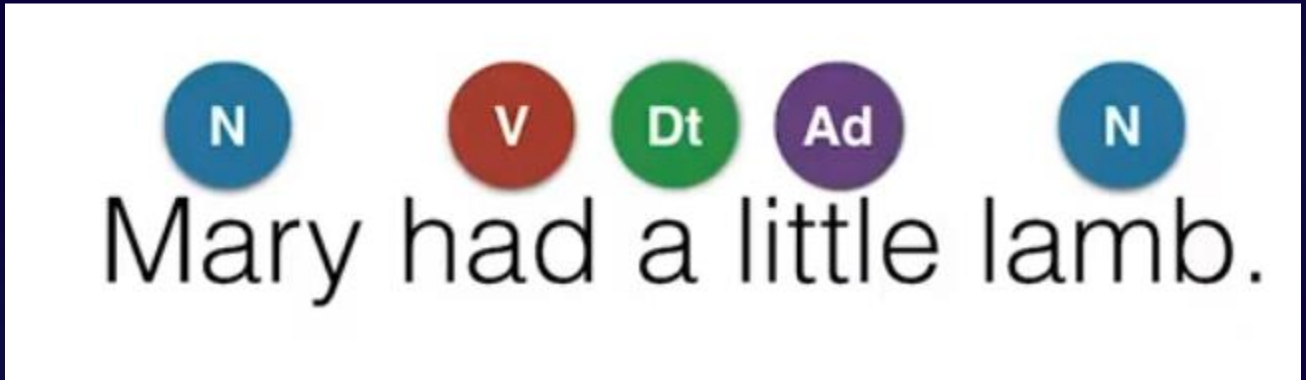
How hard is POS tagging?

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well*, *beat*)	32	
	7 tags	2	(*still*, *down*)	6	(*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags			4	(*'s*, *half*, *back*, *a*)
	9 tags			3	(*that*, *more*, *in*)

Methods of POS Tagging

- Rule Based
  - ENGTWOL
- Stochastic
  - HMM, CRF
- N-gram tagging
- Transformation Based Tagging
  - Brill Tagger

POS Tagging using HMM

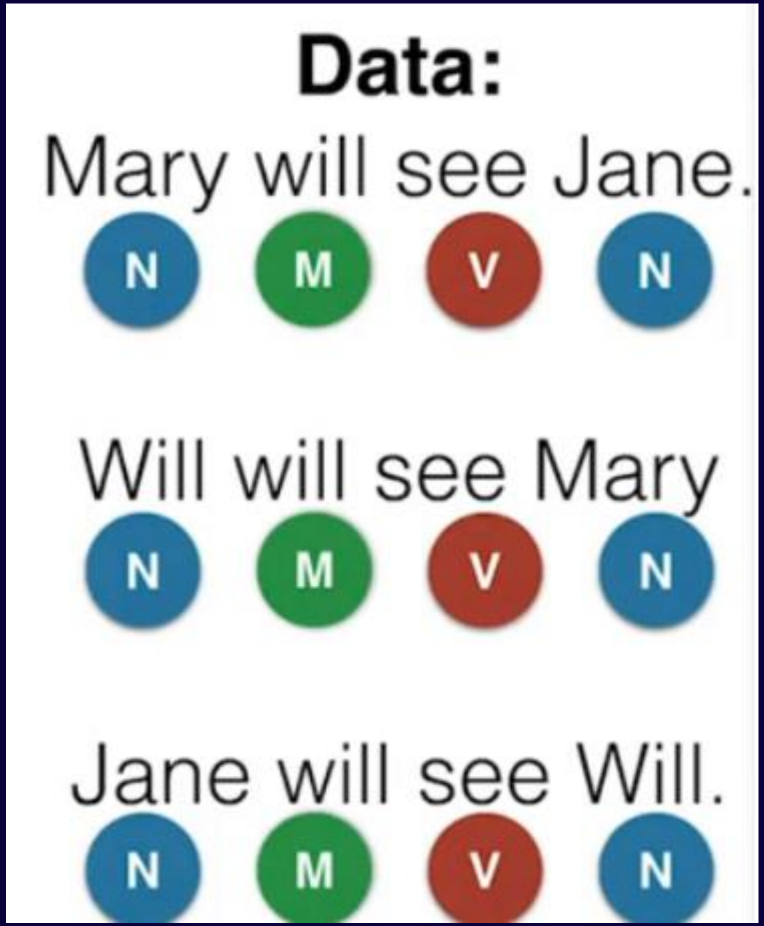


- The collection of tags used for a particular task is known as a tagset.
- The POS tagging in this project is using a universal tagset.
- They are total 12 tags in this tagset

POS Tagging using HMM

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

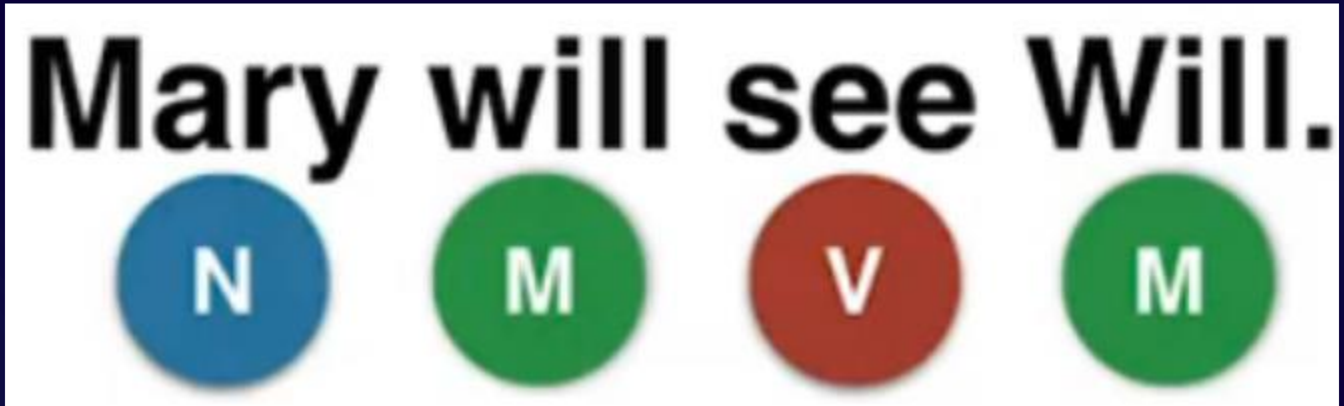
POS Tagging using HMM



POS Tagging using HMM

	N	V	M
Mary	2	0	0
see	0	3	0
Jane	2	0	0
Will	2	0	3

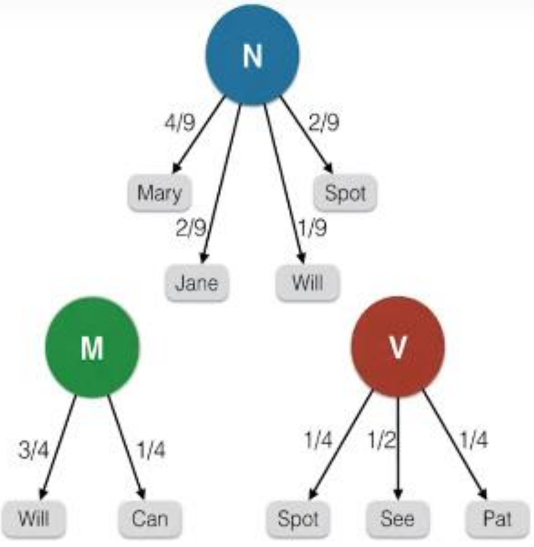
Incorrect POS Tagging using MFC



POS Tagging using HMM

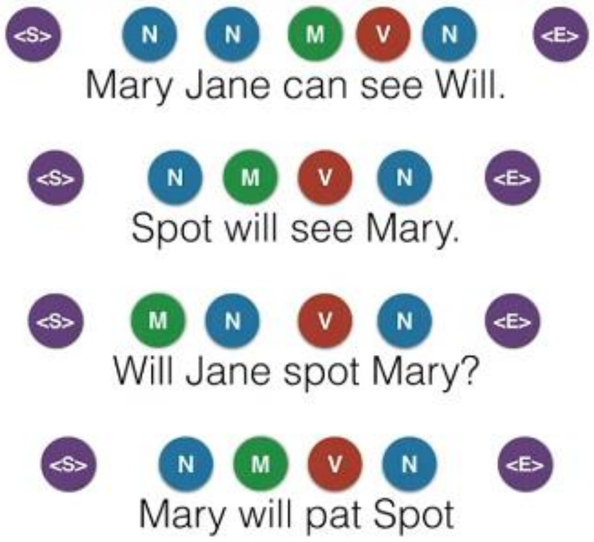
Emission Probabilities

	N	M	V
Mary	4/9	0	0
Jane	2/9	0	0
Will	1/9	3/4	0
Spot	2/9	0	1/4
Can	0	1/4	0
See	0	0	1/2
Pat	0	0	1/4

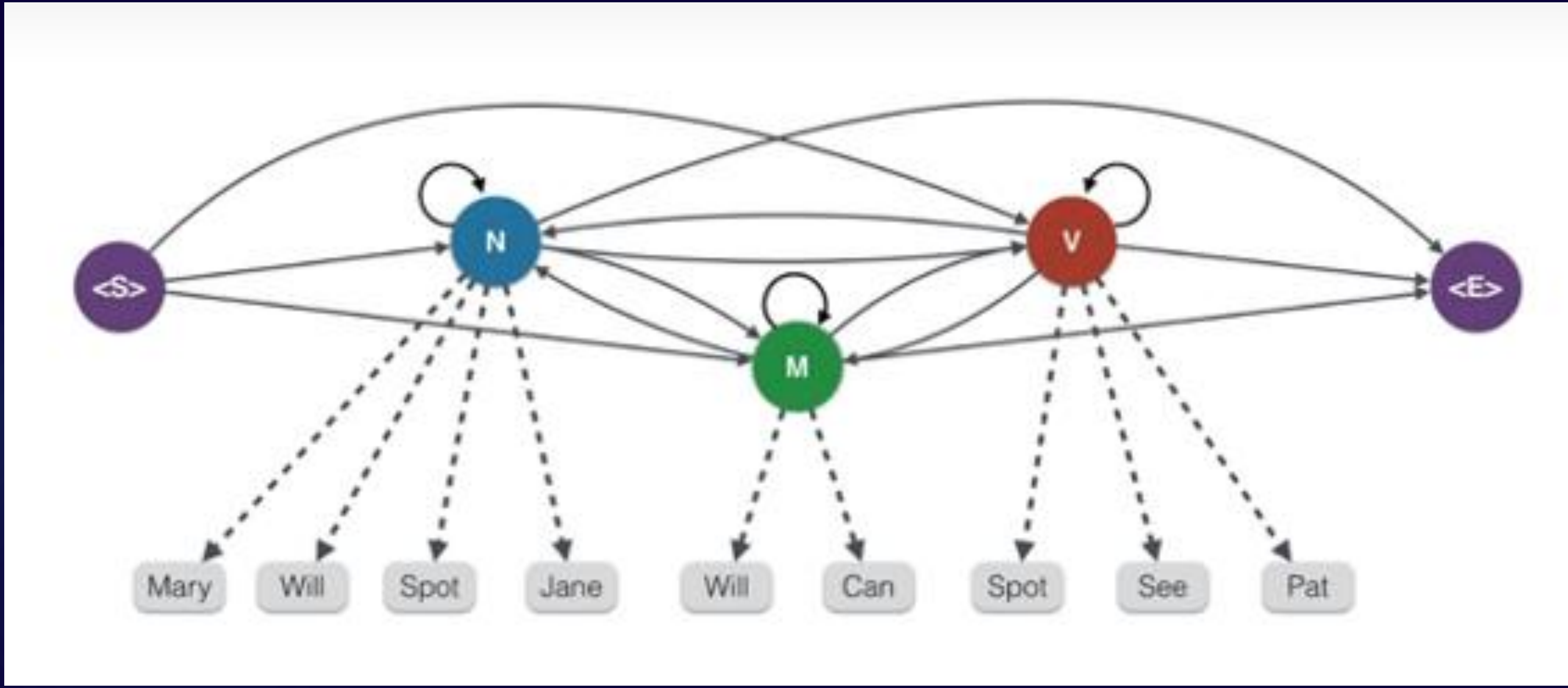


Transition Probabilities

	N	M	V	<E>
<S>	3/4	1/4	0	0
N	1/9	1/3	1/9	4/9
M	1/4	0	3/4	0
V	1	0	0	0



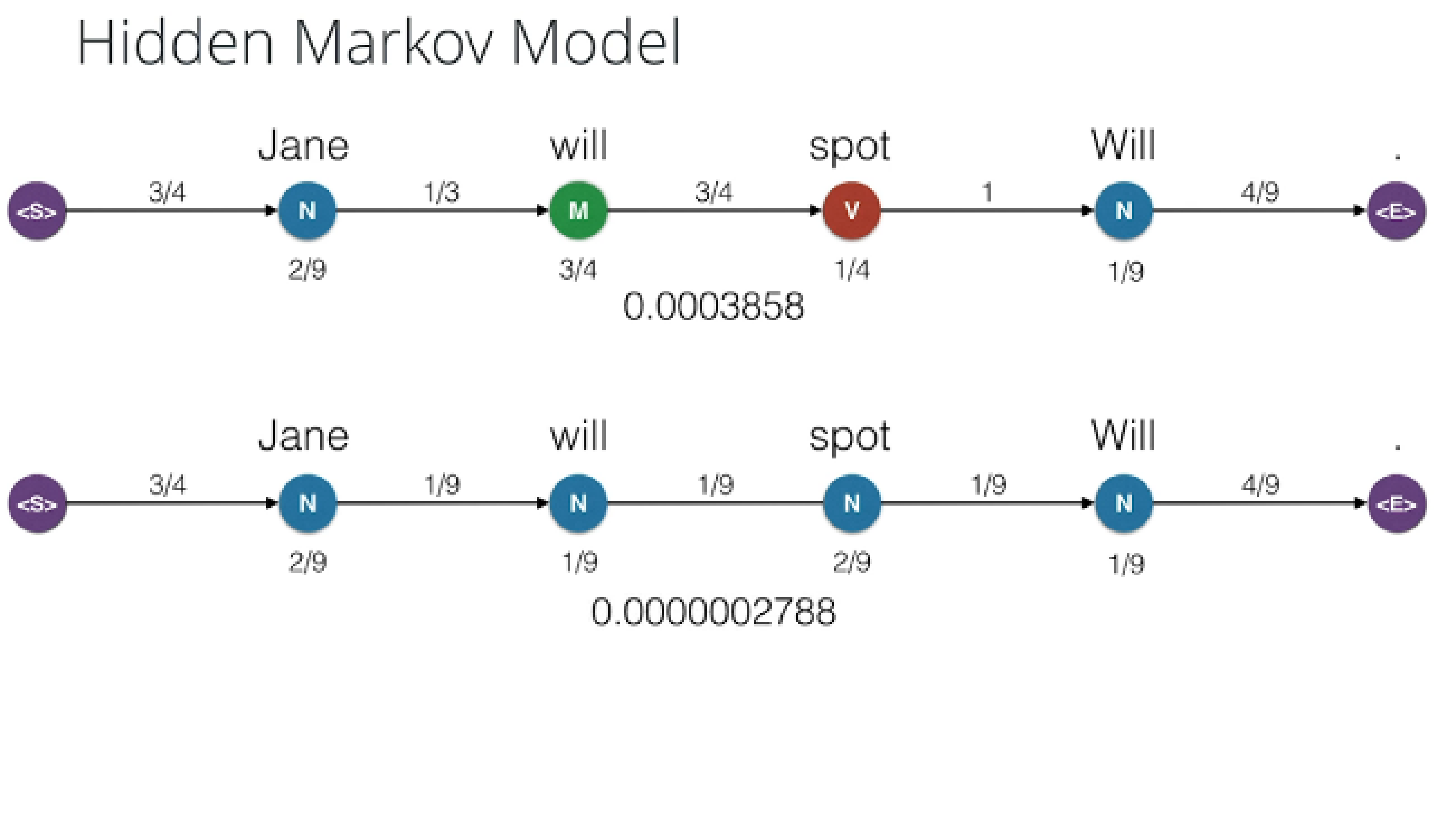
POS Tagging using HMM



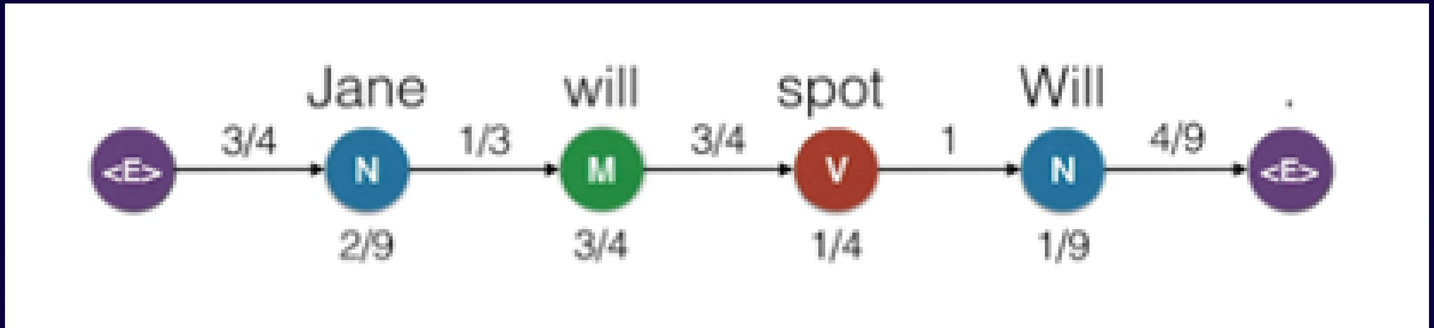
POS Tagging using HMM

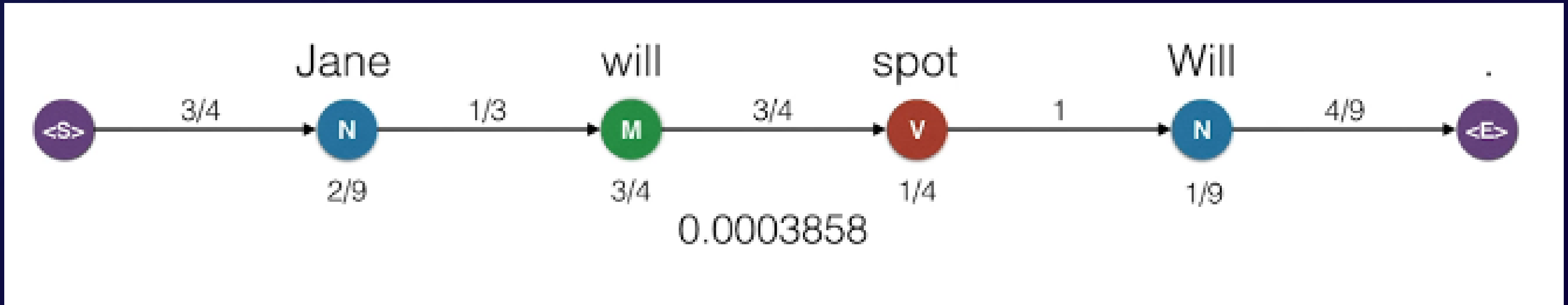
- HMMM tagger has 1 hidden state for each possible tag
- And 2 distributions
- Emission probabilities and Transition probabilities
- Emission probability – probability of observing a given word
- Transition probabilities give the conditional probability of moving between tags
- HMM uses Maximum likelihood estimate
- To find POS for sentence
- Jane will spot Will

POS Tagging using HMM



OUTPUT - POS Tagging using HMM





Rule Based Tagging

- Start with a dictionary
- Assign all possible tags to words from 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 words of English 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 when VBN|VBD follows “<start> PRP”

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

Stage 1 of ENGTWOL Tagging

First Stage: Run words through FST morphological analyzer 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) ;following which is 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

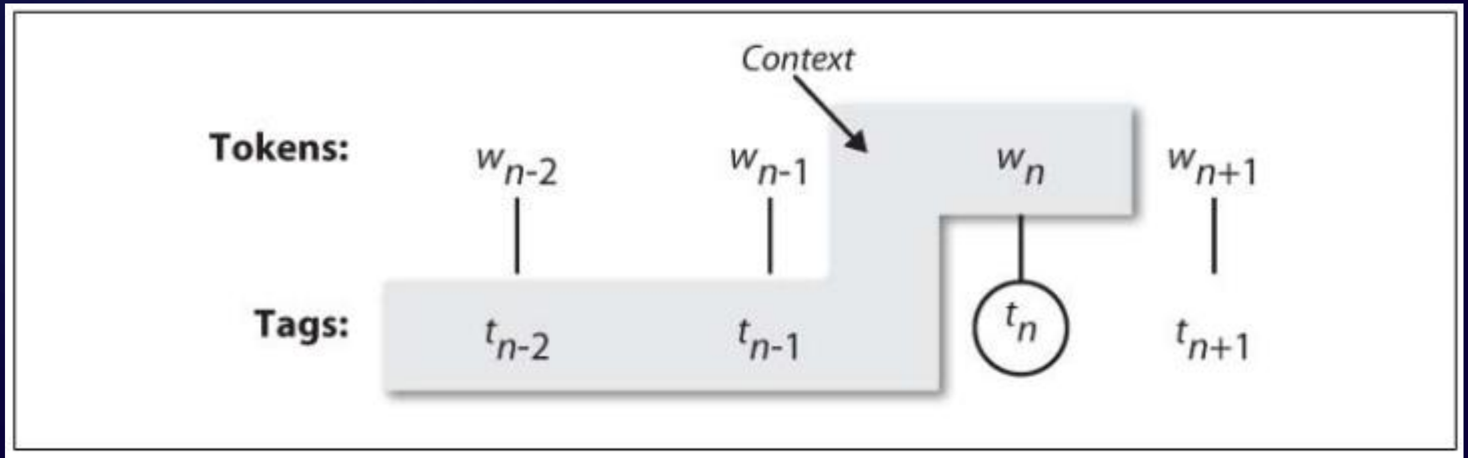
Else eliminate ADV

N-Gram Tagger

- Unigram tagger use 1 item or context, consider 1 token
- This tagging is based on training data – memorized data
- Useless for new text
- Tag for “wind” will be same irrespective of context “the wind” or “to wind”

N-Gram Tagger

- ❖N-Gram tagger is generalized unigram tagger
- ❖Context is current word and POS tag of n-1 preceding tokens
- ❖When n=3, we consider tags of 2 previous words
- ❖N-gram tagger picks the tag most likely in the given context
- ❖As n increases, sparse data problem increases



Default Tagger

- Simplest tagger
- Assigns same tag to each word
- default_tagger = nltk.DefaultTagger('NN')
- default_tagger.tag(tokens) [('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('green', 'NN'), ('eggs', 'NN'), ('and', 'NN'), ('ham', 'NN'), (',', 'NN'), ('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('them', 'NN'), ('Sam', 'NN'), ('I', 'NN'), ('am', 'NN'), ('!', 'NN')]
- Establishes a baseline

Regular Expression Tagger

- Assign tag to token on basis of matching pattern
- Any word ending in ed is the past participle of a verb
- Any word ending with 's is a possessive noun
- >>> patterns = [
- ... (r'.*ing$', 'VBG'), # gerunds
- ... (r'.*ed$', 'VBD'), # simple past
- ... (r'.*es$', 'VBZ'), # 3rd singular present
- ... (r'.*ould$', 'MD'), # modals
- ... (r'.*\'s$', 'NN$'), # possessive nouns
- ... (r'.*s$', 'NNS'), # plural nouns
- ... (r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
- ... (r'.*', 'NN') # nouns (default) ...]

Lookup Tagger

- A lot of high-frequency words do not have the NN tag
- Find the hundred most frequent words and store their most likely tag
- use this information as the model for a "lookup tagger"
- what it does on some untagged input text:
- >>> sent = brown.sents(categories='news')[3] >>> baseline_tagger.tag(sent) [('``', '``'), ('Only', None), ('a', 'AT'), ('relative', None), ('handful', None), ('of', 'IN'), ('such', None), ('reports', None), ('was', 'BEDZ'), ('received', None), ("''", "''"), (',', ','), ('the', 'AT'), ('jury', None), ('said', 'VBD'), (',', ','), ('``', '``'), ('considering', None), ('the', 'AT'), ('widespread', None), ('interest', None), ('in', 'IN'), ('the', 'AT'), ('election', None), (',', ','), ('the', 'AT'), ('number', None), ('of', 'IN'), ('voters', None), ('and', 'CC'), ('the', 'AT'), ('size', None), ('of', 'IN'), ('this', 'DT'), ('city', None), ("''", "''"), ('.', '.')]

Conditional Random Field

- Probabilistic approach to POS tagging
- Input in CRF is a set of features
- Derived using feature function

Features in CRF

- Is the first letter of the word capitalised (Generally Proper Nouns have the first letter capitalised)?
- Is it the first word of the sentence?
- Is it the last word of the sentence
- Does the word contain both numbers and alphabets?
- Does it have a hyphen (generally, adjectives have hyphens - for example, words like fast-growing, slow-moving)
- Is the complete word capitalised?
- Is it a number?
- What are the first four suffixes and prefixes?(words ending with "ed" are generally verbs, words ending with "ous" like disastrous are adjectives)

Backoff Tagger

- Many words in Lookup tagger have been assigned a tag of None
- Use the lookup table first
- If it is unable to assign a tag, then use the default tagger
- This process is known as backoff

Transformation based Tagging

- Also called Brill tagging
- Require training corpus
- Transforms one state to another state by using transformation rules
- Take inspiration from rule based and stochastic tagging
- Based on rules it assigns tags to words
- Rules are automatically learned from data

Transformation based Tagging

- How does it work?
- Start with the solution
- Most beneficial transformation chosen in each cycle
- Apply to the problem transformation chosen in last step

Replace NN with VB when the previous word is TO

Transformation based Tagging

- Example
- The President said he will ask Congress to increase grants to states for vocational rehabilitation
- Two rules are
- Replace NN with VB when the previous word is TO
- Replace TO with IN when the next tag is NNS

Phrase	to	increase	grants	to	states	for	vocational	rehabilitation
Unigram	TO	*NN*	NNS	*TO*	NNS	IN	JJ	NN
Rule 1		*VB*						
Rule 2				*IN*				
Output	TO	VB	NNS	IN	NNS	IN	JJ	NN
Gold	TO	VB	NNS	IN	NNS	IN	JJ	NN

Error Analysis

Look at a confusion matrix

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	—	.2			.7		
JJ	.2	—	3.3	2.1	1.7	.2	2.7
NN		8.7	—				.2
NNP	.2	3.3	4.1	—	.2		
RB	2.2	2.0	.5		—		
VBD		.3	.5			—	4.4
VBN		2.8				2.6	—

See what errors are causing problems

- Noun (NN) vs ProperNoun (NNP) vs Adj (JJ)
- Preterite (VBD) vs Participle (VBN) vs Adjective (JJ)

Evaluation

The result is compared with a manually coded “Gold Standard”

- Typically accuracy reaches 96-97%
- This may be compared with result for a baseline tagger (one that uses no context).

Important: 100% is impossible even for human annotators.

Question

- I do not like green eggs and ham, I do not like them Sam I am!

- What will be the output of default tagger?
- default_tagger = nltk.DefaultTagger('NN')

Answer

- I do not like green eggs and ham, I do not like them Sam I am!

- >>> raw = 'I do not like green eggs and ham, I do not like them Sam I am!'
- >>> tokens = nltk.word_tokenize(raw)
- >>> default_tagger = nltk.DefaultTagger('NN')
- >>> default_tagger.tag(tokens)
- [('I', 'NN'), ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('green', 'NN'),
- ('eggs', 'NN'), ('and', 'NN'), ('ham', 'NN'), (',', 'NN'), ('I', 'NN'),
- ('do', 'NN'), ('not', 'NN'), ('like', 'NN'), ('them', 'NN'), ('Sam', 'NN'),
- ('I', 'NN'), ('am', 'NN'), ('!', 'NN')]

Tagging new words

- Most-frequent-tag approach has a problem!!
- What about words that don't appear in the training set?
- For example, here are some words that occur in a small Brown Corpus test set but not the training set:

Abernathy	all-american	big-boned
absolution	alligator	boathouses
Adrien	asparagus	boxcar
ajar	baby-sitter	
Alicia	bantered	

Tagging new words

- New words added to (newspaper) language 20+ per month
- Plus many proper names …
- Increases error rates by 1-2%
  - Method 1: assume they are nouns
  - Method 2: assume the unknown words have a probability distribution similar to words only occurring once in the training set.
  - Method 3: Use morphological information, e.g., words ending with –ed tend to be tagged VBN.

Part-of-Speech Tagging: Summary

- Languages generally have a small set of **closed class** words that are highly frequent, ambiguous, and **open-class** words like nouns, verbs, adjectives.
  - Various part-of-speech tagsets exist for English, of between 40 and 200 tags.
  - For Turkish, the size of a tagset can be more than 1000.
- **Part-of-speech tagging** is the process of assigning a part-of-speech label to each of a sequence of words.
- The **probabilities in HMM taggers** are estimated by maximum likelihood estimation on tag-labeled training corpora.
  - **Viterbi algorithm** is used for decoding, finding the most likely tag sequence.
  - ***Beam search*** is a variant of Viterbi decoding that maintains only a fraction of high scoring states rather than all states during decoding.