Outline

- Word Classes: What are they and why are they useful?
- Tagsets for English
- Why is Part-of-Speech Tagging Difficult?
- Automated Methods for Part-of-Speech Tagging
  - Rule-based Tagging
  - Hidden Markov Model-based Tagging

Word Classes: What are they?

- Most of us are familiar with the notion of word classes (parts-of-speech, POS) from school grammar lessons, second language learning or dictionaries.
- The following eight parts of speech were defined over 2,000 years ago and form the basis of almost all part of speech inventories
  - ◇ noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- Modern catalogues of word classes range from < 50 to > 150 and vary on level of granularity.

Word Classes: Open vs Closed

- An important distinction is that between closed and open classes.
  - ◇ closed classes are those for which new words do not appear, e.g. prepositions, articles
    of in under a the these and or but
  - ◇ open classes are those for which new words are continually appearing, i.e. nouns, verbs, adjectives and adverbs
    diss shopaholic ecotourist cyberterrorism ram-raid

Word Classes: Definition

- **Word classes** (parts-of-speech, POS) may be defined in terms of
  - ◇ syntactic and morphological distributional properties
    - words that occur in the same constructions or take the same affixes are in the same class
  - ◇ semantic function
    - words that perform the same semantic function are in the same class
- Syntactic/morphological distributional regularities tend to be favoured over semantic regularities as a basis for defining word classes.

Word Classes: Semantic Function

Words fall into the same class if they perform the same semantic function.

- Nouns perform the semantic function of identifying objects, concepts, places and events.

  computer, justice, Sheffield, battle

- Adjectives perform the semantic function of further qualifying objects, concepts, places and events.

  new, black, furious

- For example, in "*The black computer is new*" *computer* identifies an object and *black* qualifies it.
- But, compare:
  - *Computer science is dull.*
  - *Jet black is her favourite colour.*

  Here *computer* qualifies (like an adjective), and *black* identifies (like a noun). Should these two classes be merged?

Word Classes: Distributional regularities

Words fall into the same class if they can appear in the same constructions.

- Consider the construction: It was very ___
  - It was very new.
  - It was very black.
  - * It was very computer.

- Most nouns can function as modifiers in some context – but there are contexts in which adjectives can occur but where nouns cannot.

- Further, while some adjectives (like black) can function as nouns when they name a quality directly, many cannot do so:
  - That black is very attractive.
  - * That furious is very attractive.

Word Classes: Why are they useful?

Word classes are useful because they give us information about a word and its neighbours

- A word's class may determine its pronunciation
  - ◇ cóntent (noun) vs. contént (adjective); óbject (noun) vs. objéct (verb);

  Useful for speech synthesis/speech recognition
- A word's class suggests which morphological affixes it can take – useful in stemming/morphological analysis for, e.g. information retrieval.
- Certain word classes more useful than others in suggesting a document's content – e.g. nouns. Can be used in document gisting.
- Shallow parsing can be accomplished using patterns of POS tags.
  - ◇ E.g. patterns like: ADJ NOUN NOUN can be used for terminology extraction
  - ◇ E.g. patterns like: Mr. PNOUN PNOUN can be used for person name recognisers

  Similar patterns useful for extracting other named entities, dates, times

Word Classes: Why are they useful? (cont)

- Word sense disambiguation assisted by knowing POS
  - ◇ E.g. bridge as noun has at least three senses, but only one as verb
- Word class of current word helps predict word class of next
  - ◇ E.g. possessive pronouns (my, your, his, her, its) likely to be followed by a noun; personal pronouns (I, you, he) likely to be followed by a verb

  This can be exploited in language models for speech recognition
- Parsers typically require word class information in order to discover constituency structure.
- POS-tagged corpora useful in linguistic research
  - ◇ E.g. to find frequencies of particular constructions

Tagsets for English

- Part-of-speech (POS) tagging is critically dependent on the inventory of POS tags
- A number of POS tagsets have been developed:
  - **Brown tagset** 87-tag tagset developed for use with the Brown Corpus
  - **Penn Treebank tagset** 45-tag tagset developed for use with the Penn Treebank
  - **C5 tagset** 61-tag tagset used to tag the British National Corpus
  - **C7 tagset** 146-tag tagset also used to tag the British National Corpus

Example Tagged Text

An example sentence tagged with Penn Treebank tags (from the Treebank):

The/DT trade/NN gap/NN is/VBZ expected/VBN to/TO widen/VB to/TO about/IN $/$ 9/CD billion/CD from/IN July /NNP 's/POS $/$ 7.6/CD billion/CD ,/, according/VBG to/TO a /DT survey/NN by IN MMS/NNP International/NNP ,/, a /DT unit/NN of/IN McGraw-Hill/NNP Inc./NNP ,/, New/NNP York/NNP ./.

Penn Treebank Tagset

The Penn Treebank part-of-speech tags, including punctuation.

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

Why is POS Tagging Difficult?

- POS tagging would be easy if there was a one-to-one mapping from words to POS tags.
- However, this is not the case. Many words have multiple POS tags, i.e. they are **ambiguous** with respect to POS tag. For example:
  - study may be noun or verb
  - retired may be a past tense verb, a past participle or an adjective
  - can may be an auxiliary, a verb or a noun
- In some cases context makes it easy to decide which tag is appropriate for an ambiguous word. In other cases it is not so easy.

POS Tagging: Problem Cases (1)

- **Preposition(IN)/Particle(RP)/Adverb(RB) overlap**. Words like around can be all three.

  Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG
  All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN
  Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 250/CD

- **Labelling noun modifiers**. In the Penn Treebank tagging scheme nouns modifying nouns are labelled as nouns unless they are hypenated, in which case they are adjectives; unless they are hypenated proper nouns in which case they are proper nouns

  cotton/NN sweater/NN
  income-tax/JJ return/NN
  the/DT Gramm-Rudman/NP Act/NP

POS Tagging: Problem Cases (2)

- **Simple Past/Past Participle/Adjective overlap**. Verb forms ending -ed can function to indicate past occurrence of an event (either simple past or past participle) or to express a property.

  They were married/VBN by the vicar yesterday at 5:00.

  They married/VBD young and enjoyed a long life together.

  The married/JJ couple left early.

Why is POS Tagging Difficult?

- How difficult is the problem?
  - ◇ Most words in English have only one tag; but many of the most common words have multiple tags. For example in the Brown corpus:
    - 11.5% of word types are ambiguous
    - >40% of word tokens are ambiguous
- Tag ambiguity for word types in the Brown corpus:

	Original 87-tag corpus		Treebank 45-tag corpus	
1 tag	44,019		38,857	
2 tags	4,967		8844	
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*)

Automated Methods for Part-of-Speech Tagging

- Input to a POS tagging problem is:
  - ◇ A sequence of words that is
    - tokenised (e.g. punctuation separated from words)
    - split into sentences (most taggers tag one sentence at a time)

    E.g. The trade gap is expected to widen to about $ 9 billion from July 's $ 7.6 billion , according to a survey by MMS International , a unit of McGraw-Hill Inc. , New York .
  - ◇ A tagset (e.g. Brown, Penn, C5)
- Output is the same sequence of words with a single best tag added to each word.
  The/DT trade/NN gap/NN is/VBZ expected/VBN to/TO widen/VB to/TO about/IN $/$ 9/CD billion/CD from/IN July/NNP 's/POS $/$ 7.6/CD billion/CD ,/, according/VBG to/TO a/DT survey/NN by/IN MMS/NNP International/NNP ,/, a/DT unit/NN of/IN McGraw-Hill/NNP Inc./NNP ,/, New/NNP York/NNP ./.

Automated Methods for Part-of-Speech Tagging

Tagging algorithms may be classified along a number of dimensions:

- Rule-based vs stochastic/probabilistic
  - Rule-based algorithms apply a set of rules to tag the current word – rule antecedents typically test the word and tag/possible tags of the current word $+/-$ several words
  - Stochastic/probabilistic approaches apply a probabalistic model to compute the probability of the current word having a given tag in the current context

- Hand-crafted vs machine learning
  - In hand-crafted approaches rules are written by human experts
  - Machine learning approaches may be supervised or unsupervised
    - supervised approaches learn rules/probabilistic models from tagged training corpora
    - unsupervised approaches learn models from untagged training corpora

Rule-based Tagging

- Rule-based taggers typically use a two stage architecture:
  1. Use a dictionary to assign all possible tags to each word
  2. Apply rules to eliminate all but one tag for each word

- One well-known rule-based tagger is the **EngCG** tagger (English constraint grammar) (Karlsson et al. 1995, Voutilainen, 1999)

- **EngCG** lexicon
  - contains $\sim$ 56,000 entries.
  - Entries have the form:

Word	POS	Additional POS Features
smaller	ADJ	COMPARATIVE
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG

Rule-based Tagging (cont)

- Stage 1 returns all possible POS tags for each token
  E.g. for the phrase Pavlov had shown that salivation ... the following is returned (one line per possible tag/correct tag in bold)

Pavlov	**PAVLOV N NOM SG PROPER**
had	**HAVE V PAST VFIN SVO**
	HAVE PCP2 SVO
shown	**SHOW PCP2 SV00 SVO SV**
that	ADV
	PRON DEM SG
	DET CENTRAL DEM SG
	CS
salivation	**N NOM SG**

Rule-based Tagging (cont)

- In stage 2 constraints are applied to output of stage 1 to rule out incorrect tags (i.e. tags that are inconsistent with the context)
- An example constraint used to rule out all readings of that except the adverbial one (as in isn't that odd):

```
ADVERVBIAL-THAT-RULE
Given input "that"
if
    (+1 A/ADV/QUANT); /* if next word is adj, adverb, or quantifier */
    (+2 SENT-LIM); /* and following which is a sentence boundary, */
    (NOT -1 SVOC/A); /* and the previous word is not a verb like */
        /* 'consider' which allows adjs as object complements */
then eliminate non-ADV tags
else eliminate ADV tag
```

  - ◇ First two conditions of rule antecedent check that that immediately precedes a sentence final adjective, adverb or quantifier.
  - ◇ Third condition eliminates cases of that preceded by verbs like consider or believe that can take adjectives as object complements (as in I consider that odd)

Rule-based Tagging (cont)

- Another rule for that expresses the constraint that the complementizer tag for that is appropriate if
  - ◇ the previous word is a verb that expects a clausal complement (believe, think, show)
  - ◇ that is followed by the beginning of a noun phrase and a finite verb
- EngCG-2 contains 3,744 manually authored constraints

Hidden Markov Model Tagging

- Probabilistic tagging dates back to 1965
- Hidden Markov Models (HMMs) are a widely explored approach
- View use of HMMs for POS tagging as a special case of Bayesian inference
- POS tagging can be treated as a sequence classification task:
  - Observe a sequence of words and assign them a sequence of POS tags

Hidden Markov Model Tagging (cont)

- What is the best sequence of tags corresponding to a sequence of words?
- Of all possible tag sequences t_1^n to assign to the word sequence w_1^n we want the one that is most probable given the observed word sequence w_1^n
  - ◇ i.e. we want the tag sequence such that $P(t_1^n|w_1^n)$ is highest.
- Using $\hat{}$ to denote our estimate of the best tag sequence we have:

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n)$$

- How can we compute $P(t_1^n|w_1^n)$ for a given tag sequence t_1^n and word sequence w_1^n?

Hidden Markov Model Tagging (cont)

- Use **Bayes' rule**:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

to transform this into something which is easier to compute:

$$\begin{aligned}
\hat{t}_1^n &= \arg\max_{t_1^n} P(t_1^n|w_1^n) \\
&= \arg\max_{t_1^n} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)} \\
&= \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)
\end{aligned}$$

- In words: the most probable tag sequence $\hat{t}_1^n$ given a word sequence w_1^n can be computed by taking the product of two probabilities for each tag sequence and choosing the tag sequence for which this probability is greatest.

Hidden Markov Model Tagging (cont)

- In the equation

$$\hat{t}_1^n = \arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$$

- The two terms are:
  - The **prior probability** of the tag sequence $P(t_1^n)$; and
  - The **likelihood** of the word string $P(w_1^n|t_1^n)$

- But $\arg\max_{t_1^n} P(w_1^n|t_1^n)P(t_1^n)$ is still hard to compute.

Hidden Markov Model Tagging (cont)

- HMM taggers make two simplifying assumptions:
  1. the probability of a word appearing is dependent only on its own POS tag and is independent of other words and tags around it. i.e.

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

  2. The probability of a tag appearing is dependent only on the preceding tag (the **bigram** assumption).

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

- Thus we have:

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n|w_1^n) \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Hidden Markov Model Tagging (cont)

- The **tag transition probabilities** $P(t_i|t_{i-1})$ represent the probabilty of a tag given the preceding tag.
  - E.g. since determiners like the frequently appear before nouns and adjectives in phrases such as the clash and the yellow brick road we expect the probabilities $P(\text{NN}|\text{DT})$ and $P(\text{JJ}|\text{DT})$ to be high.
  - Can compute maximum likelihood estimate (MLE) of tag transition probability using a corpus whose words have been labelled with POS

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

  - E.g. MLE of transition probability $P(\text{NN}|\text{DT})$ is

$$P(\text{NN}|\text{DT}) = \frac{C(\text{DT}, \text{NN})}{C(\text{DT})}$$

    i.e. the proportion of occurrences of DT in which it is followed by NN

Hidden Markov Model Tagging (cont)

- The **word likelihood probabilities** $P(w_i|t_i)$ represent the probabilty of a word given a particular tag.
  - ◇ e.g. if we see the tag VBZ we might guess the associated word is is since is is a very common word in English
  - ◇ Can again compute maximum likelihood estimate of a word likelihood probability using a POS-labelled corpus

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

  - ◇ e.g. MLE of word likelihood probability $P(is|\text{VBZ})$ is

$$P(is|\text{VBZ}) = \frac{C(\text{VBZ}, is)}{C(\text{VBZ})}$$

    i.e. the proportion of occurrences of VBZ in which it is associated with the word is

Computing the Most Likely Tag Sequence: Example

- Aim: show how for a sentence with ambiguous taggings the correct tag sequence has higher probability than one of the wrong taggings.
- Consider the following sentence and two possible taggings, using the Brown corpus tagset:

  (1) Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NR
  (2) Secretariat/NNP is/VBZ expected/VBN to/TO race/NN tomorrow/NR

- Note these taggings differ only for the word race. In this context the first tagging is correct since race here is a verb and not a noun (which it would be in I shall go to the race tomorrow).
  - ◇ Note also that there are other tag ambiguities here. E.g. expected could be JJ, VBN or VBD. We will ignore these.

Computing the Most Likely Tag Sequence: Example (cont)

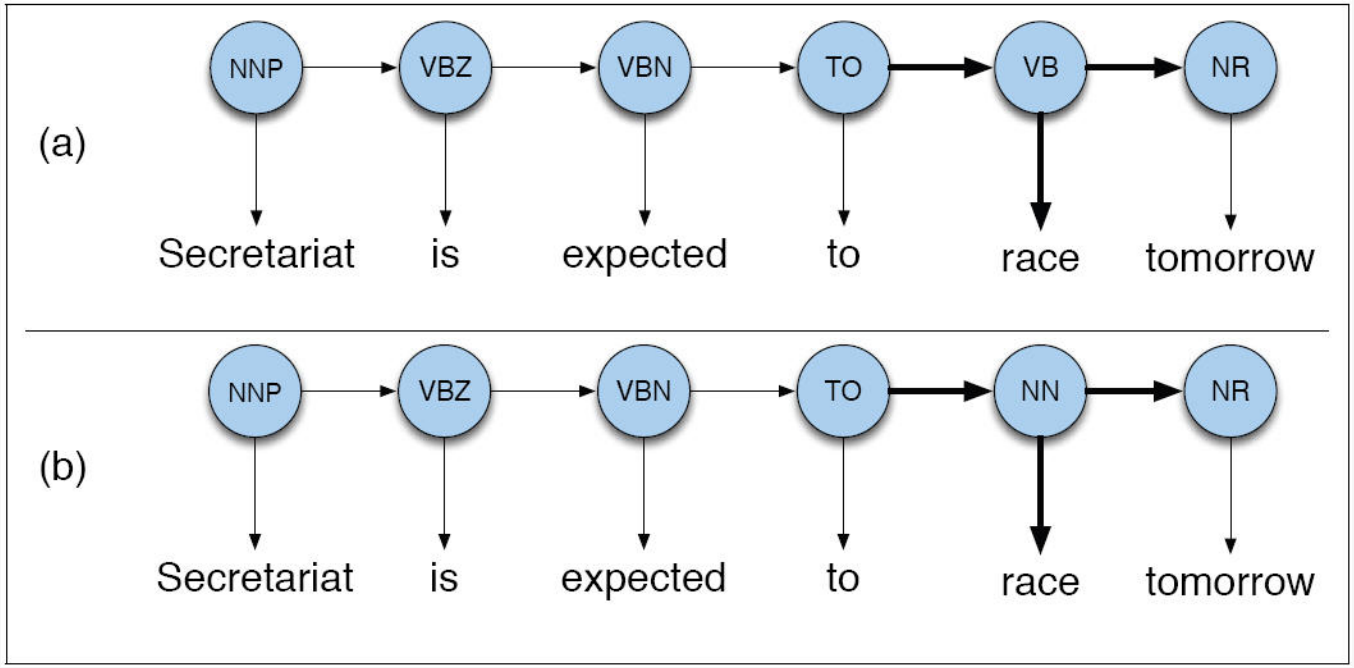
- Recall that in HMM tagging we are approximating the correct tag sequence by:

$$\hat{t}_1^n \approx \arg\max_{t_1^n} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

- Thus the probabilities that are going to make a difference in computing the probabilities of the two tag sequences are:
  - ◇ The tag transition probabilities:
    - P(VB|TO) and P(NR|VB) from the tag sequence in (1)
    - P(NN|TO) and P(NR|NN) from the tag sequence in (2)
  - ◇ The word likelihoods:
    - P(race|VB) from the tag sequence in (1)
    - P(race|NN) from the tag sequence in (2)

Computing the Most Likely Tag Sequence: Example (cont)

- Graphical representation of differing probabilities



Computing the Most Likely Tag Sequence: Example (cont)

- Suppose we use the 1M word tagged Brown corpus to calculate the maximum likelihood estimate of these probabilities. We get these results

$$P(NN|TO) = .00047 \quad P(race|NN) = .00057 \quad P(NR|VB) = .0027$$
$$P(VB|TO) = .83 \quad P(race|VB) = .00012 \quad P(NR|NN) = .0012$$

  Notes:
  - Verb much more likely to follow infinite marker TO than noun – as expected
  - Adverbial noun (NR) more likely to follow verb than noun – as expected
  - race more likely to appear as a noun than as a verb
- Multiplying the appropriate probabilities to compute the differing parts of the tag sequences:

$$P(VB|TO)P(NR|VB)P(race|VB) = .00000027$$
$$P(NN|TO)P(NR|NN)P(race|NN) = .00000000032$$

  Thus the HMM bigram tagging model yields the correct tagging in sequence (1).

Formalizing HMM Taggers

- A Hidden Markov Model is a weighted finite state automaton
  - ◇ Like a FSA except that transitions between states have associated probabilities
  - ◇ Sum of probabilities leaving any state equal to 1
- States in HMM used to model *hidden* events or variables viewed as causal factors in a probabilistic model (e.g. POS tags)
- States generate or emit observed events according to a probability distribution across a set of possible observations (e.g. words).

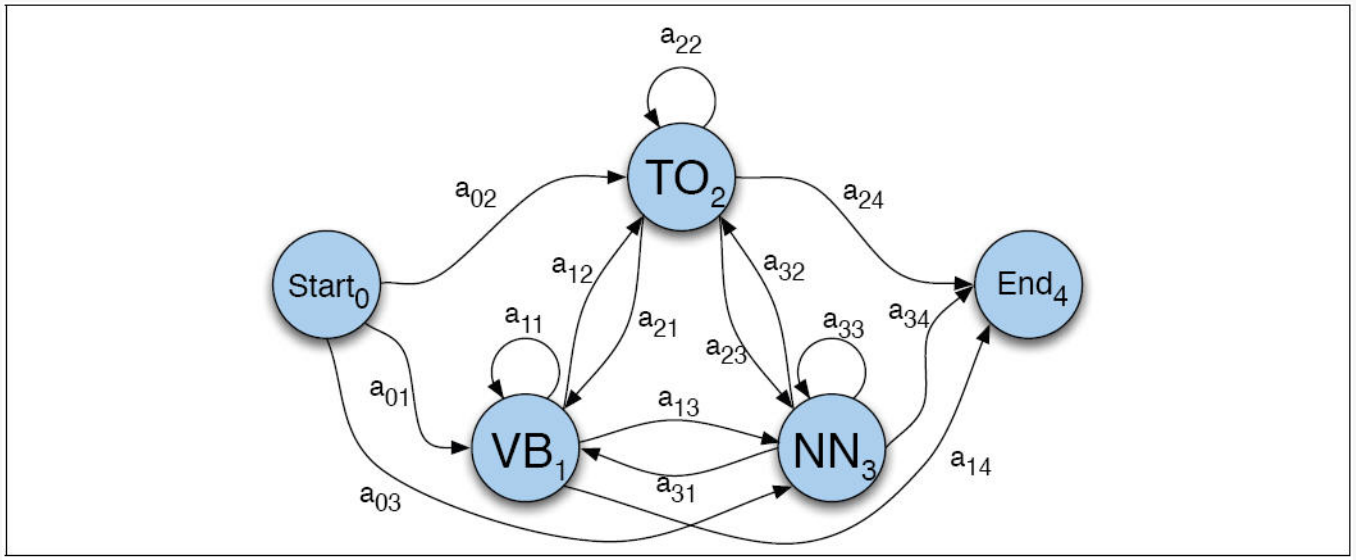
Formalizing HMM Taggers

- Formally a HMM is specified by the following:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**.
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$.
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** that are not associated with observations, together with transition probabilities $a_{01} a_{02} \ldots a_{0n}$ out of the start state and $a_{1F} a_{2F} \ldots a_{nF}$ into the end state.

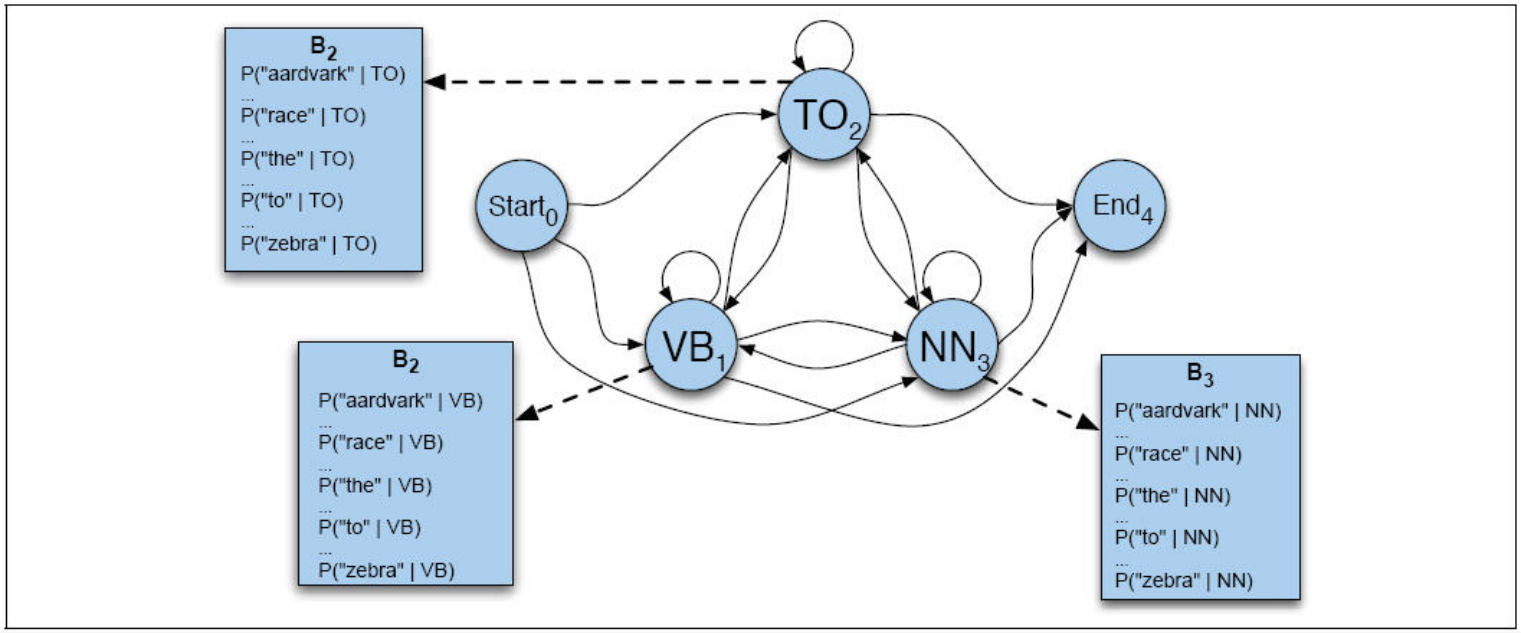
Formalizing HMM Taggers

- For POS tagging
  - ◇ HMM states model tags and the transition probabilities in the HMM model tag transition probabilities (prior probabilities)
  - ◇ HMM observations are the words to be tagged and observation likelihoods model the word likelihoods.
- The weighted FSA showing the hidden states of the HMM and the transition probabilities:



Formalizing HMM Taggers (cont.)

- Observation likelihoods for the preceeding HMM. Each state has an associated vector or probabilities, one for each possible observation word.



The Viterbi Algorithm

- Given an HMM containing hidden variables and an observation sequence the task of determining which sequence of variables is the underlying source of the observed sequence is called the **decoding** task.

- The **Viterbi** algorithm is the most common algorithm used for decoding HMMs, both for POS tagging and for speech recognition.
  - The Viterbi algorithm is an application of **dynamic programming**

- Given an HMM and an observation sequence, the Viterbi algorithm returns the state-path through the HMM which assigns maximum likelihood to the observation sequence.

The Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*

create a path probability matrix *viterbi*$[N+2,T]$
for each state s **from** 1 **to** N **do** ; initialization step
  viterbi$[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  backpointer$[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s}$
$viterbi[q_F,T] \leftarrow \max_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
$backpointer[q_F,T] \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T] * a_{s,q_F}$; termination step
return the backtrace path by following backpointers to states back in time from *backpointer*$[q_F,T]$

- Notes:
  - $a[s',s]$ is the transition probability from the previous state s' to the current state s
  - $b_s(o_t)$ is the observation likelihood of o_t given s

The Viterbi Algorithm – Example

- Suppose we want to tag the sentence – I want to race.
- We have the following tag transition probabilities $P(t_i|t_{i-1})$ estimated from the Brown Corpus – for the HMM a array

	VB	TO	NN	PPSS
<s>	.019	.0043	.041	.067
VB	.0038	.035	.047	.0070
TO	.83	0	.00047	0
NN	.0040	.016	.087	.0045
PPSS	.23	.00079	.0012	.00014

and the following observation likelihoods for the HMM b table

	I	want	to	race
VB	0	.0093	0	.00012
TO	0	0	.99	0
NN	0	.000054	0	.00057
PPSS	.37	0	0	0

The Viterbi Algorithm – Example

- The algorithm begins by initialising the *viterbi*[] array with
  - ◇ one row for each state in the HMM and
  - ◇ one column for each word in the observation sequence plus one extra row at the beginnning and the end

- Begin in first column. Set probability of start state to 1 and other states to 0.

- Move to next column. Compute probability of moving from every state in column $t-1$ to each state in column t.

The Viterbi Algorithm – Example

- For each state q_j at time t (observation o_t) $viterbi[j, t]$ is given by

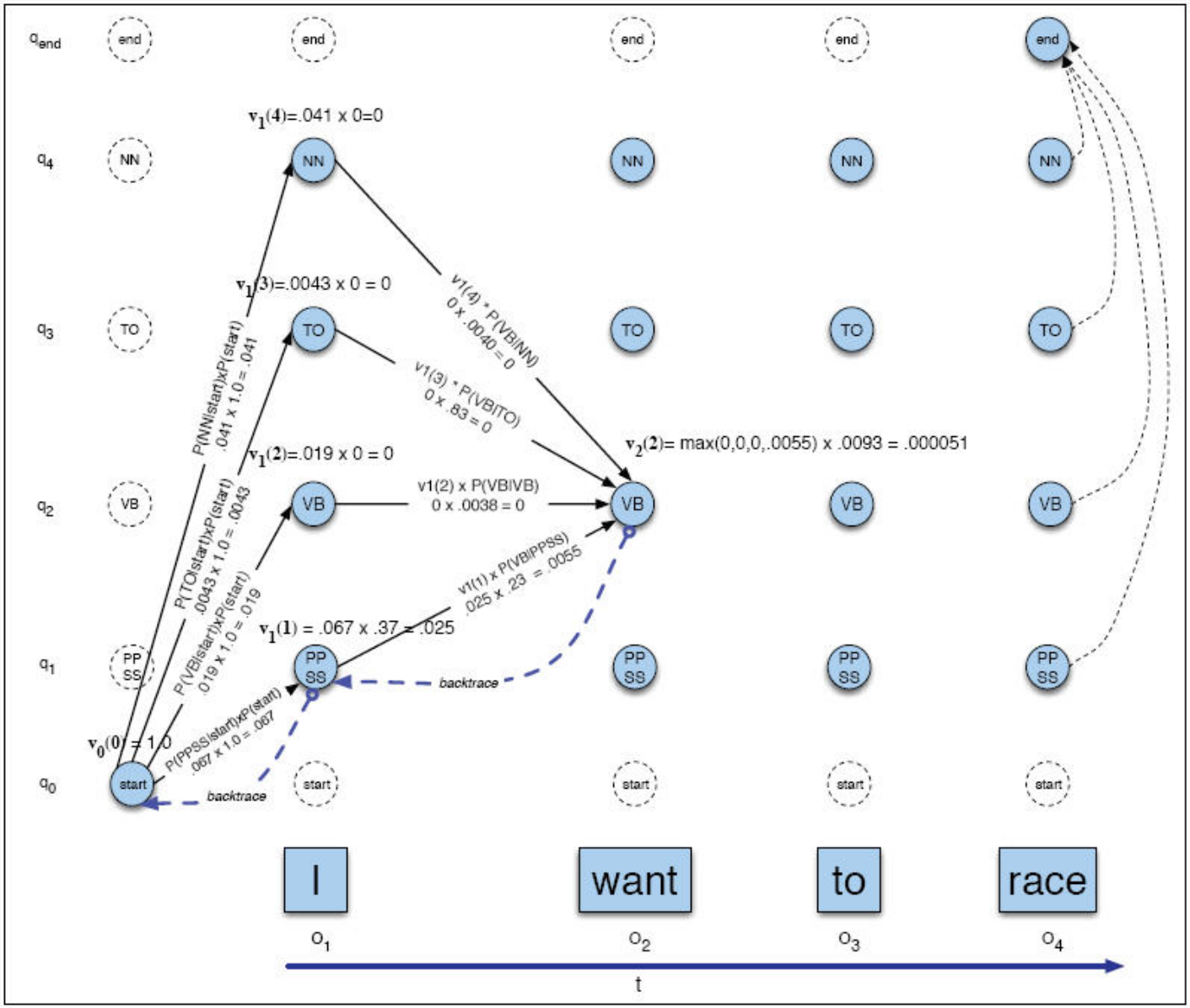
$$viterbi[j, t] = \max_{1 \leq i \leq N} viterbi[i, t-1] * a_{ij} * b_j(o_t)$$

  The three factors on the right hand side here are:

  - $viterbi[i, t-1]$ 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

- **Back tracing** from the end state allows us to reconstruct the correct tag sequence

The Viterbi Algorithm – Example



Summary

- Word classes and tagsets
- Why PoS tagging is difficult
- Automated methods for PoS tagging
  - Rule-based tagging
  - HMM tagging
  - Viterbi decoding