Part-of-Speech Tagging

3007/7059 Artificial Intelligence

Slides by Wei Zhang

School of Computer Science

University of Adelaide

Outline

- Part-of-Speech Tagging (POS)
- POS via Hidden Markov Model

Part-of-Speech Tagging

- Part-of-speech tagging (POS) is the process of assigning a part-of-speech tag to each tagging word in an input text.
  - The input to a tagging algorithm is a sequence of (tokenized) words and a tagset, and the output is a sequence of tags, one per token.

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	: ; ... – -

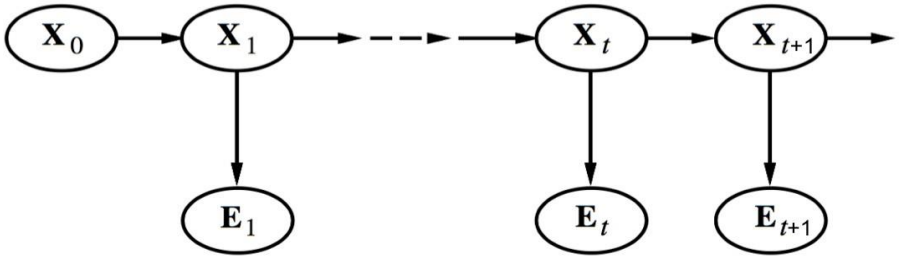
Penn Treebank tagset

Part-of-Speech Tagging

- POS tags have been used for a variety of NLP tasks and are extremely useful since they provide linguistic signal on how a word is being used within the scope of a phrase, sentence, or document.
- POS tagging is not a mandatory step in all NLP tasks.
- It can reduce ambiguity, but cannot eliminate ambiguity.
- It is helpful on dealing with unlabelled or unseen datasets.

POS via HMM

- Hidden Markov Model:
  - An HMM is a probabilistic sequence model: given a sequence of units (words, letters, morphemes, sentences, whatever), it computes a probability distribution over possible sequences of labels and chooses the best label sequence.



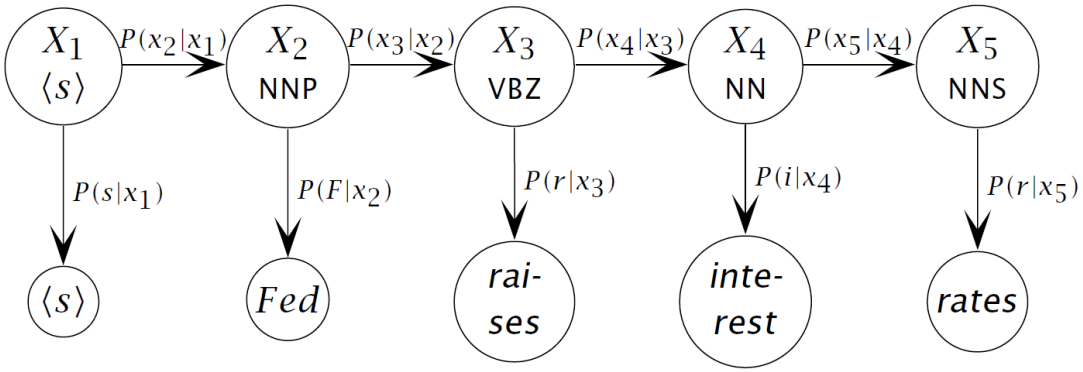
- Two models in HMM:
  - Transition model $P(\mathbf{X}_{t+1}|\mathbf{X}_t)$

    First order Markov assumption: the present state depends only on the immediate previous state.
  - Observation (sensor) model $P(\mathbf{E}_{t+1}|\mathbf{X}_{t+1})$

    Sensor Markov assumption: the probability of observing $\mathbf{E}_t$ depends only on the state $\mathbf{X}_t$.

POS via HMM

- In HMM, words that we see in the input are observed events E_t and POS tags are hidden states X_t



- We normally do supervised training, and then (Bayesian network style) inference to decide POS tags
  - HMM is trained on a fully labelled dataset- a set of sentences with each word annotated with a POS tag-setting parameters by maximum likelihood estimates on this training data.
  - Decoding: the task of determining the hidden variables sequence corresponding to the sequence of observations.
    - Viterbi algorithm for decoding.

POS via HMM - Training

Training:
Input: A tagged corpus, a tagset
Output: transition model and observation model

- E.g., WSJ corpus: contains a million words published in the Switchboard Wall Street Journal in 1989.
- Penn Treebank tagset includes 45 tags.

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	: ; ... – -

POS via HMM - Training

Training:
Input: A tagged corpus, a tagset
Output: transition model and observation model
Notation: W (word in sentence) as observation variables, T (tags) as state variables.

- Transition model:

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

We compute the maximum likelihood estimate of this transition probability by counting, out of the times we see the first tag in a labeled corpus, how often the first tag is followed by the second.

e.g., in the WSJ corpus, MD (modal) occurs 13124 times, of which it is followed by VB (verb base form) 10471, for an MLE estimate of:

$$P(VB|MD) = \frac{C(MD, VB)}{C(MD)} = \frac{10471}{13124} = .80$$

POS via HMM - Training

Training:
Input: A tagged corpus, a tagset
Output: transition model and observation model
Notation: W (word in sentence) as observation variables, T (tags) as state variables.

- Observation model:

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

e.g., in the WSJ corpus, 13124 occurrences of MD in the WSJ corpus, it is associated with will 4046 times:

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

POS via HMM - Decoding

Input: transition model, observation model, sentences to be tagged (the observations).
Output: a set of sequences of tags.
Task: the goal of HMM decoding is to choose the tag sequence t_1^n that is most probable given the observation sequence of *n* words w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P\left(t_1^n | w_1^n\right)$$

By using Bayes rule:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)}{P\left(w_1^n\right)}$$

Remove the denominator :

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)$$

POS via HMM - Decoding

Input: transition model, observation model, sentences to be tagged (the observations).

Output: a set of sequences of tags.

Task: the goal of HMM decoding is to choose the tag sequence t_1^n that is most probable given the observation sequence of *n* words w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)$$

As assuming the probability of a word appearing depends only on its own tag and is independent of neighboring words and tags:

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

POS via HMM - Decoding

Input: transition model, observation model, sentences to be tagged (the observations).
Output: a set of sequences of tags.
Task: the goal of HMM decoding is to choose the tag sequence t_1^n that is most probable given the observation sequence of *n* words w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P\left(w_1^n | t_1^n\right) P\left(t_1^n\right)$$

Assume we target bigram POS tagging, is that the probability of a tag is dependent only on the previous tag, rather than the entire tag sequence **:**

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

POS via HMM - Decoding

Input: transition model, observation model, sentences to be tagged (the observations).

Output: a set of sequences of tags.

Task: the goal of HMM decoding is to choose the tag sequence t_1^n that is most probable given the observation sequence of *n* words w_1^n:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} \underbrace{P(w_i|t_i)}_{\text{Observation model}} \underbrace{P(t_i|t_{i-1})}_{\text{Transition model}}$$

Problem: For an HMM with *N* hidden states and an observation sequence of *T* observations, there are N^T possible hidden sequences. For real tasks, where N and T are both large, it is not possible to compute.

POS via HMM - Decoding

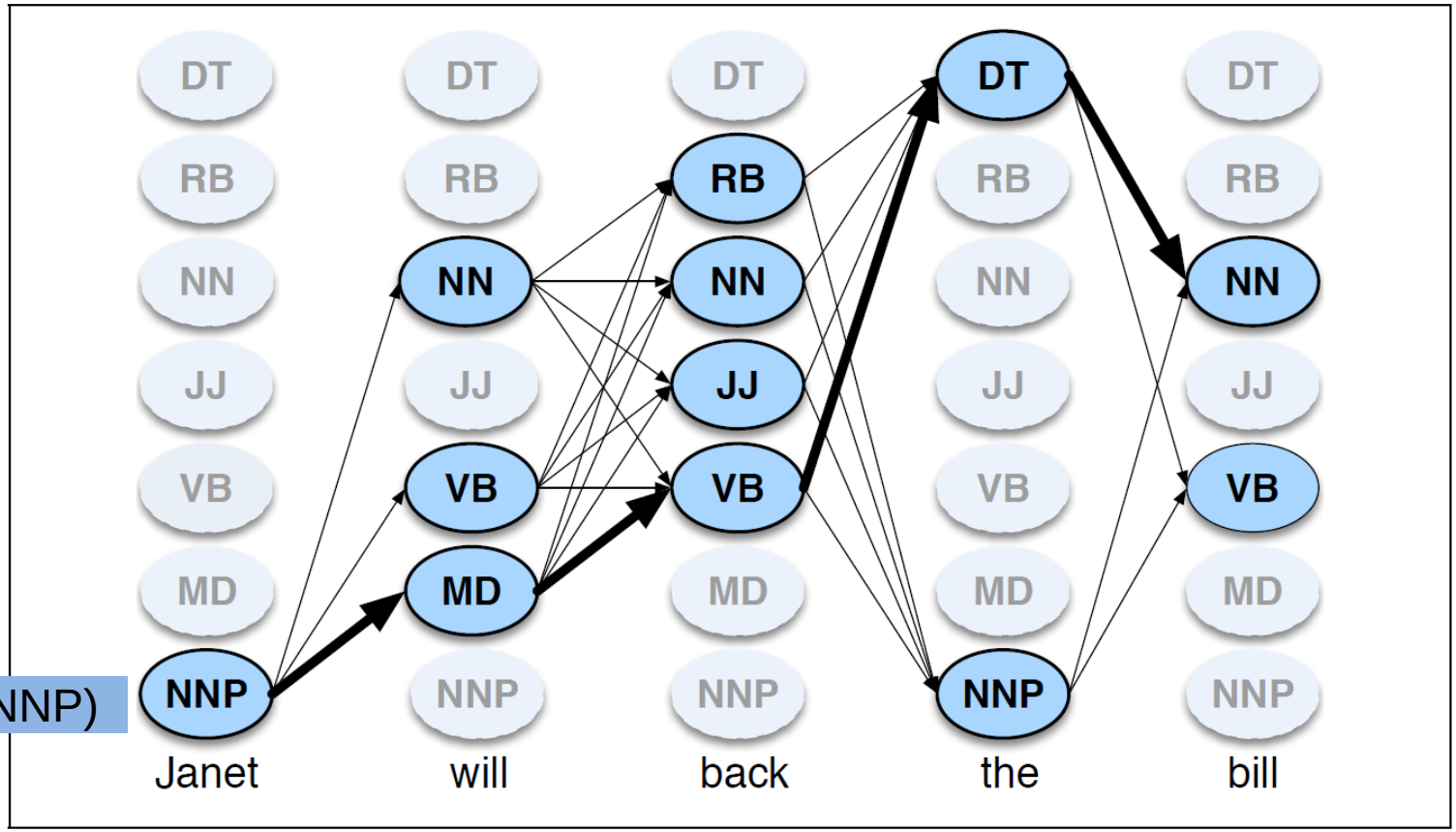
- Dynamic programming: simplifies a complicated problem by breaking it down into simpler sub-problems in a recursive manner.

- Viterbi algorithm
  - The Viterbi algorithm first sets up a probability matrix, with columns represent observation and rows are for each state.
  - Then recursively update the matrix.
  - Backtrace the most probable path.

POS via HMM - Decoding

States

P(janet|NNP)



Observations

POS via HMM - Decoding

- Viterbi algorithm

Let $v_t(j)$ represents the probability that the HMM is in state *j* after seeing the first *t* observations and passing through **the most probable state sequence** $x_1 \ldots x_{t-1}$ given the HMM's observation and transition models, we compute the Viterbi probability by taking **the most probable of the extensions of the paths that lead to the current cell.**

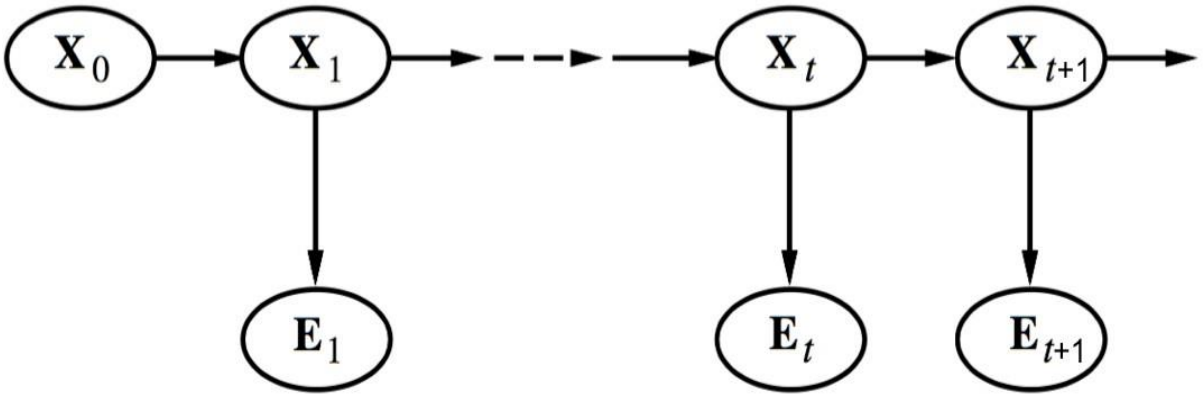
For a given state x_j at time t, n is the number of states:

$$v_t(j) = \max_{i \in [1,n]} v_{t-1}(i) P(x_j|x_i) P(e_t|x_j)$$

POS via HMM - Decoding

- Viterbi algorithm

$$v_t(j) = \max_{i \in [1,n]} v_{t-1}(i) P(x_j|x_i) P(e_t|x_j)$$



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

POS via HMM – decoding example

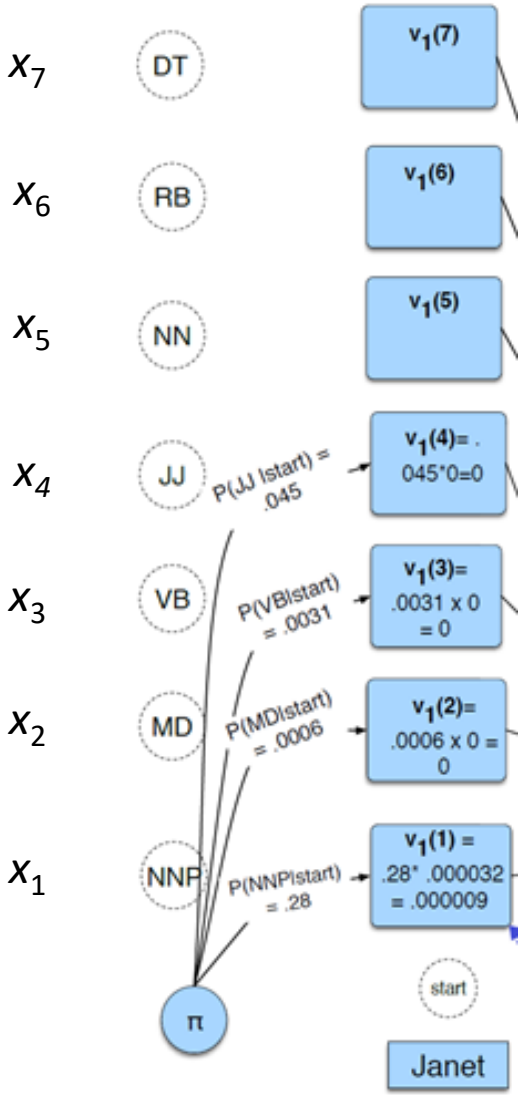
- The transition probabilities computed from the WSJ corpus without smoothing. Rows are labeled with the conditioning event; thus *P(VB|MD)* is 0.7968.

	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

- The Observation likelihoods computed from the WSJ corpus without smoothing.

	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

POS via HMM – decoding example



$$v_t(j) = \max_{i \in [1,n]} v_{t-1}(i) P(x_j|x_i) P(e_t|x_j)$$

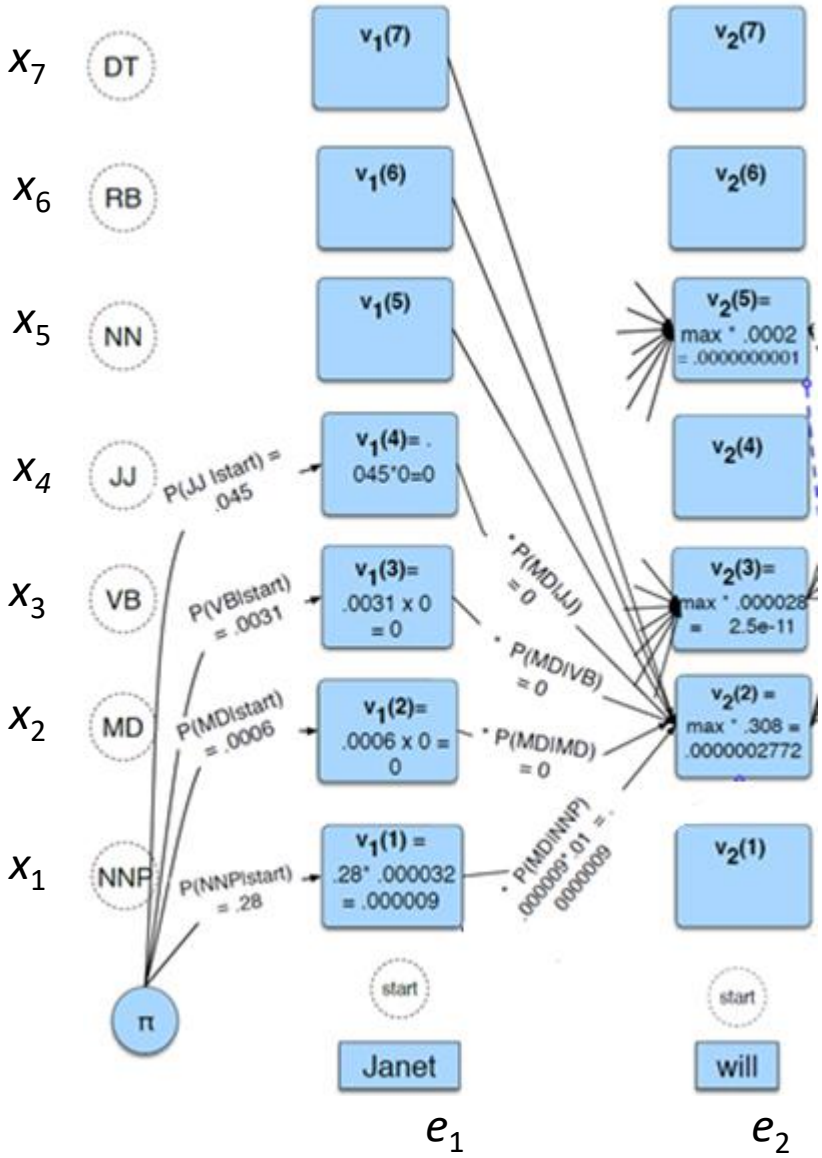
$V_1(j) = P(x_j|\text{start}) P(\text{janet}|x_j)$

$V_1(1)$= P(NNP|start)P(janet|NNP)
=0.2767*0.000032=0.000009

$V_1(2)$= P(MD|start)P(janet|MD)
=0.0006* 0 =0

$V_1(3)$= P(VB|start)P(janet|DT) =0
$V_1(4)$= P(JJ|start)P(janet|JJ) =0
$V_1(5)$= P(NN|start)P(janet|NN) =0
$V_1(6)$= P(RB|start)P(janet|RB) =0
$V_1(7)$= P(DT|start)P(janet|DT) =0

POS via HMM – decoding example



$$v_t(j) = \max_{i \in [1,n]} v_{t-1}(i) P(x_j|x_i) P(e_t|x_j)$$

$V_2(j)$ =max $V_1(i)$ $P(x_j|x_i)P(will|x_j)$

…

When j = 2, MD:

$V_1(1)$ *P(MD|NNP)P(will|MD)

$V_1(2)$ *P(MD|MD)P(will|MD)

$V_1(3)$ *P(MD|VB)P(will|MD)

$V_1(4)$ *P(MD|JJ)P(will|MD)

$V_1(5)$ *P(MD|NN)P(will|MD)

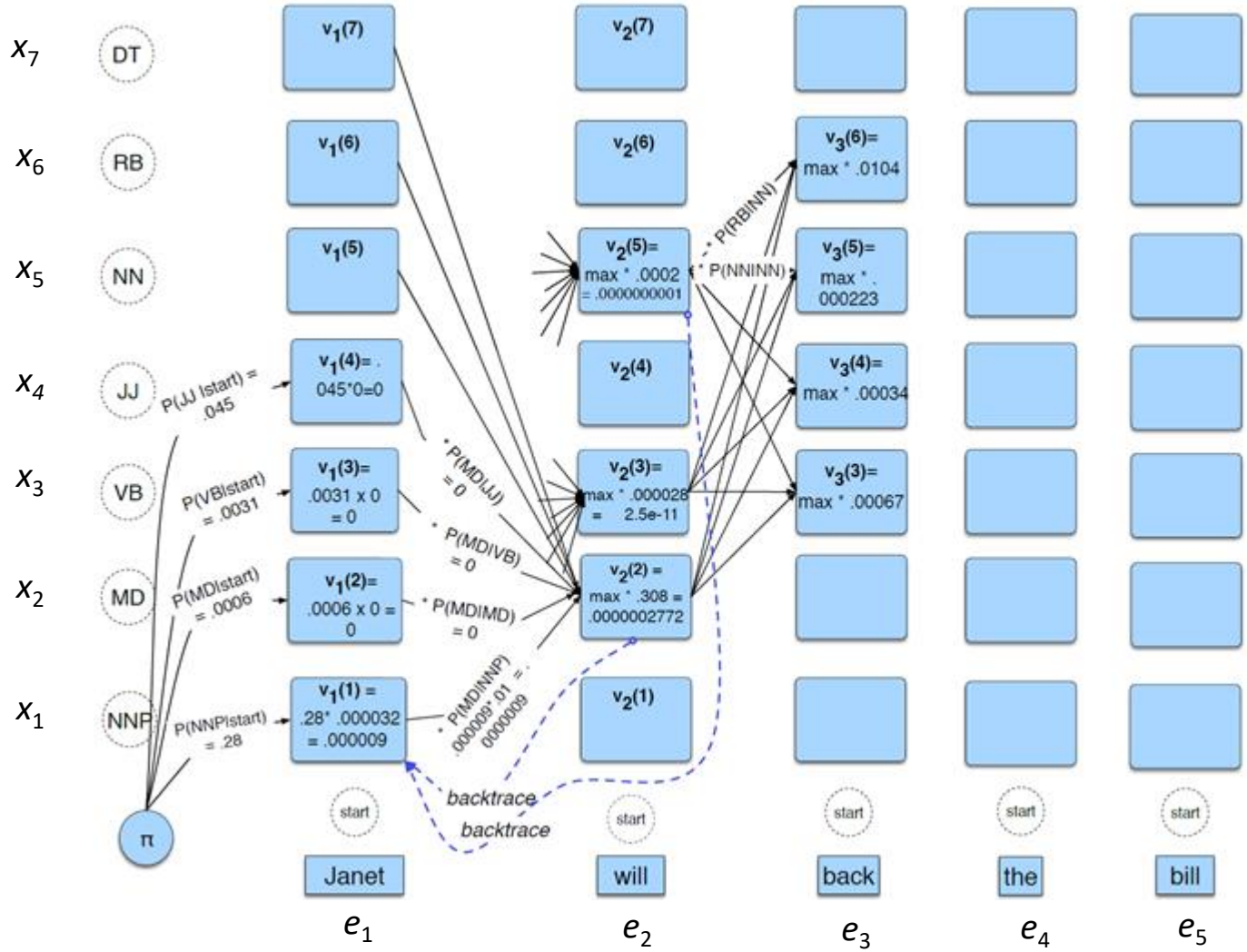
$V_1(6)$ *P(MD|RB)P(will|MD)

$V_1(7)$ *P(MD|DT)P(will|MD)

max

$V_2(2)$ = max = 0.000009 * 0.308431
=0.0000002772

POS via HMM – decoding example



After the cells are filled in, backtracing from the end state, we should be able to reconstruct the correct state sequence.

Viterbi Backtracing

- Record backpointers when performing forward computation.
- By keeping track of the path of hidden states that led to each state, Viterbi then at the end backtracing the best path to the beginning.

Reference

Book:

Speech and Language Processing Dan Jurafsky and James H. Martin.

https://web.stanford.edu/~jurafsky/slp3/