Introduction to NLP

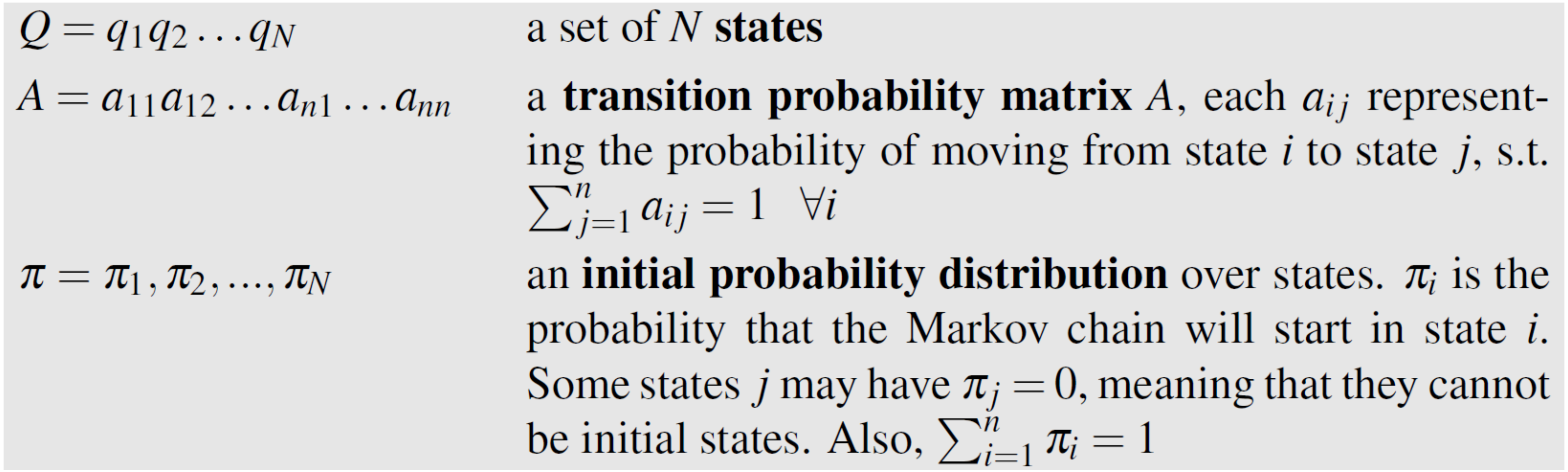
221.

Hidden Markov Models

Markov Models

- Sequence of random variables that aren’t independent
- Examples
  - Weather reports
  - Text
  - Stock market numbers

Definition



$Q = q_1 q_2 \ldots q_N$	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. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Properties

- Limited horizon:

  $P(X_{t+1} = s_k | X_1,\ldots,X_t) = P(X_{t+1} = s_k | X_t)$

- Time invariant (stationary)

  $P(X_{t+1} = s_k | X_t) = P(X_2=s_k | X_1)$

- Definition
  - in terms of a transition matrix A and initial state probabilities Π.

Andrey Markov

Example

$$\begin{tikzcd}
f \arrow[d, "1.0"'] & a \arrow[loop above, "0.2"] \arrow[r, "0.8"] & b \arrow[loop above, "1.0"] \\
e \arrow[r, "1.0"] & d \arrow[ul, "0.2"'] \arrow[u, "0.7"'] \arrow[r, bend left, "0.1"] & c \arrow[u, "0.3"] \arrow[l, bend left, "0.7"] \\
 & \arrow[u, "\textit{start}"'] &
\end{tikzcd}$$

Visible MM

$P(X_1,\ldots X_T) = P(X_1)\ P(X_2|X_1)\ P(X_3|X_1,X_2)\ \ldots\ P(X_T|X_1,\ldots,X_{T-1})$

$= P(X_1)\ P(X_2|X_1)\ P(X_3|X_2)\ \ldots\ P(X_T|X_{T-1})$

$$= \pi_{X_1} \prod_{t=1}^{T-1} a_{X_t X_{t+1}}$$

$P(d, a, b) = P(X_1=d)\ P(X_2=a|X_1=d)\ P(X_3=b|X_2=a)$

$= 1.0 \times 0.7 \times 0.8$

$= 0.56$

Hidden Markov Models

- Motivation
  - Observing a sequence of symbols
  - The sequence of states that led to the generation of the symbols is hidden
  - The states correspond to hidden (latent) variables
- Definition
  - Q = states
  - O = observations, drawn from a vocabulary
  - q_0,q_f = special (start, final) states
  - A = state transition probabilities
  - B = symbol emission probabilities
  - Π = initial state probabilities
  - μ = (A,B,Π) = complete probabilistic model

Hidden Markov Models

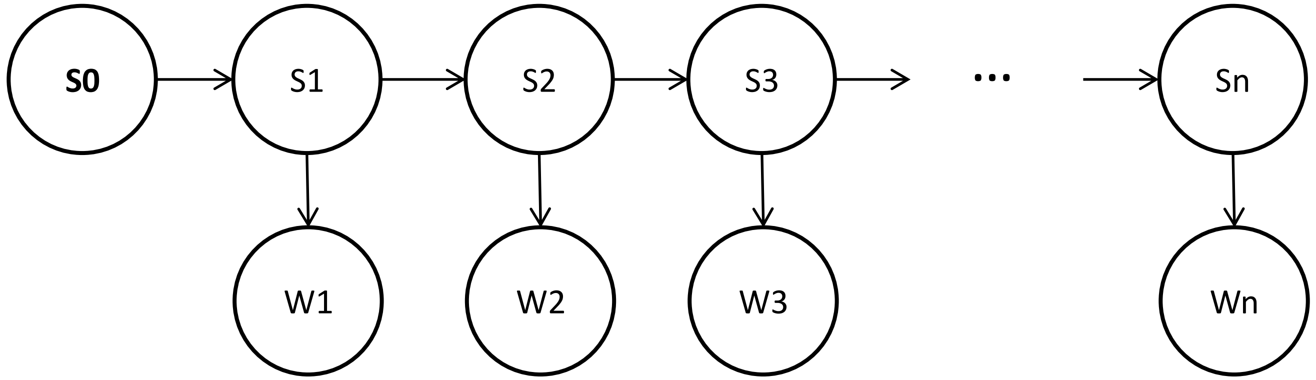
- Uses
  - Part of speech tagging
  - Speech recognition
  - Gene sequencing

Noisy Channel Model Applications

- Text-to-text (e.g., text summarization)
- Speech recognition
- Spelling Correction
- Optical Character Recognition
    - P(text|pixels) = P(text) P(pixels|text)

Hidden Markov Models

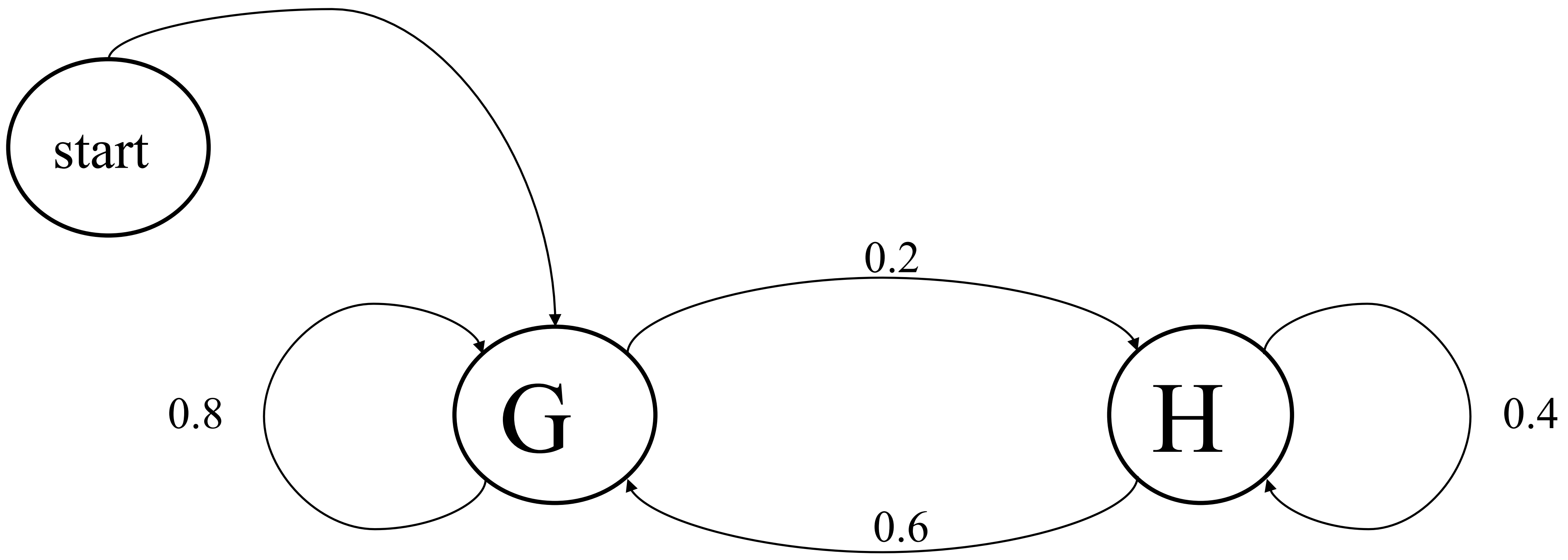
- Can be used to model state sequences and observation sequences
- Example:
  - $P(\mathbf{s},\mathbf{w}) = \prod_i P(s_i|s_{i-1})P(w_i|s_i)$



Generative Algorithm

- Pick start state from Π
- For t = 1..T
    - Move to another state based on A
    - Emit an observation based on B

State Transition Probabilities


$$\begin{tikzcd}
\text{start} \arrow[d] & & \\
G \arrow[loop left, "0.8"] \arrow[rr, bend left, "0.2"] & & H \arrow[ll, bend left, "0.6"] \arrow[loop right, "0.4"]
\end{tikzcd}$$

Emission Probabilities

symbol states

- $P(O_t=k|X_t=s_i,X_{t+1}=s_j) = b_{ijk}$

symbols

states

	x	**y**	**z**
G	0.7	0.2	0.1
H	0.3	0.5	0.2

All Parameters of the Model

- Initial
  - P(G|start) = 1.0, P(H|start) = 0.0
- Transition
  - P(G|G) = 0.8, P(G|H) = 0.6, P(H|G) = 0.2, P(H|H) = 0.4
- Emission
  - P(x|G) = 0.7, P(y|G) = 0.2, P(z|G) = 0.1
  - P(x|H) = 0.3, P(y|H) = 0.5, P(z|H) = 0.2

Observation sequence "yz"

- Starting in state G (or H), P(yz) = ?
- Possible sequences of states:
  - GG
  - GH
  - HG
  - HH
- P(yz) = P(yz|GG) + P(yz|GH) + P(yz|HG) + P(yz|HH) =

  = .8 x .2 x .8 x .1

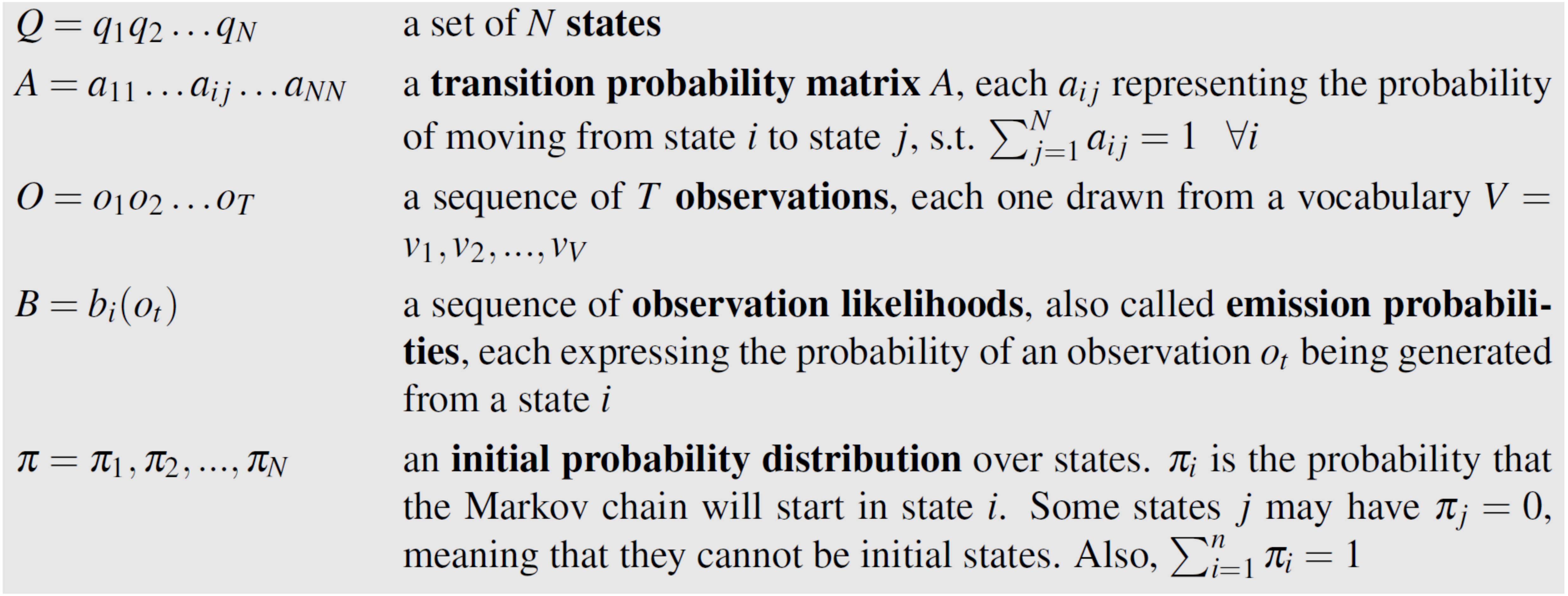
  + .8 x .2 x .2 x .2

  + .2 x .5 x .4 x .2

  + .2 x .5 x .6 x .1

  = .0128+.0064+.0080+.0060 =.0332

Hidden Markov Model



$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	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
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

States and Transitions

- An HMM is essentially a weighted finite-state transducer
  - The states encode the most recent history
  - The transitions encode likely sequences of states
    - e.g., Adj-Noun or Noun-Verb
    - or perhaps Art-Adj-Noun
  - Use MLE to estimate the probabilities
- Another way to think of an HMM
  - It’s a natural extension of Naïve Bayes to sequences

Emissions

- Estimating the emission probabilities
  - Harder than transition probabilities (why?)
  - There may be novel uses of word/POS combinations
- Suggestions
  - It is possible to use standard smoothing
  - As well as heuristics (e.g., based on the spelling of the words)

Sequence of Observations

- The observer can only see the emitted symbols
- Observation likelihood
  - Given the observation sequence S and the model $\mu = (A,B,\Pi)$, what is the probability $P(S|\mu)$ that the sequence was generated by that model.
- Being able to compute the probability of the observations sequence turns the HMM into a language model

Tasks with HMM

- Given $\mu = (A,B,\Pi)$, find $P(O|\mu)$
  - Uses the Forward Algorithm
- Given O, μ, find $(X_1,\ldots X_{T+1})$
  - Uses the Viterbi Algorithm
- Given O and a space of all possible $\mu_{1..m}$, find model μ_i that best describes the observations
  - Uses Expectation-Maximization

Inference

- Find the most likely sequence of tags, given the sequence of words
  - $t^* = \text{argmax}_t P(t|w)$
- Given the model μ, it is possible to compute P (t|w) for all values of t
  - In practice, there are way too many combinations
- Greedy Search
- Beam Search
  - One possible solution
  - Uses partial hypotheses
  - At each state, only keep the k best hypotheses so far
  - May not work

Viterbi Algorithm

- Find the best path up to observation i and state s
- Characteristics
  - Uses dynamic programming
  - Memoization
  - Backpointers

The **Viterbi** algorithm was first applied to speech and language processing in the context of speech recognition by Vintsyuk (1968) but has what Kruskal (1983) calls a "remarkable history of multiple independent discovery and publication". Kruskal and others give at least the following independently-discovered variants of the algorithm published in four separate fields:

Citation	**Field**
Viterbi (1967)	information theory
Vintsyuk (1968)	speech processing
Needleman and Wunsch (1970)	molecular biology
Sakoe and Chiba (1971)	speech processing
Sankoff (1972)	molecular biology
Reichert et al. (1973)	molecular biology
Wagner and Fischer (1974)	computer science

Algorithm 12 Generative process for the hidden Markov model

```
$y_0 \leftarrow \Diamond, \quad m \leftarrow 1$
repeat
    $y_m \sim \text{Categorical}(\boldsymbol{\lambda}_{y_{m-1}})$    ▷ sample the current tag
    $w_m \sim \text{Categorical}(\boldsymbol{\phi}_{y_m})$    ▷ sample the current word
until $y_m = \blacklozenge$    ▷ terminate when the stop symbol is generated
```

Viterbi Algorithm

Finally, we can give a formal definition of the Viterbi recursion as follows:

1. **Initialization:**

$$\begin{aligned} v_1(j) &= \pi_j b_j(o_1) \qquad 1 \leq j \leq N \\ bt_1(j) &= 0 \qquad\qquad\quad 1 \leq j \leq N \end{aligned}$$

2. **Recursion**

$$\begin{aligned} v_t(j) &= \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T \\ bt_t(j) &= \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T \end{aligned}$$

3. **Termination:**

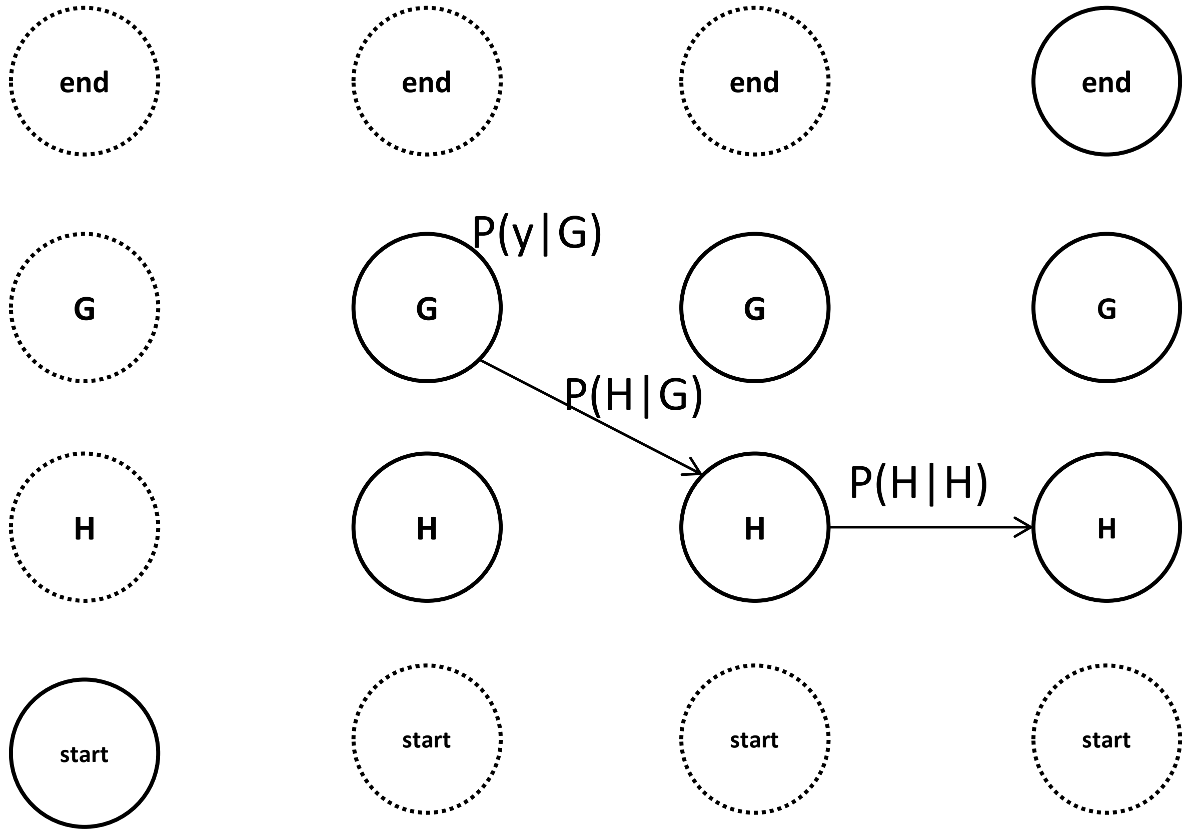
$$\text{The best score:} \quad P* = \max_{i=1}^{N} v_T(i)$$

$$\text{The start of backtrace:} \quad q_T* = \operatorname*{argmax}_{i=1}^{N} v_T(i)$$

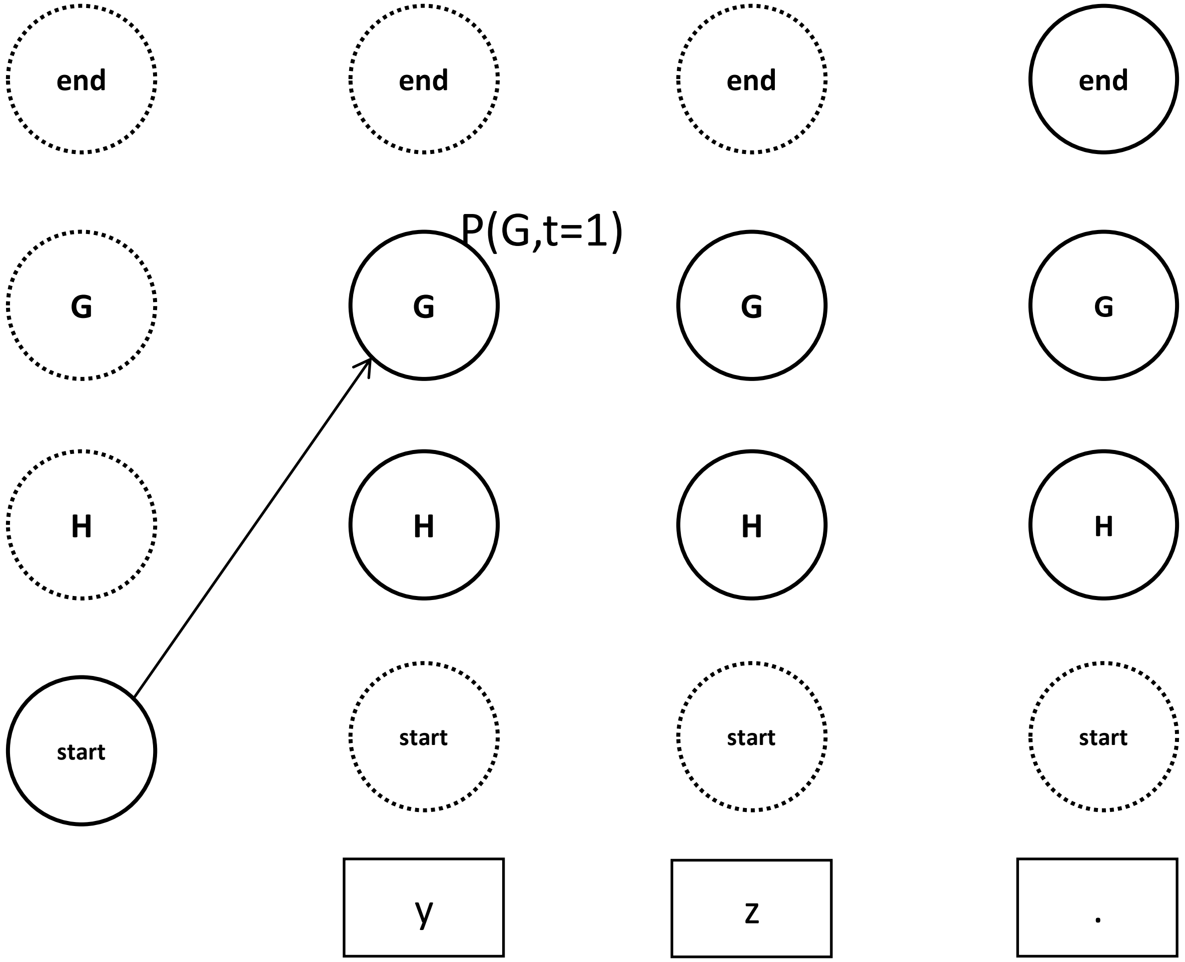
Algorithm 11 The Viterbi algorithm. Each $s_m(k, k')$ is a local score for tag $y_m = k$ and $y_{m-1} = k'$.

```
for $k \in \{0, \ldots K\}$ do
    $v_1(k) = s_1(k, \Diamond)$
for $m \in \{2, \ldots, M\}$ do
    for $k \in \{0, \ldots, K\}$ do
        $v_m(k) = \max_{k'} s_m(k, k') + v_{m-1}(k')$
        $b_m(k) = \text{argmax}_{k'} s_m(k, k') + v_{m-1}(k')$
$y_M = \text{argmax}_k s_{M+1}(\blacklozenge, k) + v_M(k)$
for $m \in \{M - 1, \ldots 1\}$ do
    $y_m = b_m(y_{m+1})$
return $\boldsymbol{y}_{1:M}$
```

HMM Trellis

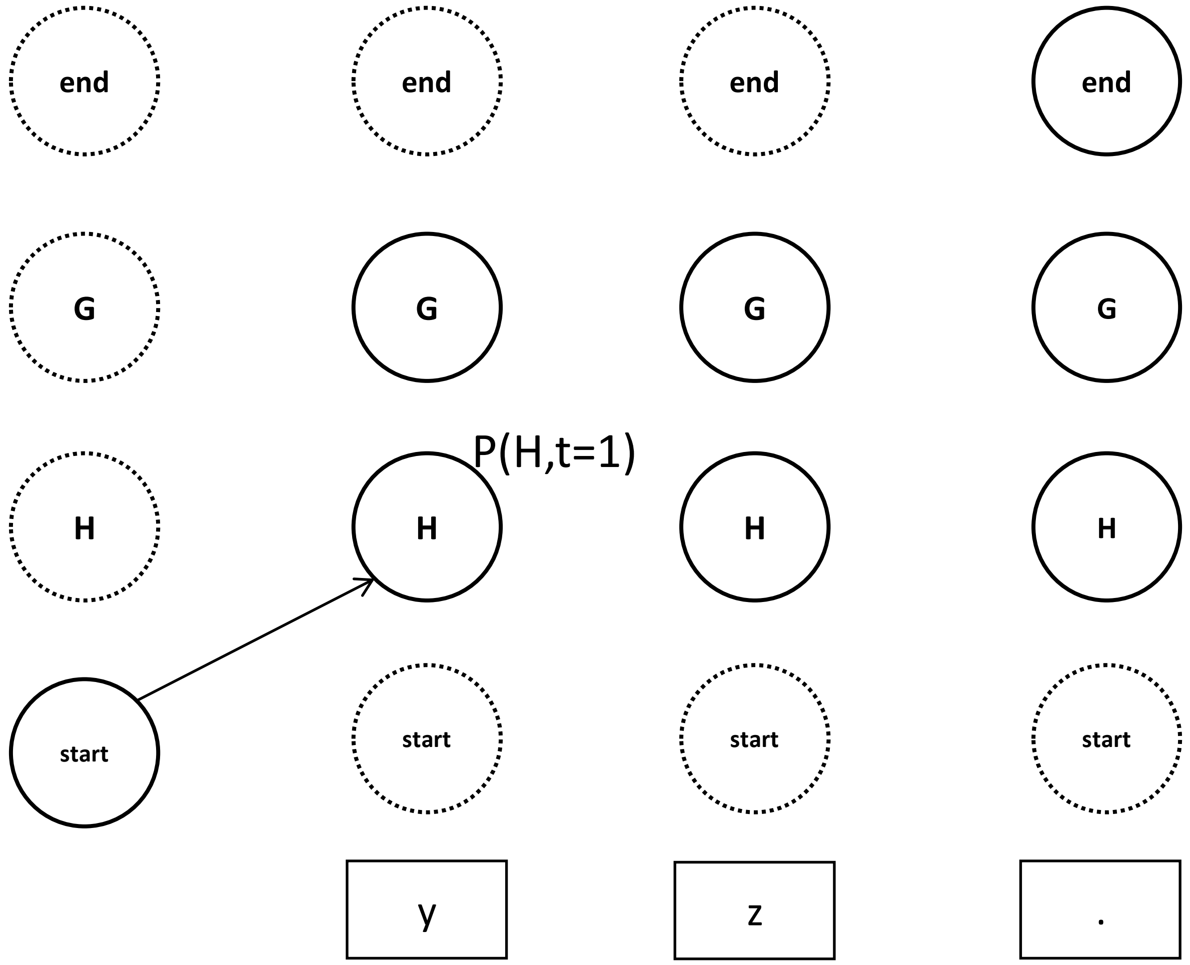


HMM Trellis



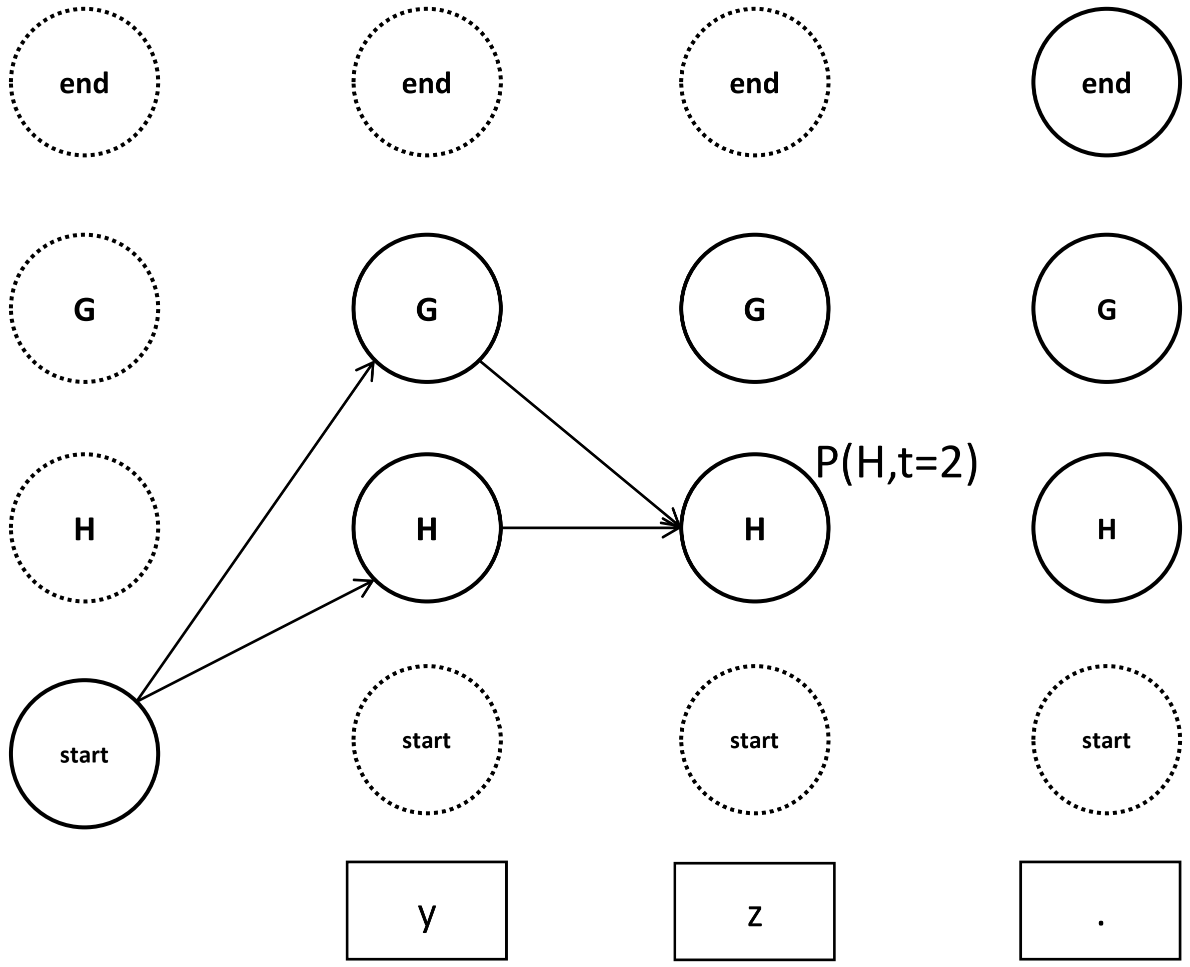
P(G,t=1) =
P(start) x P(G|start) x P(y|G)

HMM Trellis



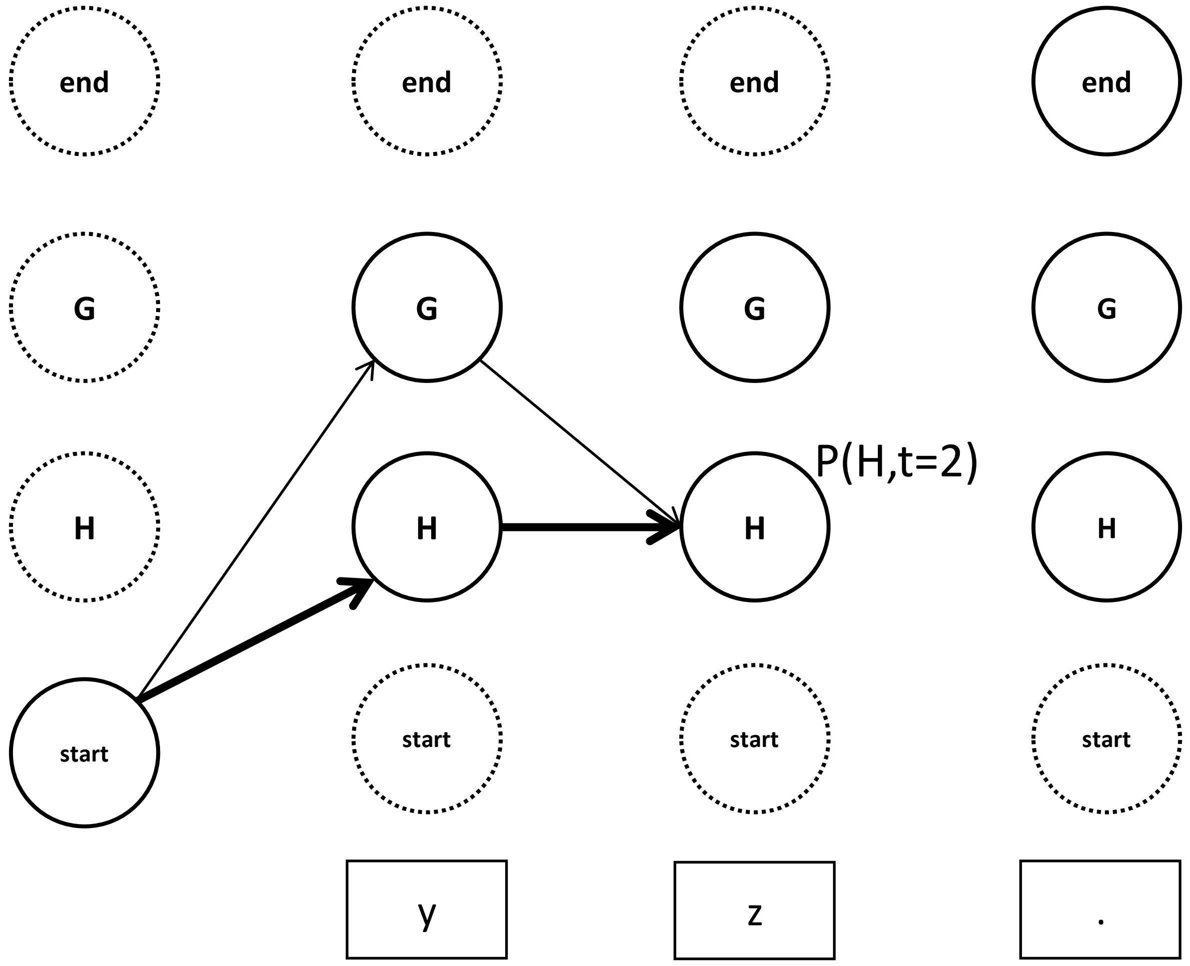
P(H,t=1) =
P(start) x P(H|start) x P(y|H)

HMM Trellis

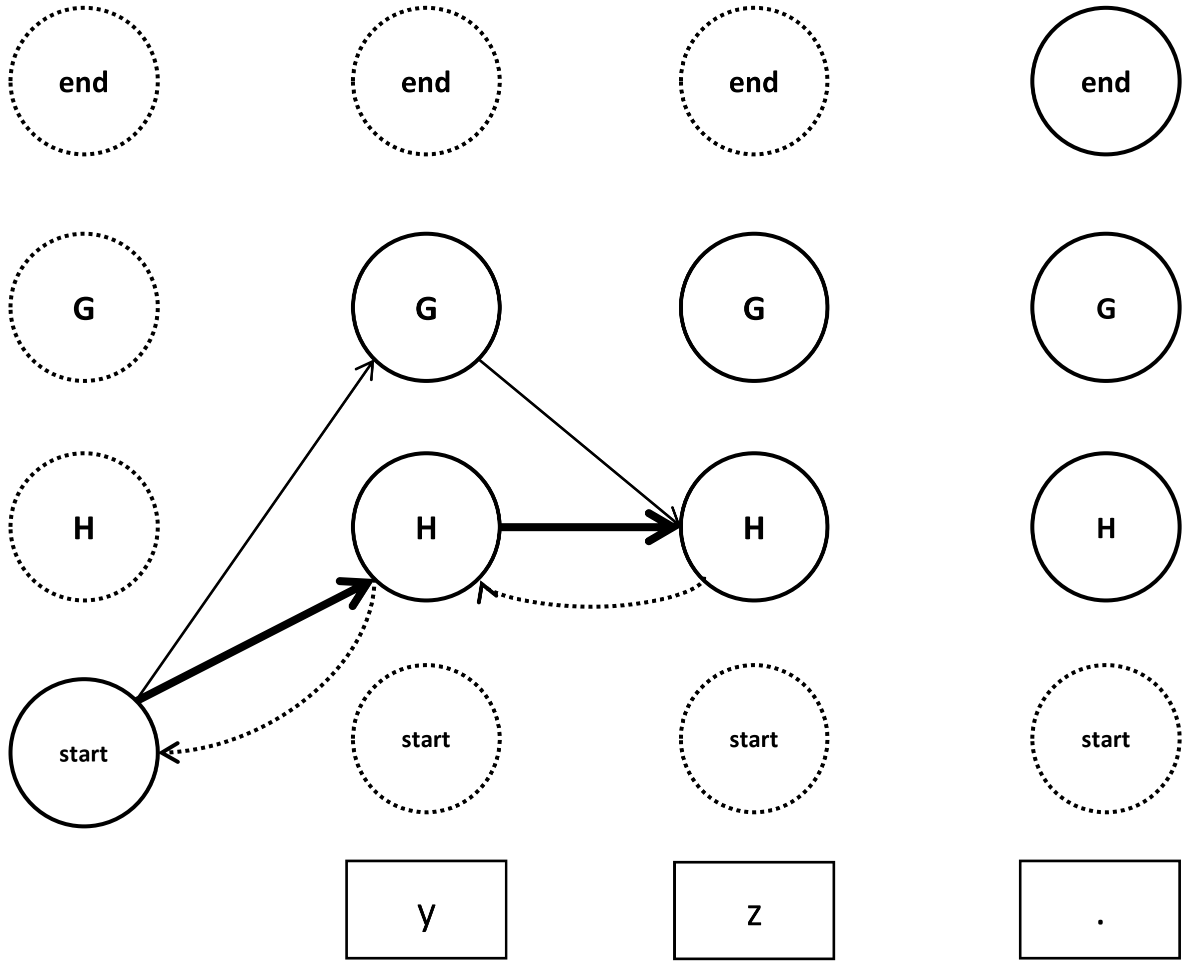


P(H,t=2) =
max (P(G,t=1) x P(H|G) x P(z|H),
P(H,t=1) x P(H|H) x P(z|H))

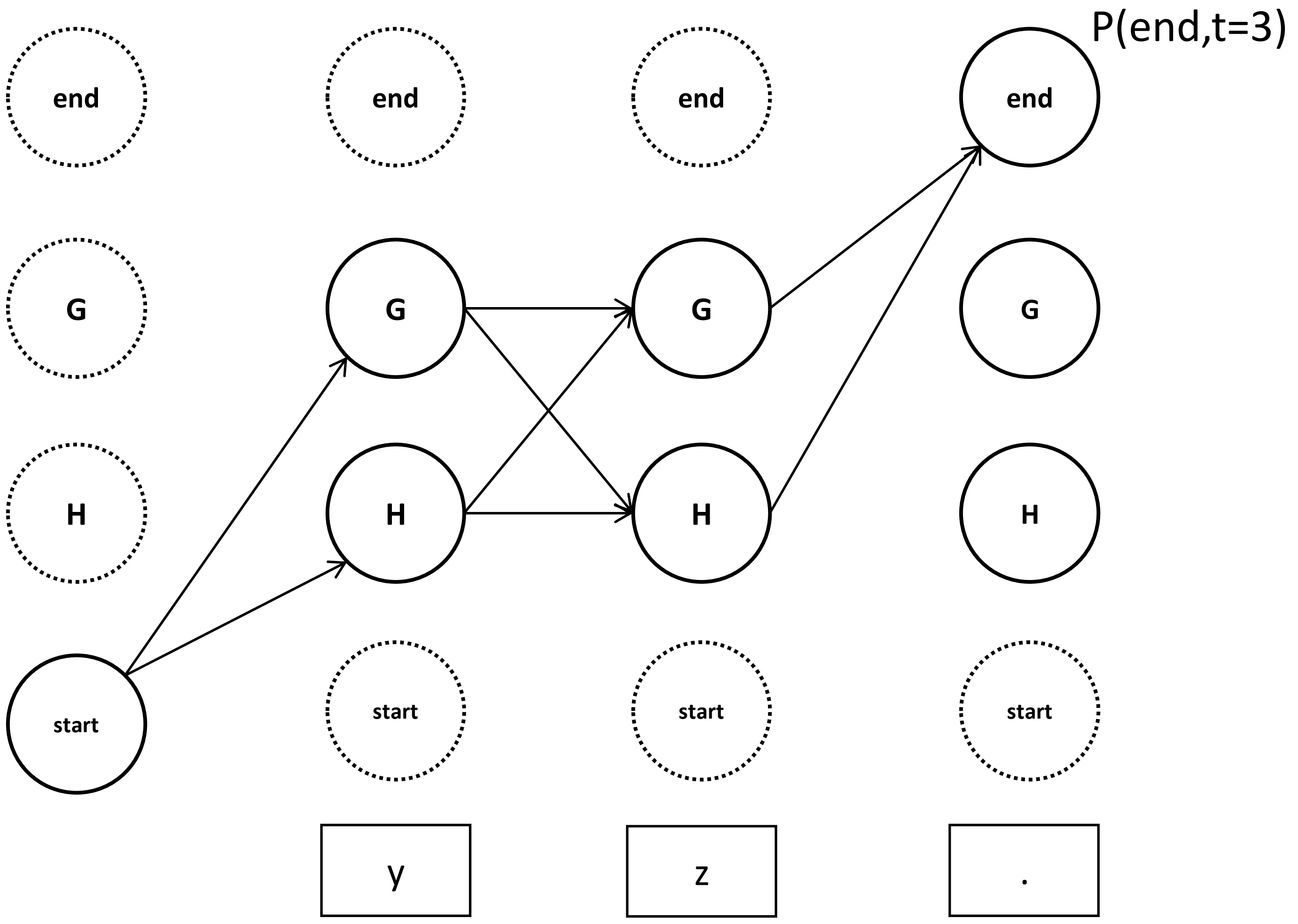
HMM Trellis



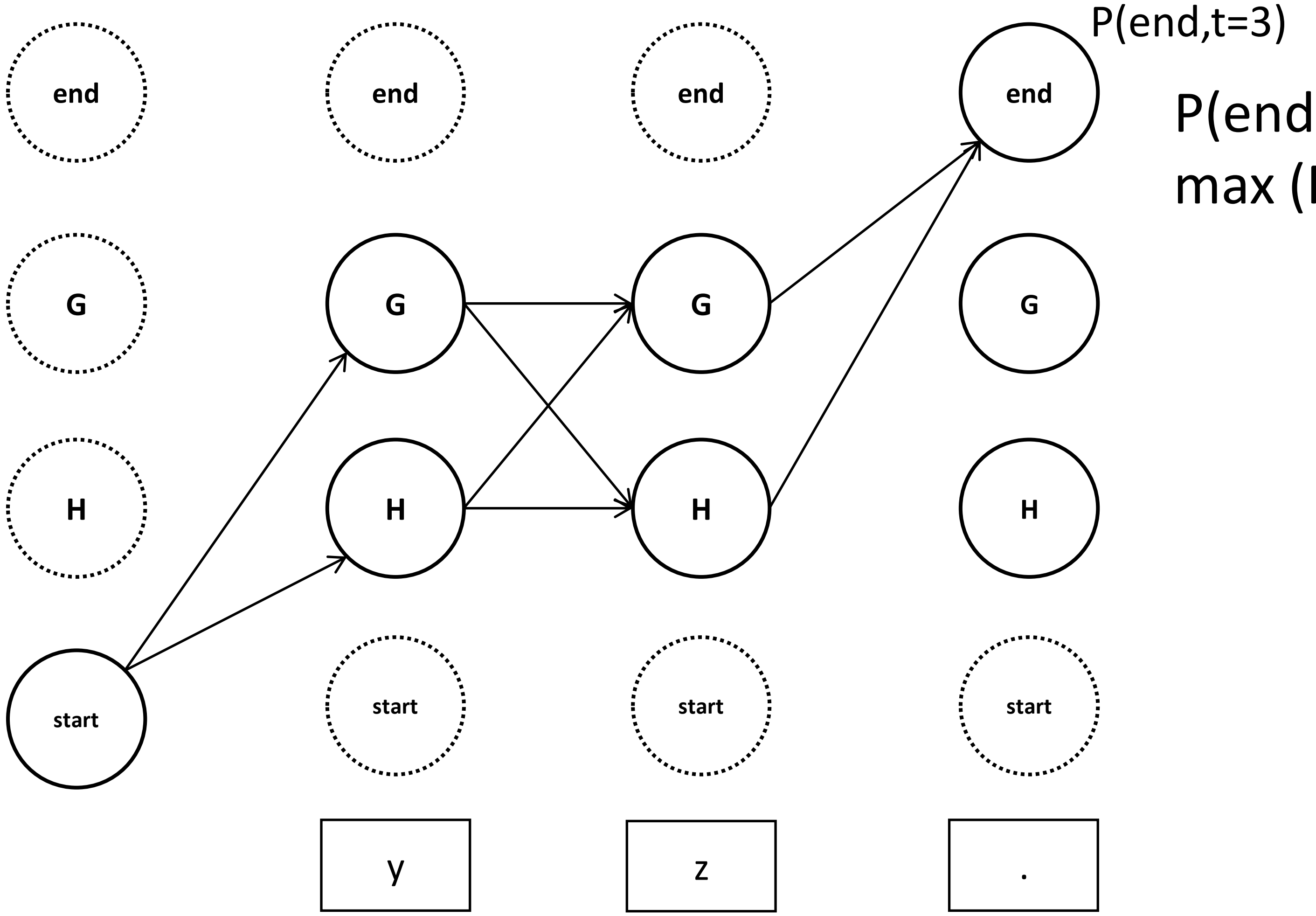
HMM Trellis



HMM Trellis

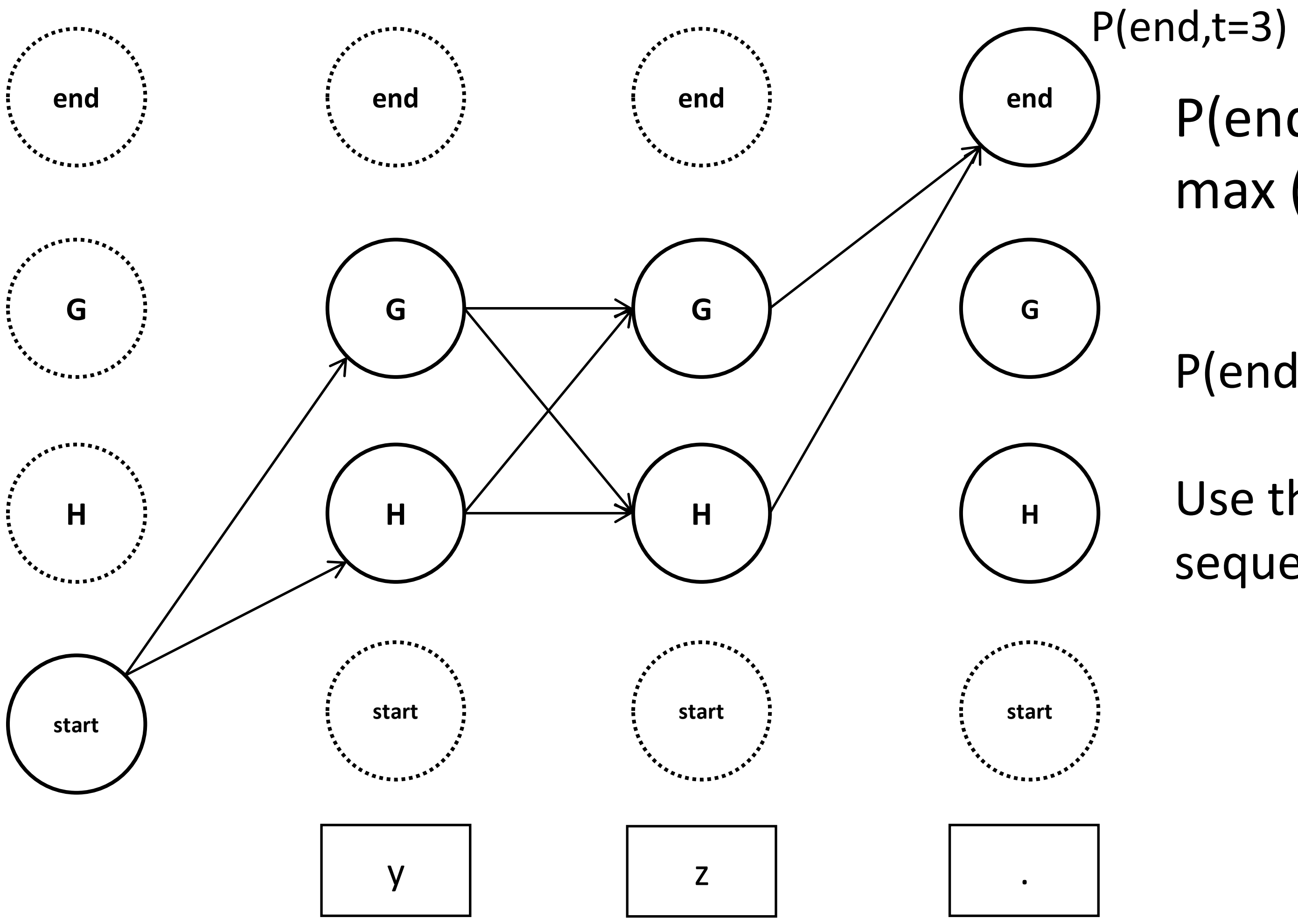


HMM Trellis



P(end,t=3) =
max (P(G,t=2) x P(end|G),
P(H,t=2) x P(end|H))

HMM Trellis



P(end,t=3) =
max (P(G,t=2) x P(end|G),
P(H,t=2) x P(end|H))

P(end,t=3) = best score for the sequence

Use the backpointers to find the sequence of states.

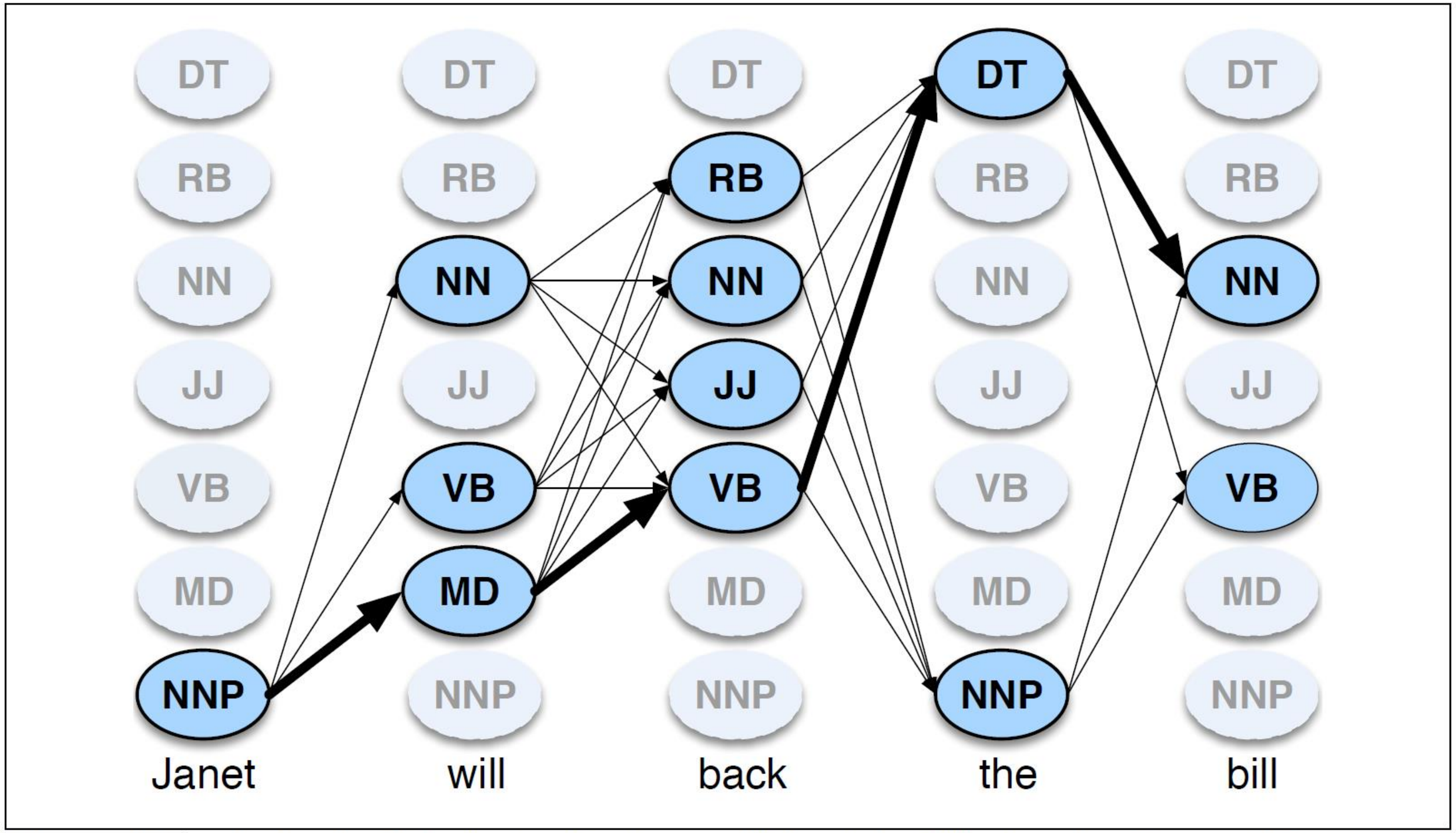


Figure 8.6 A sketch of the lattice for *Janet will back the bill*, showing the possible tags (q_i) for each word and highlighting the path corresponding to the correct tag sequence through the hidden states. States (parts of speech) which have a zero probability of generating a particular word according to the B matrix (such as the probability that a determiner DT will be realized as *Janet*) are greyed out.

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

	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

Figure 8.7 The A transition probabilities $P(t_i|t_{i-1})$ computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus $P(VB|MD)$ is 0.7968.

	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

Figure 8.8 Observation likelihoods B computed from the WSJ corpus without smoothing, simplified slightly.

Beam Search

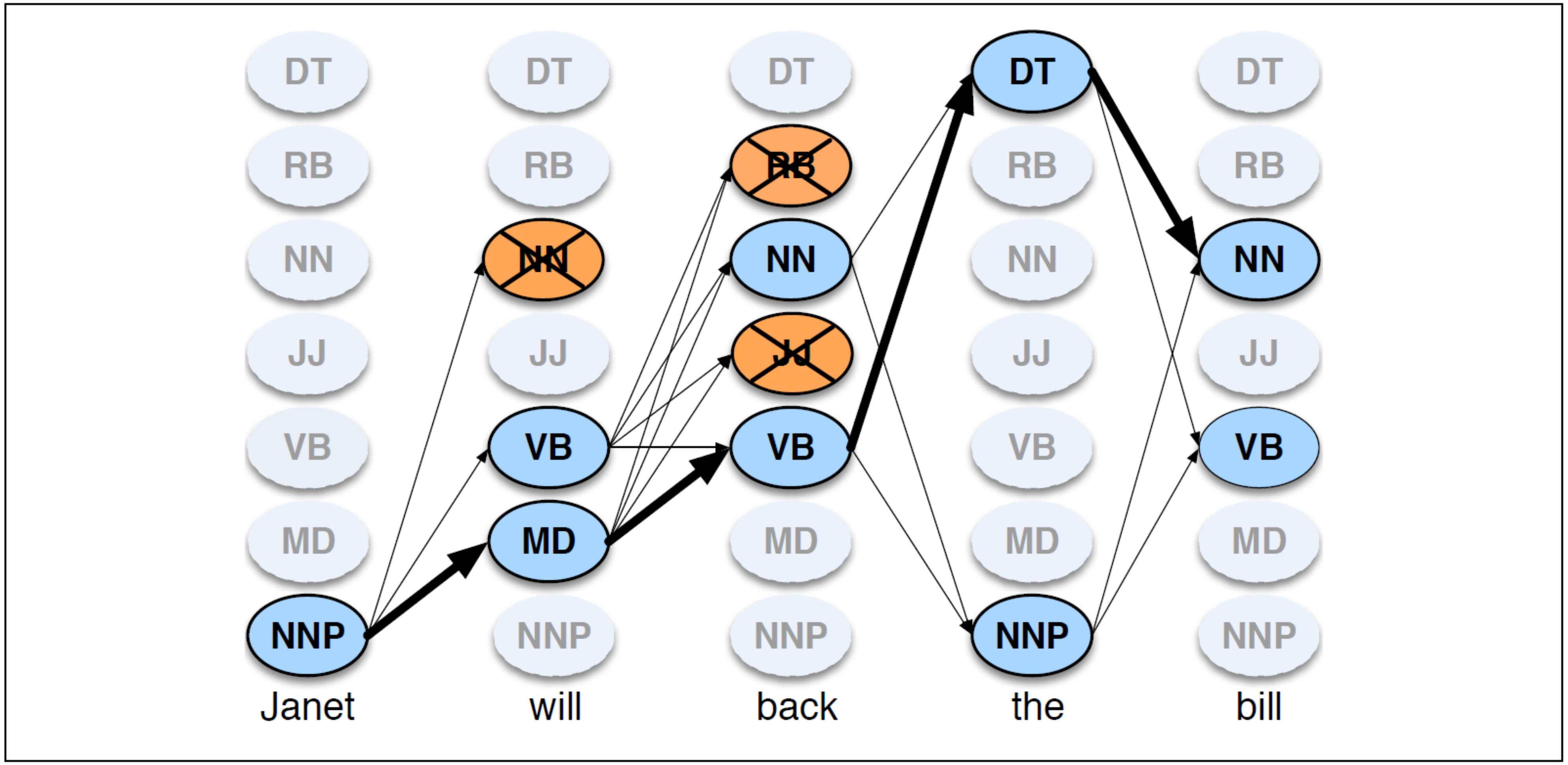


Figure 8.11 A beam search version of Fig. 8.6, showing a beam width of 2. At each time t, all (non-zero) states are computed, but then they are sorted and only the best 2 states are propagated forward and the rest are pruned, shown in orange.

Some Observations

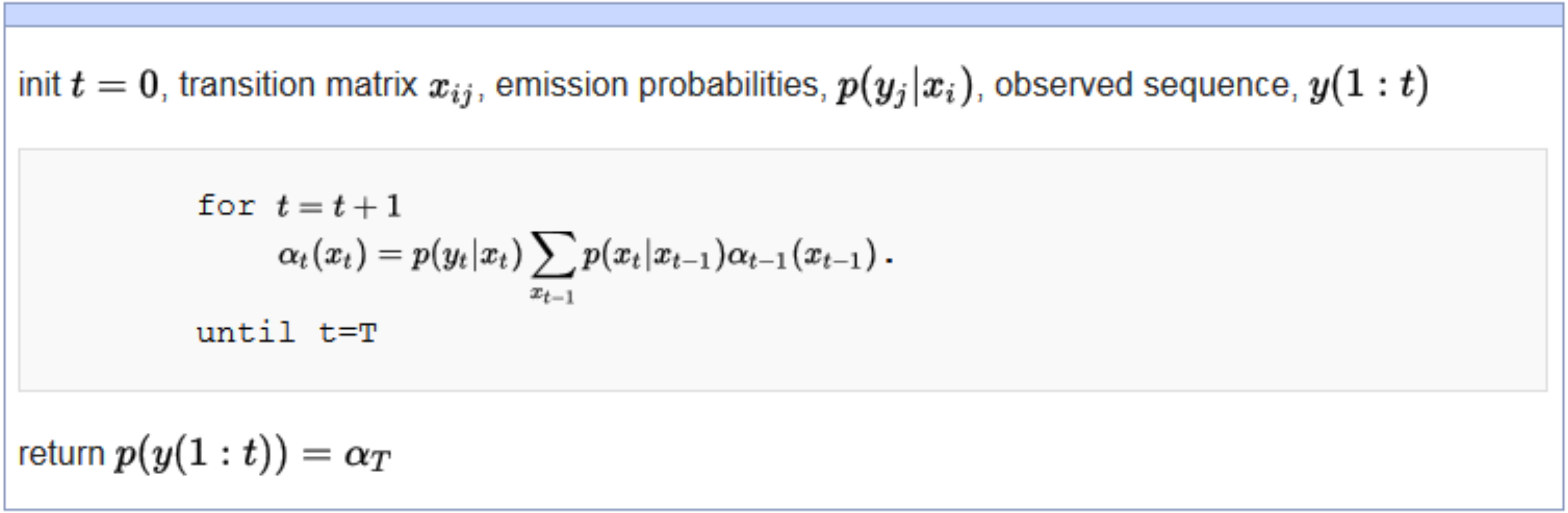
- Advantages of HMMs
    - Relatively high accuracy
    - Easy to train
- Higher-Order HMM
    - The previous example was about bigram HMMs
    - How can you modify it to work with trigrams?

How to compute P(O)

- Viterbi was used to find the most likely sequence of states that matches the observation
- What if we want to find all sequences that match the observation
- We can add their probabilities (because they are mutually exclusive) to form the probability of the observation
- This is done using the Forward Algorithm

The Forward Algorithm

- Used to compute the probability of a sequence
- Very similar to Viterbi
- Instead of *max* we use *sum*



```
init $t = 0$, transition matrix $x_{ij}$, emission probabilities, $p(y_j|x_i)$, observed sequence, $y(1:t)$

    for $t = t + 1$
        $\alpha_t(x_t) = p(y_t|x_t) \sum_{x_{t-1}} p(x_t|x_{t-1}) \alpha_{t-1}(x_{t-1})$.
    until t=T

return $p(y(1:t)) = \alpha_T$
```

Introduction to NLP

222.

Learning in Hidden Markov Models

HMM Learning

- Supervised
  - Training sequences are labeled
- Unsupervised
  - Training sequences are unlabeled
  - Known number of states
- Semi-supervised
  - Some training sequences are labeled

Supervised HMM Learning

- Estimate the static transition probabilities using MLE

$$a_{ij} = \frac{Count(q_t = s_i, q_{t+1} = s_j)}{Count(q_t = s_i)}$$

- Estimate the observation probabilities using MLE

$$b_j(k) = \frac{Count(q_i = s_j, o_i = v_k)}{Count(q_i = s_j)}$$

- Use smoothing

Unsupervised HMM Training

- Given:
  - observation sequences
- Goal:
  - build the HMM
- Use EM (Expectation Maximization) methods
  - forward-backward (Baum-Welch) algorithm
  - Baum-Welch finds an approximate solution for $P(O|\mu)$

Outline of Baum-Welch

- Algorithm
  - Randomly set the parameters of the HMM
  - Until the parameters converge repeat:
    - E step – determine the probability of the various state sequences for generating the observations
    - M step – reestimate the parameters based on these probabilities
- Notes
  - the algorithm guarantees that at each iteration the likelihood of the data $P(O|\mu)$ increases
  - it can be stopped at any point and give a partial solution
  - it converges to a local maximum

Introduction to NLP

231.

Statistical POS Tagging

The POS task

- Example
    - Bahrainis vote in second round of parliamentary election
- Jabberwocky (by Lewis Carroll, 1872)

`Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.

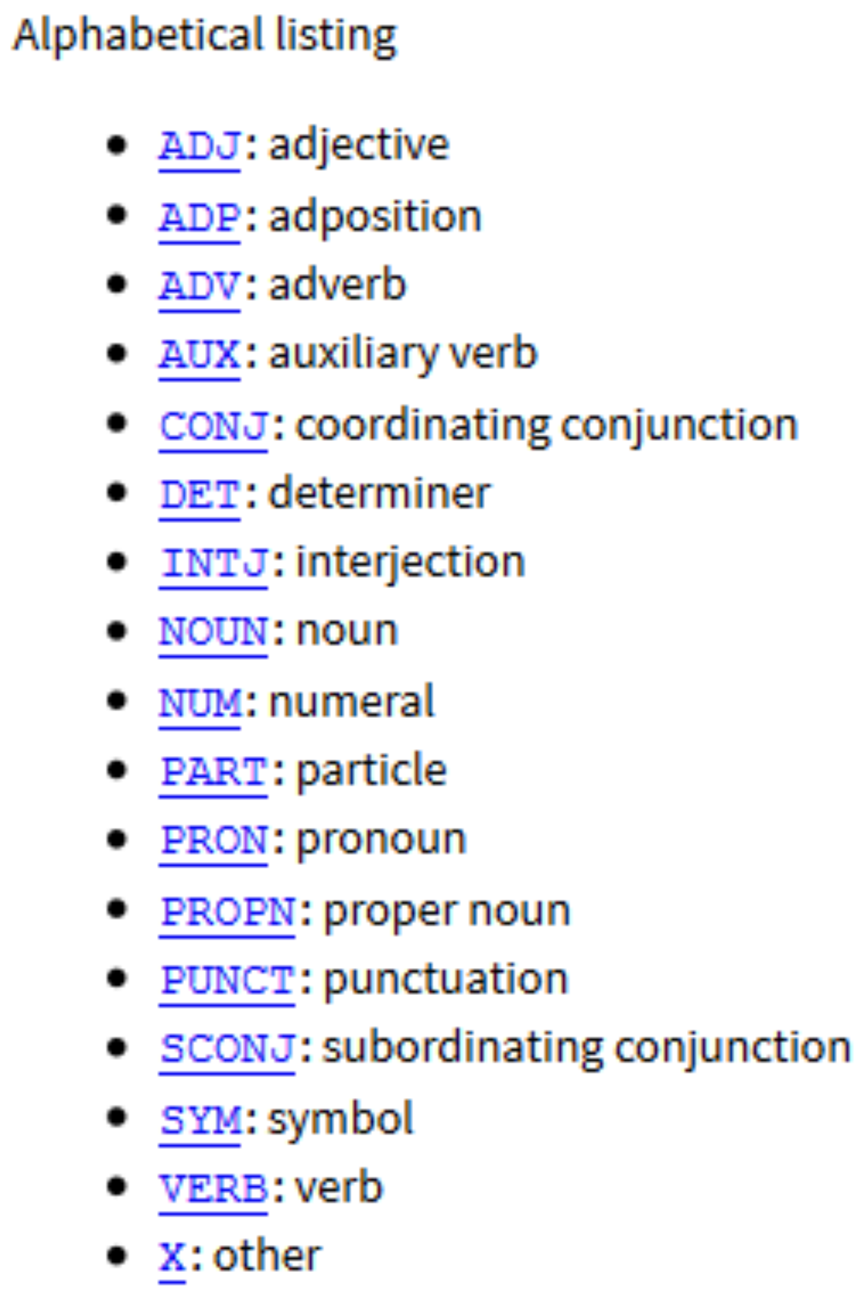
Penn Treebank tagset (1/2)

Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1
DT	determiner	the
EX	existential there	*there* is
FW	foreign word	d‘oeuvre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	green
JJR	adjective, comparative	greener
JJS	adjective, superlative	greenest
LS	list marker	1)
MD	modal	could, will
NN	noun, singular or mass	table
NNS	noun plural	tables
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	predeterminer	*both* the boys
POS	possessive ending	friend*'s*

Penn Treebank tagset (2/2)

Tag	**Description**	**Example**
PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	adverb	however, usually, naturally, here, good
RBR	adverb, comparative	better
RBS	adverb, superlative	best
RP	particle	give *up*
TO	to	*to* go, *to* him
UH	interjection	uhhuhhuhh
VB	verb, base form	take
VBD	verb, past tense	took
VBG	verb, gerund/present participle	taking
VBN	verb, past participle	taken
VBP	verb, sing. present, non-3d	take
VBZ	verb, 3rd person sing. present	takes
WDT	wh-determiner	which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	whose
WRB	wh-abverb	where, when

Universal POS

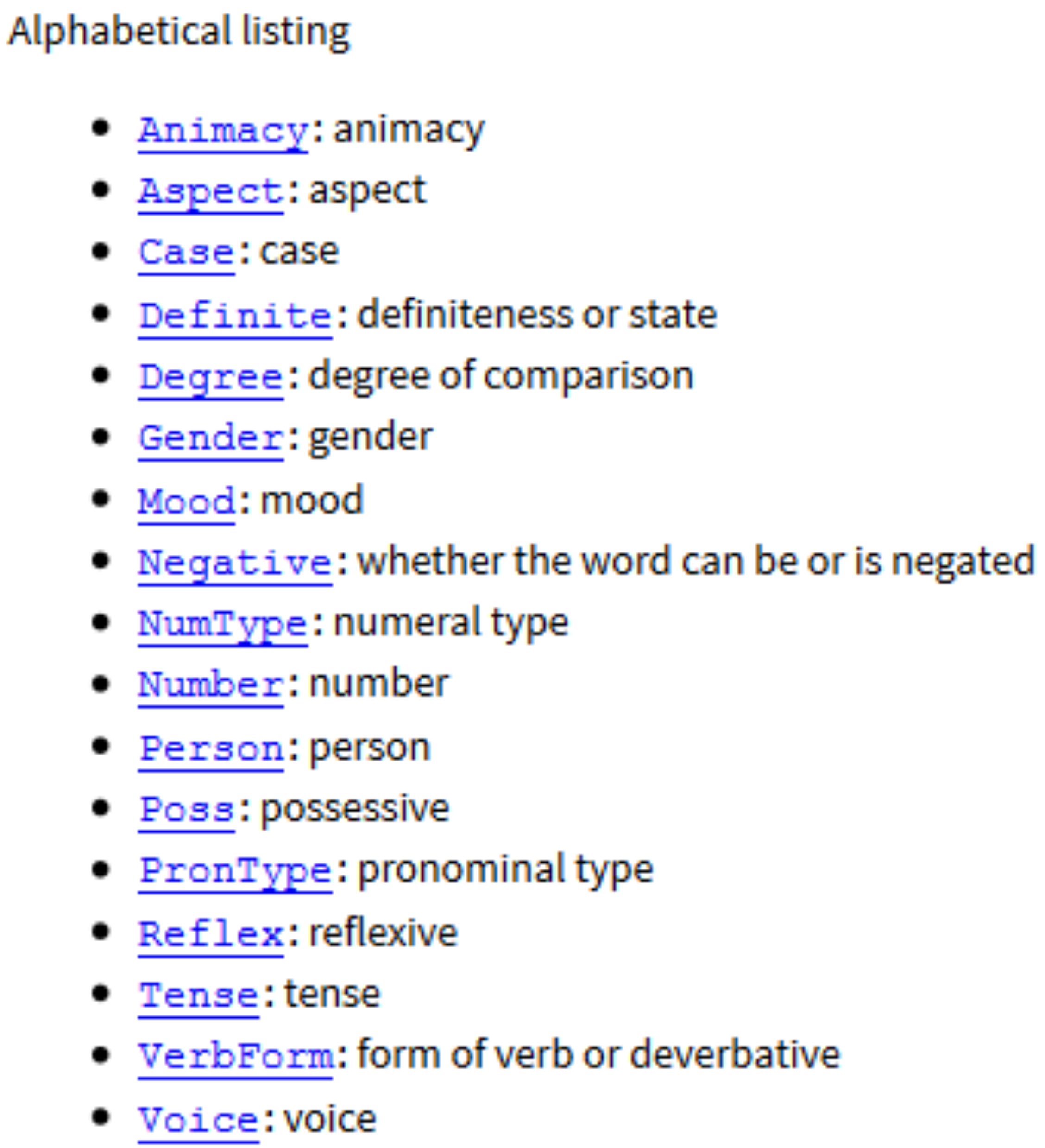


Alphabetical listing

- ADJ: adjective
- ADP: adposition
- ADV: adverb
- AUX: auxiliary verb
- CONJ: coordinating conjunction
- DET: determiner
- INTJ: interjection
- NOUN: noun
- NUM: numeral
- PART: particle
- PRON: pronoun
- PROPN: proper noun
- PUNCT: punctuation
- SCONJ: subordinating conjunction
- SYM: symbol
- VERB: verb
- X: other

ersaldependencies.org/u/pos/

Universal Features



Alphabetical listing

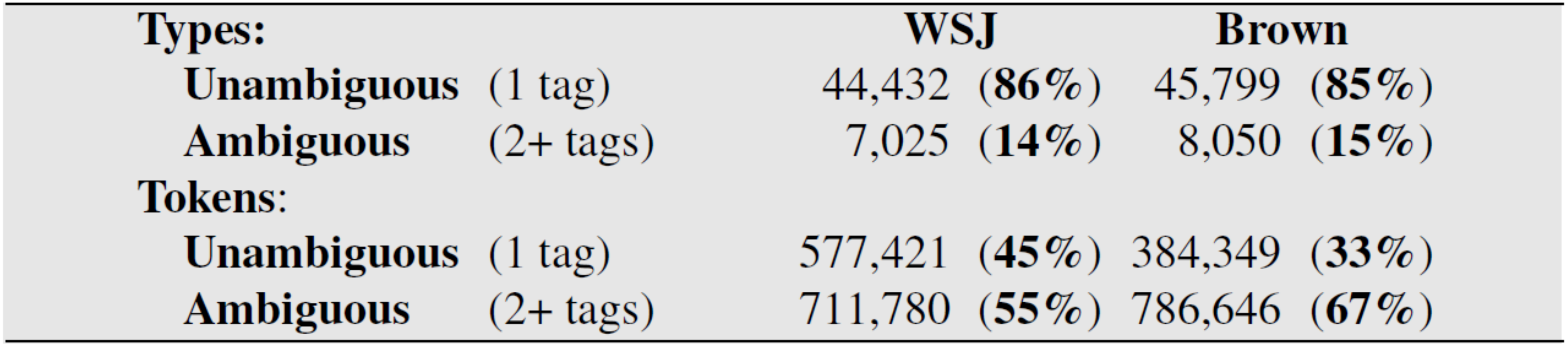
- Animacy: animacy
- Aspect: aspect
- Case: case
- Definite: definiteness or state
- Degree: degree of comparison
- Gender: gender
- Mood: mood
- Negative: whether the word can be or is negated
- NumType: numeral type
- Number: number
- Person: person
- Poss: possessive
- PronType: pronominal type
- Reflex: reflexive
- Tense: tense
- VerbForm: form of verb or deverbative
- Voice: voice

http://universaldependencies.org/u/feat/

Some Observations

- Ambiguity
  - count (noun) vs. count (verb)
  - 11% of all types but 40% of all tokens in the Brown corpus are ambiguous.
  - Examples
    - *like* can be tagged as ADP VERB ADJ ADV NOUN
    - *present* can be tagged as ADJ NOUN VERB ADV

POS Ambiguity



Types:		**WSJ**		**Brown**	
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%**)

Figure 10.2 The amount of tag ambiguity for word types in the Brown and WSJ corpora, from the Treebank-3 (45-tag) tagging. These statistics include punctuation as words, and assume words are kept in their original case.

Example from J&M

Example

- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/NNS**
- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/VBZ**

Classifier-based POS Tagging

- A baseline method would be to use a classifier to map each individual word into a likely POS tag
    - Why is this method unlikely to work well?

Sources of Information

- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/NNS**
- Bethlehem/NNP Steel/NNP Corp./NNP ,/, hammered/VBN by/IN higher/JJR **costs/VBZ**

- Knowledge about individual words
  - lexical information
  - spelling (-or)
  - capitalization (IBM)
- Knowledge about neighboring words

Evaluation

- Baseline
  - tag each word with its most likely tag
  - tag each OOV word as a noun.
  - around 90%
- Current accuracy
  - around 97% for English
  - compared to 98% human performance

HMM Tagging

- T = argmax P(T|W)
  - where $T=t_1,t_2,\ldots,t_n$
- By Bayes' theorem
  - P(T|W) = P(T)P(W|T)/P(W)
- Thus we are attempting to choose the sequence of tags that maximizes the RHS of the equation
  - P(W) can be ignored
  - P(T) is called the prior, P(W|T) is called the likelihood.

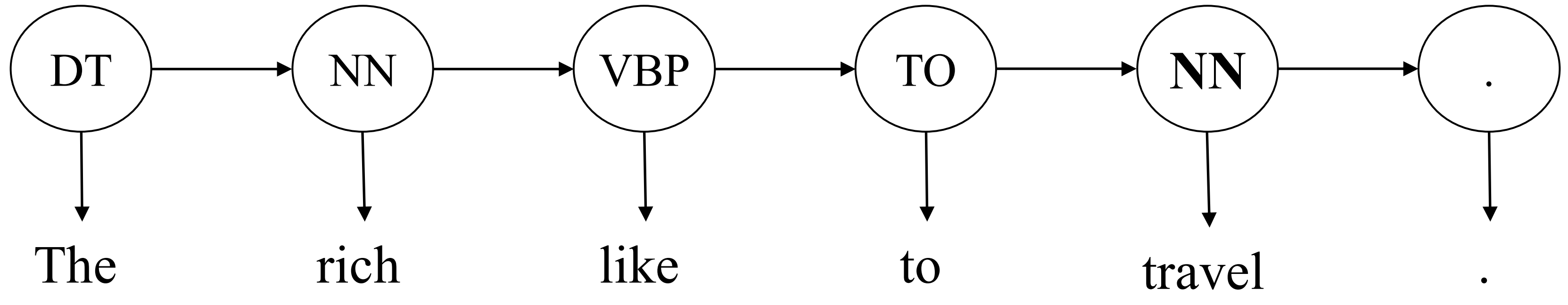
HMM Tagging

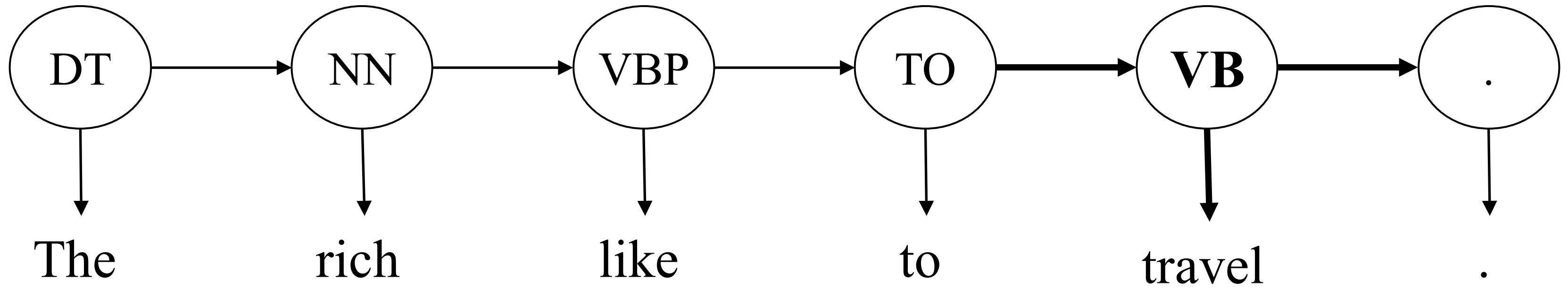
- Complete formula
  - $P(T)P(W|T) = \prod P(w_i|w_1t_1\ldots w_{i-1}t_{i-1}t_i)P(t_i|t_1\ldots t_{i-2}t_{i-1})$
- Simplification 1:
  - $P(W|T) = \prod P(w_i|t_i)$
- Simplification 2:
  - $P(T)= \prod P(t_i|t_{i-1})$
- Bigram approximation
  - $T = \text{argmax } P(T|W) = \text{argmax} \prod P(w_i|t_i)\ P(t_i|t_{i-1})$

Example

- The/DT rich/JJ like/VBP to/TO travel/VB ./.

Example


DT
NN
VBP
TO
NN
.
The
rich
like
to
travel
.


DT
NN
VBP
TO
VB
.
The
rich
like
to
travel
.

Maximum Likelihood Estimates

- Transition probabilities

  P(NN|JJ) = C(JJ,NN)/C(JJ) = 22301/89401 = .249

- Emission probabilities

  P(this|DT) = C(DT,this)/C(DT) = 7037/103687 = .068

Evaluating Taggers

- Data set
  - Training set
  - Development set
  - Test set
- Tagging accuracy
  - how many tags right

HMM POS Results

- Assigning each word its most likely tag: 90%
- Trigram HMM: 95% (55% on unknown words)
- Tuned HMM (Brants 1998): 96.2% (86.0%)
- SOTA (Bi-LSTM CRF): 97.5% (89+%)

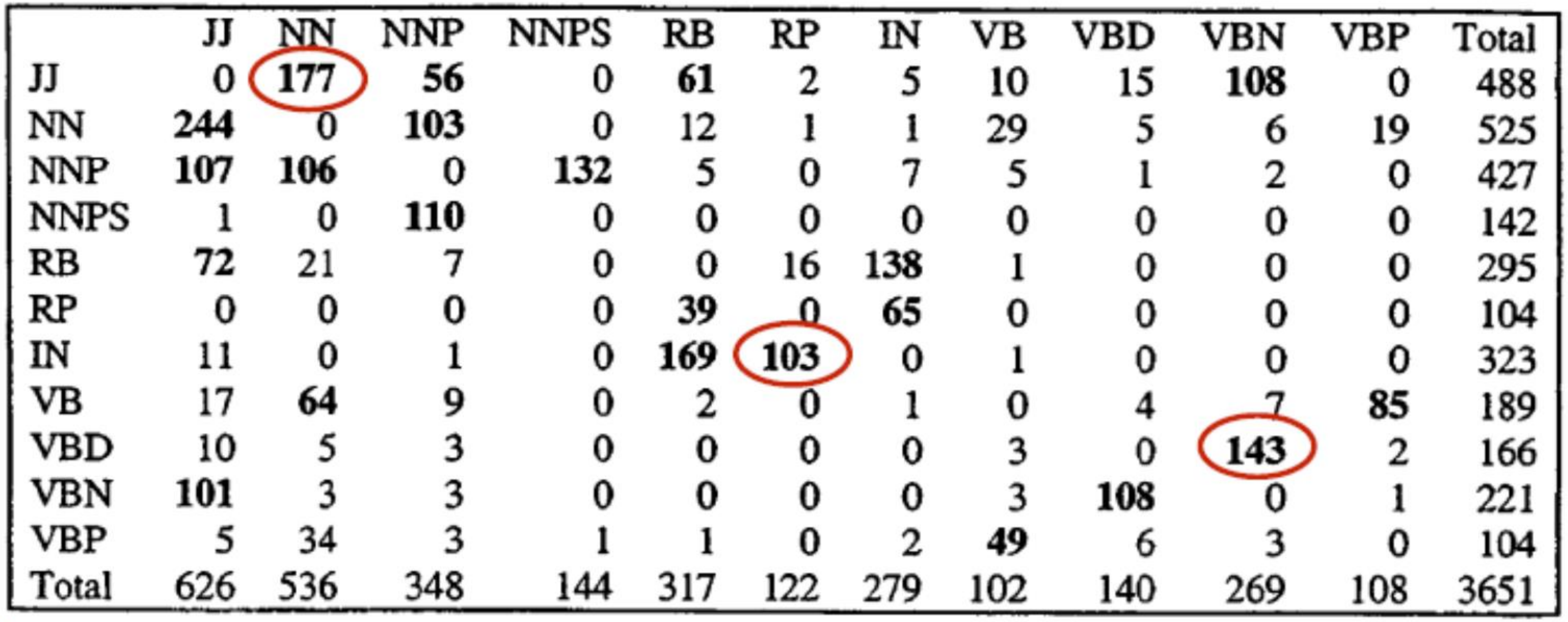
Numbers thanks to Dan Klein and Greg Durrett

Remaining Errors

- Words not seen with that tag in training: 4.5%
- Unknown word: 4.5%
- Could get right: 16% (needs parsing)
- Difficult decision: 20% (“set” = VBP or VBD?)
- Underspecified/unclear, gold standard inconsistent/wrong: 58% (e.g., is “discontinued” JJ or VBN)

[Manning 2011]

Confusion Matrix



	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

JJ/NN	NN			
official	knowledge			

VBD	RP/IN	DT	NN
made	up	the	story

RB	VBD/VBN	NNS
recently	sold	shares

[Example from Toutanova+Manning'00 via Dan Klein]

Notes on POS

- New domains
  - Lower performance
- New languages
  - Morphology matters! Also availability of training data
- Distributional clustering
  - Combine statistics about semantically related words
  - Example: names of companies
  - Example: days of the week
  - Example: animals

Brown Clustering

- Words with similar vector representations (embeddings) are clustered together, in an agglomerative (recursive) way
- For example, “Monday”, “Tuesday”, etc. may form a new vector “Day of the week”
- Published by Brown et al. [1992]

Example

Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays
people guys folks fellows CEOs chaps doubters commies unfortunates blokes
down backwards ashore sideways southward northward overboard aloft downwards adrift
water gas coal liquid acid sand carbon steam shale iron
great big vast sudden mere sheer gigantic lifelong scant colossal
American Indian European Japanese German African Catholic Israeli Italian Arab
mother wife father son husband brother daughter sister boss uncle
machine device controller processor CPU printer spindle subsystem compiler plotter
John George James Bob Robert Paul William Jim David Mike
feet miles pounds degrees inches barrels tons acres meters bytes
had hadn't hath would've could've should've must've might've
that tha theat
head body hands eyes voice arm seat eye hair mouth

Example

- Input:
  - this is one document . it has two sentences but the program only cares about spaces .
  - here is another document . it also has two sentences .
  - and here is a third document with one sentence .
  - this document is short .
  - the dog ran in the park .
  - the cat was chased by the dog .
  - the dog chased the cat .

[code by Michael Heilman: https://github.com/mheilman/tan-clustering]

```
.                  1011               9
the                011                7
is                 110                4
document           1110               4
dog                000                3
it                 101001             2
one                11111              2
sentences          1010111            2
chased             00111              2
two                1010100            2
has                1010110            2
here               111101             2
this               1000               2
cat                0010               2
and                11110010           1
sentence           11110011           1
ran                01011              1
in                 0100               1
spaces             10101011011        1
another            1010001            1
cares              101010111          1
also               1010000            1
only               10101011010        1
program            10101011001        1
was                001100             1
park               01010              1
but                10101011000        1
short              1001               1
with               111100001          1
by                 001101             1
a                  111100000          1
about              10101010           1
third              11110001           1
```

[code by Michael Heilman]

Notes on POS

- British National Corpus
  - http://www.natcorp.ox.ac.uk/
- Tagset sizes
  - PTB 45, Brown 85, Universal 12, Twitter 25
- Dealing with unknown words
  - Look at features like twoDigitNum, allCaps, initCaps, containsDigitAndSlash (Bikel et al. 1999)

HMM Spreadsheet

- Jason Eisner’s awesome interactive spreadsheet about learning HMMs
  - http://cs.jhu.edu/~jason/papers/#eisner-2002-tnlp
  - http://cs.jhu.edu/~jason/papers/eisner.hmm.xls