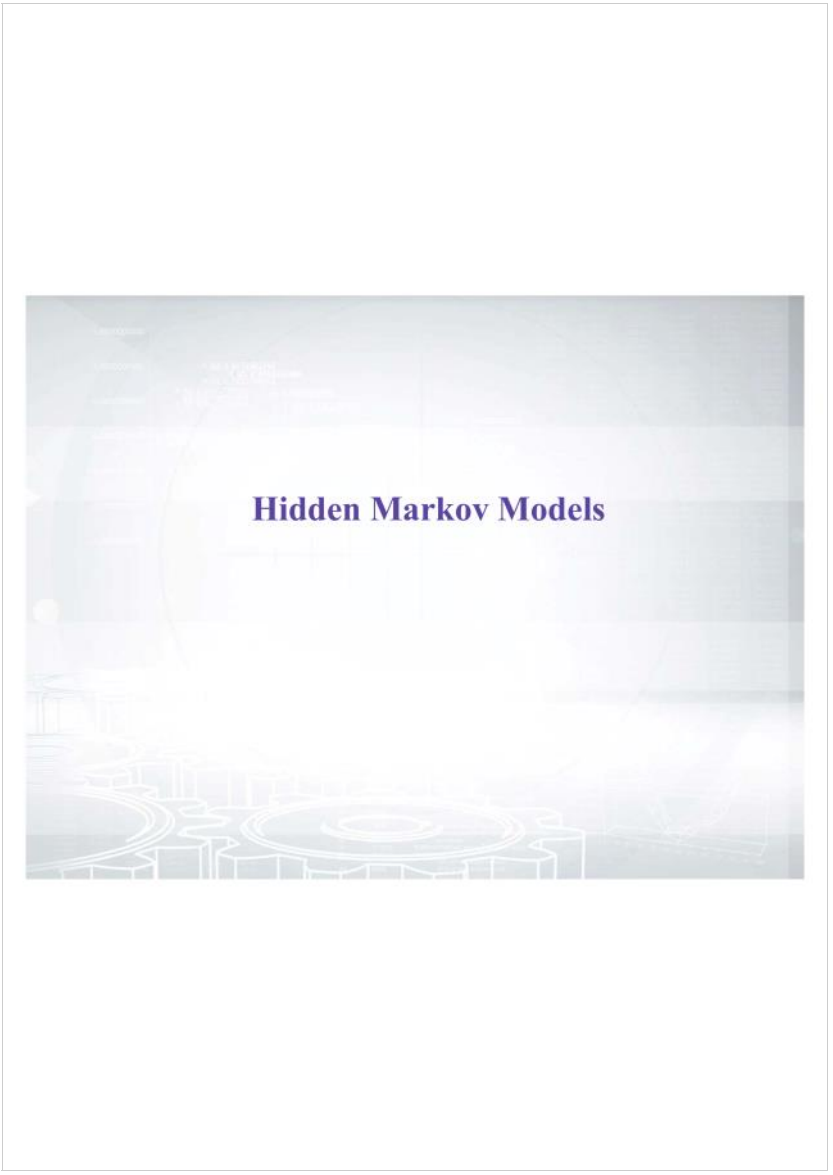
Sequence tagging

Friday, June 15, 2018 3:39 PM

hmm



Sequence labeling

Problem

- Given a sequence of *tokens*, infer the most probable sequence of *labels* for these tokens.

Examples

- part of speech tagging
- named entity recognition

Example: part-of-speech tagging

Tokens are words, and labels are PoS tags.

PRON	VERB	DET	PROPN	NOUN
I	saw	a	Heffalump	today.

PoS tags from Universal Dependencies project

Open class words	
ADJ	adjective
ADV	adverb
INTJ	interjection
NOUN	noun
PROPN	proper noun
VERB	verb

Other	
PUNCT	punctuation
SYM	symbol
X	other

Closed class words	
ADP	adposition
AUX	auxiliary verb
CCONJ	coordinating conjunction
DET	determiner
NUM	numeral
PART	particle
PRON	pronoun
SCONJ	subordinating conjunction

http://universaldependencies.org/

Example: named entity recognition

Once upon a time, a very long time ago now, about **[last Friday]**, **[Winnie-the-Pooh]** lived in a forest all by himself under the name of **[Sanders]**.

Types of named entities

Any real-world object which may have a proper name:

- persons
- organizations
- locations
- …

Also named entities usually include:

- dates and times
- units
- amounts

Approaches to sequence labeling

1. Rule-based models (example: EngCG tagger)
2. Separate label classifiers for each token
3. Sequence models (HMM, MEMM, CRF)
4. Neural networks

PoS tagging with HMMs

Notation:

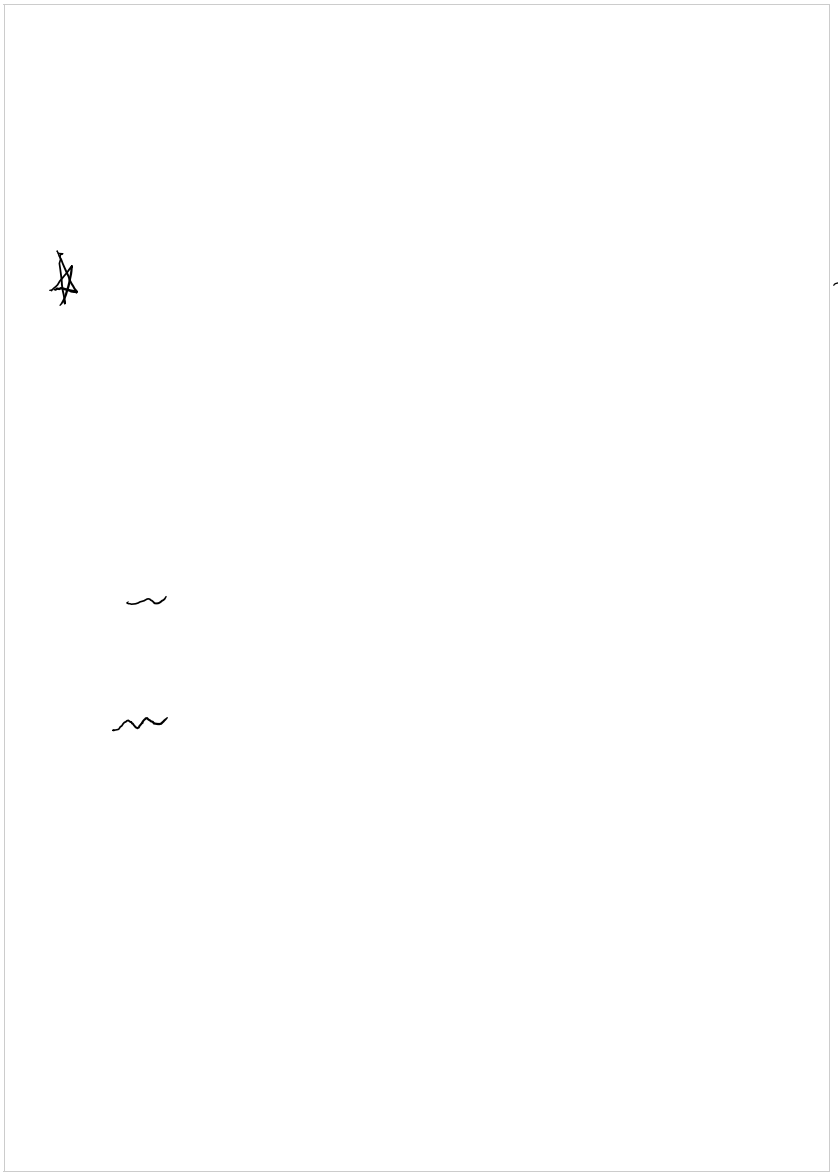
$\mathbf{x} = x_1, \ldots x_T$ is a sequence of words (input)

$\mathbf{y} = y_1, \ldots y_T$ is a sequence of their tags (labels)

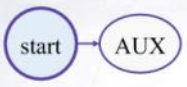
We need to find the most probable sequence of tags given the sentence:

$$\mathbf{y} = \text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}}\, p(\mathbf{x}, \mathbf{y})$$

But first, let us define the model…

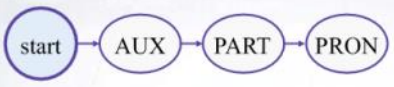


Text generation in HMM: an example



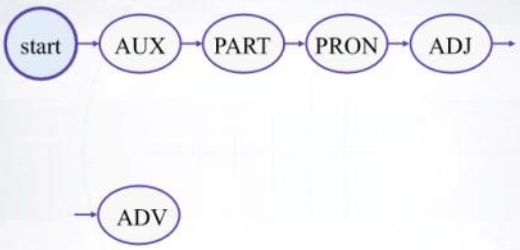
Text generation in HMM: an example

Text generation in HMM: an example

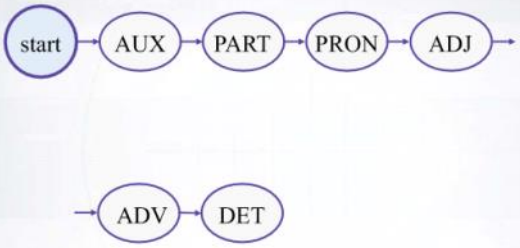


Text generation in HMM: an example

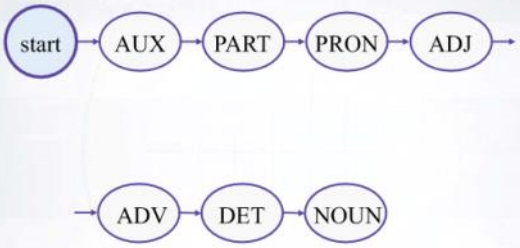
Text generation in HMM: an example



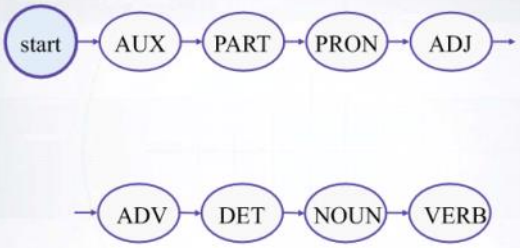
Text generation in HMM: an example



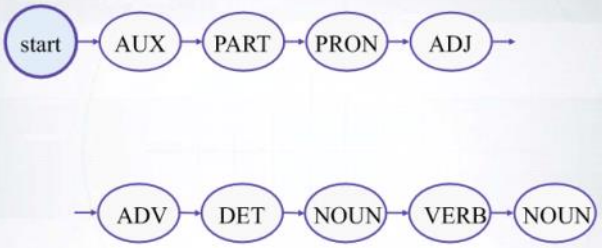
Text generation in HMM: an example



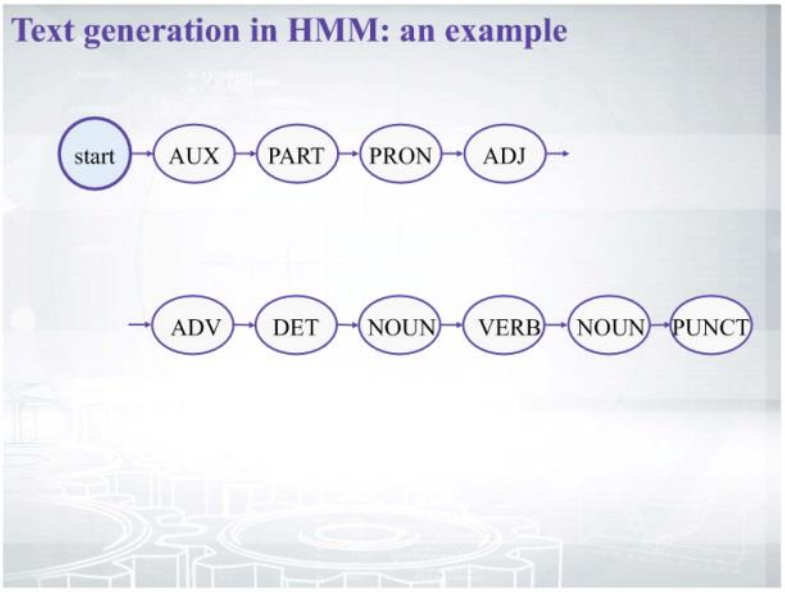
Text generation in HMM: an example



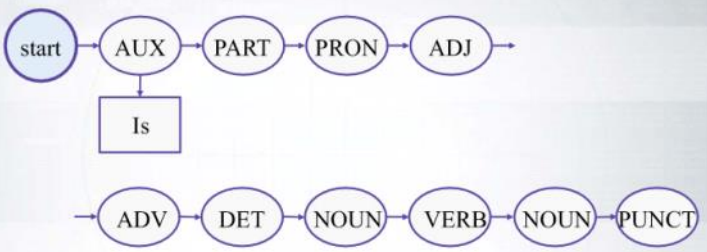
Text generation in HMM: an example



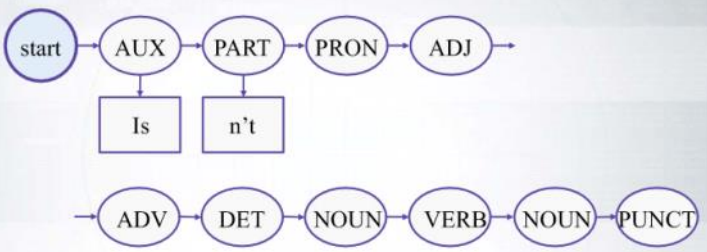
Text generation in HMM: an example



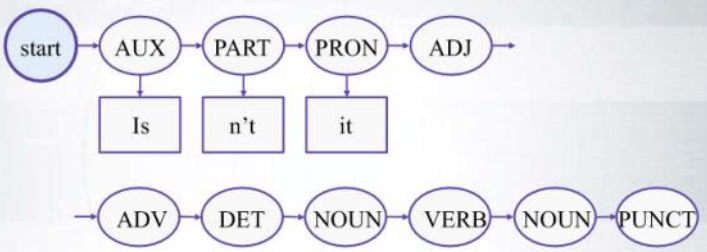
Text generation in HMM: an example



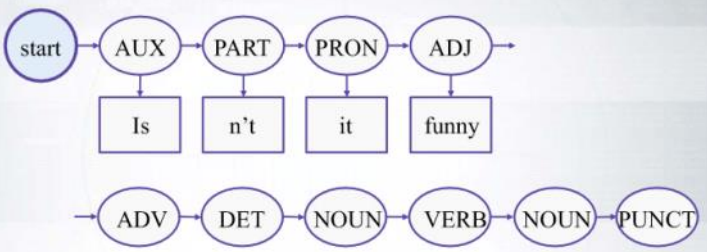
Text generation in HMM: an example



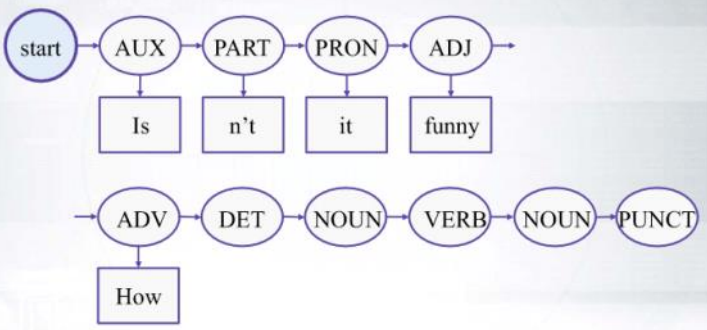
Text generation in HMM: an example



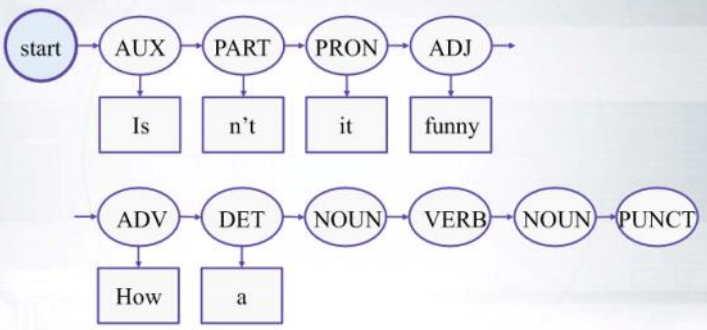
Text generation in HMM: an example



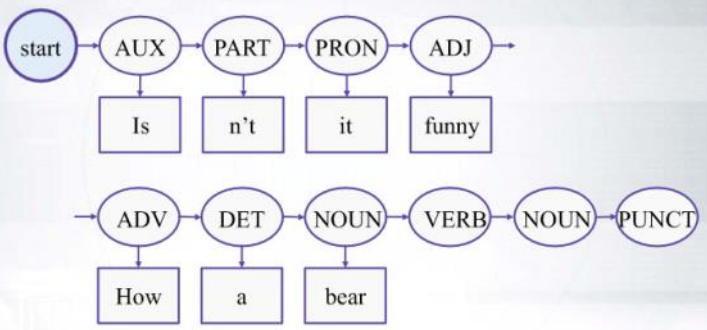
Text generation in HMM: an example



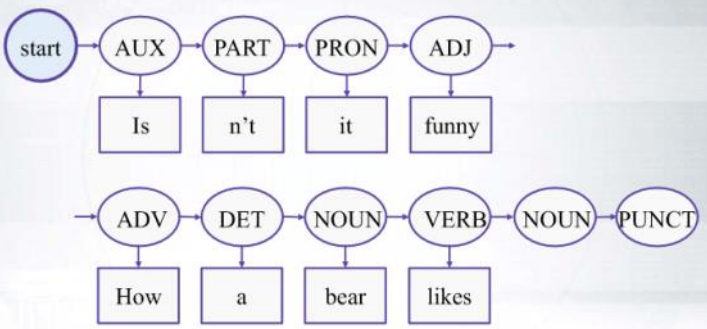
Text generation in HMM: an example



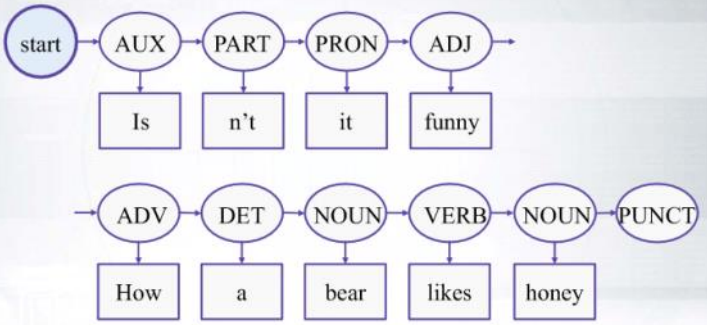
Text generation in HMM: an example



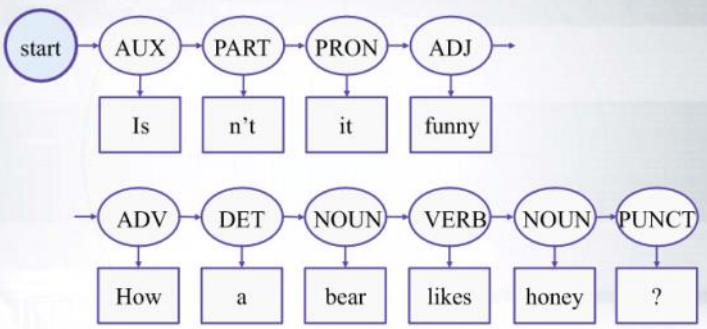
Text generation in HMM: an example



Text generation in HMM: an example



Text generation in HMM: an example



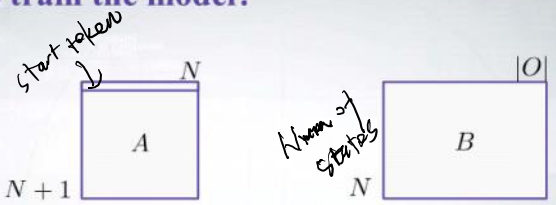
Formal definition of HMM

A Hidden Markov Model is specified by:

1. The set $S = s_1, s_2, \ldots, s_N$ of hidden states
2. The start state s_0
3. The matrix A of transition probabilities: $a_{ij} = p(s_j|s_i)$
4. The set O of possible visible outcomes
5. The matrix B of output probabilities: $b_{kj} = p(o_k|s_j)$

Num of parameters $N(N+1) + KN$

How to train the model?



If we could see the tags in train set, we would count:

$$a_{ij} = p(s_j|s_i) = \frac{c(s_i \to s_j)}{c(s_i)}$$

$$b_{ik} = p(o_k|s_i) = \frac{c(s_i \to o_k)}{c(s_i)}$$

Don't really understand ...

Supervised case: MLE

The same in more formal terms (this is MLE!):

$$a_{ij} = p(s_j|s_i) = \frac{\sum_{t=1}^{T} [y_{t-1} = s_i, y_t = s_j]}{\sum_{t=1}^{T} [y_t = s_i]}$$

Note that the corpus is considered as a single sequence of length T with special states between the sentences.

can not train indicator functions :)

Supervised case: MLE

The same in more formal terms (this is MLE!):

$$a_{ij} = p(s_j|s_i) = \frac{\sum_{t=1}^{T} [y_{t-1} = s_i, y_t = s_j]}{\sum_{t=1}^{T} [y_t = s_i]}$$

Note that the corpus is considered as a single sequence of length T with special states between the sentences.

But we do not see the labels! How to train the model?

Baum-Welch algorithm (a sketch)

E-step: posterior probabilities for hidden variables:

$$p(y_{t-1} = s_i, y_t = s_j)$$

Can be effectively done with dynamic programming (forward-backward algorithm)

M-step: maximum likelihood updates for the parameters:

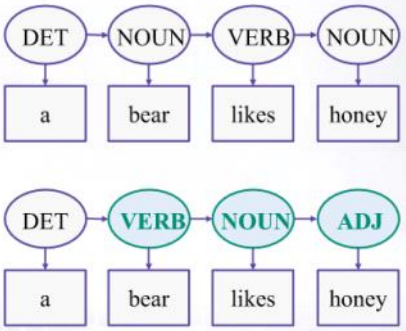
$$a_{ij} = p(s_j|s_i) = \frac{\sum_{t=1}^{T} p(y_{t-1} = s_i, y_t = s_j)}{\sum_{t=1}^{T} p(y_t = s_i)}$$

viterbi


Viterbi algorithm:
what are the most probable tags?

Motivation

The same output sentence can be generated by different sequences of hidden states:



Decoding in HMM

$$p(\mathbf{x}, \mathbf{y}) = \prod_{t=1}^{T} p(y_t|y_{t-1})p(x_t|y_t)$$

Transition probabilities

Output probabilities

Decoding problem:

What is the most probable sequence of hidden states?

$$\mathbf{y} = \text{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\mathbf{y}}\, p(\mathbf{x}, \mathbf{y})$$

Solve this problem efficiently using dynamic programming!

Viterbi decoding

Let $Q_{t,s}$ be the most probable sequence of hidden states of length t that finishes in the state s and generates $o_1, \ldots, o_t$. Let $q_{t,s}$ be the probability of this sequence.

Then $q_{t,s}$ can be computed dynamically:

$$q_{t,s} = \max_{s'} q_{t-1,s'} \cdot p(s|s') \cdot p(o_t|s)$$

Transition probabilities

Output probabilities

$Q_{t,s}$ can be determined by remembering the argmax.

要 approximation 吧?

0.25 × 0.4

An example of HMM: transition probabilities

Suppose that we have the following PoS tags:
ADJ, NOUN, VERB.

Consider the transition probabilities between tags:

from \ to	**ADJ**	**NOUN**	**VERB**
ADJ	0.4	0.4	0.2
NOUN	0.2	0.4	0.4
VERB	0.1	0.6	0.3

Note that the sum of probabilities in each row is equal to 1.

$0.25 \times 0.4 \times$

An example of HMM: transition probabilities

Suppose that we have the following PoS tags:
ADJ, NOUN, VERB.

Consider the transition probabilities between tags:

from \ to	**ADJ**	**NOUN**	**VERB**
ADJ	0.4	0.4	0.2
NOUN	0.2	0.4	0.4
VERB	0.1	0.6	0.3

Note that the sum of probabilities in each row is equal to 1.

Let all initial state probabilities be equal (to 1/3).

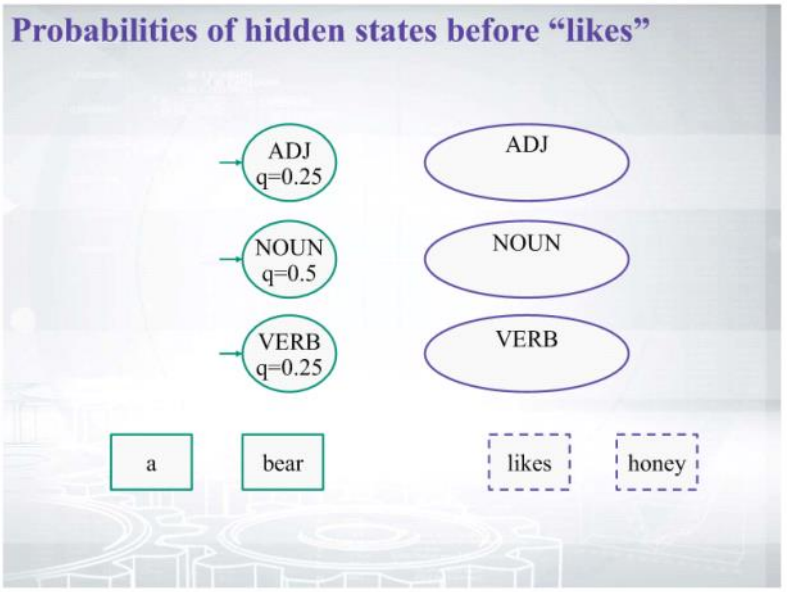
An example of HMM: output probabilities

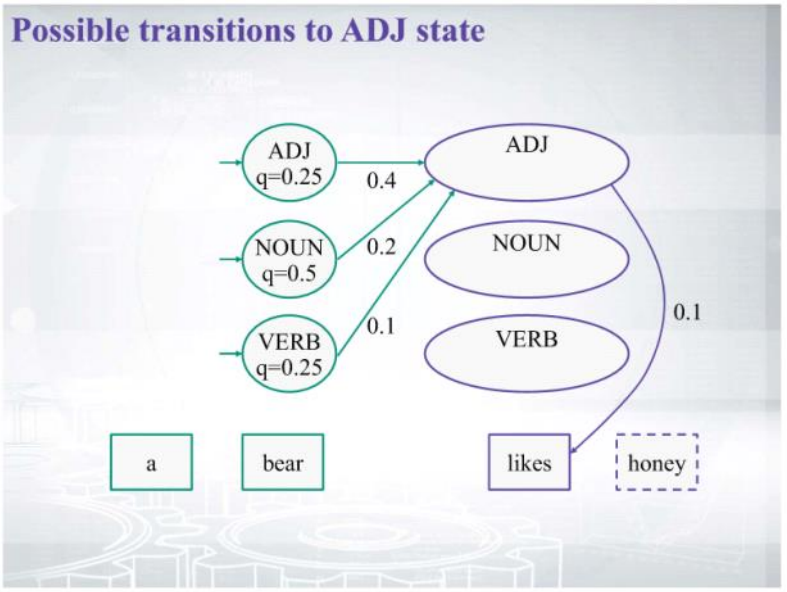
Consider the following output probabilities for the vocabulary:

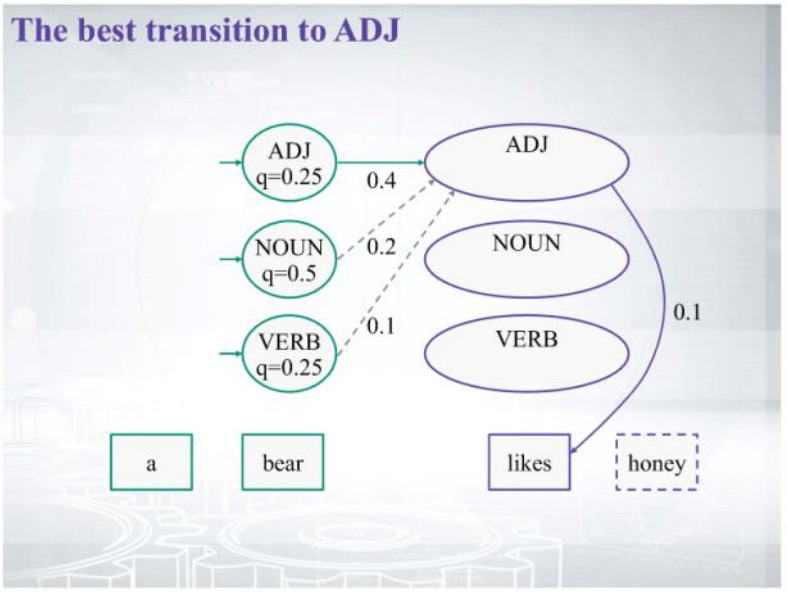
tag\word	a	bear	fly	honey	likes	sweet
ADJ	0.2	0.1	0.1	0.1	0.1	0.4
NOUN	0.1	0.2	0.2	0.2	0.2	0.1
VERB	0.1	0.2	0.2	0.1	0.3	0.1

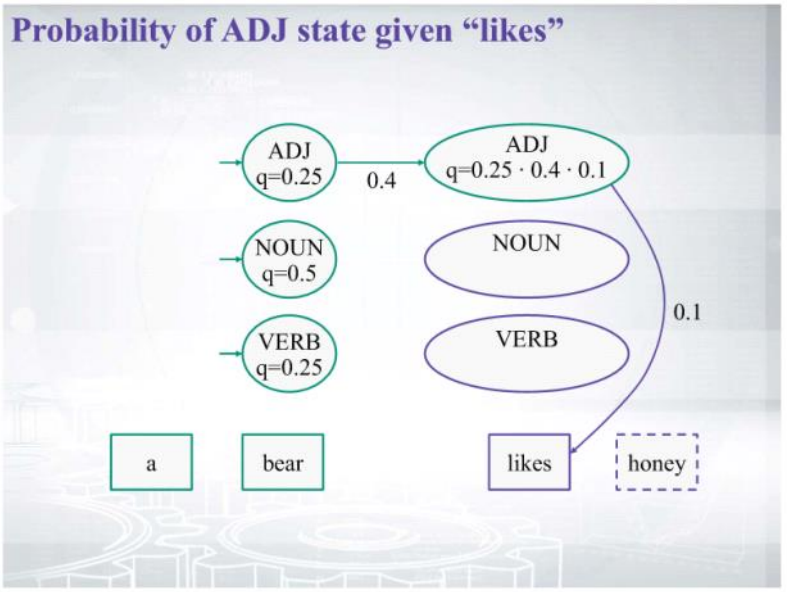
The probabilities in each row also sum to 1.

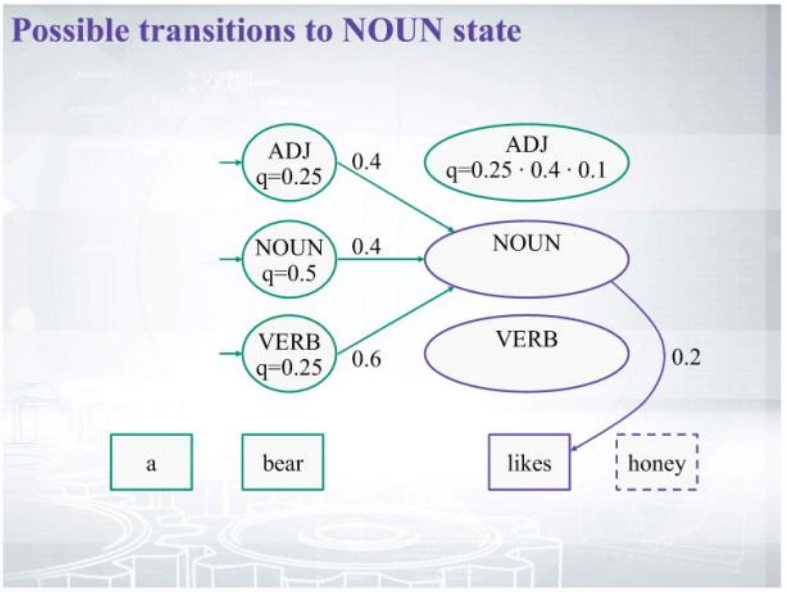
Probabilities of hidden states before “likes”

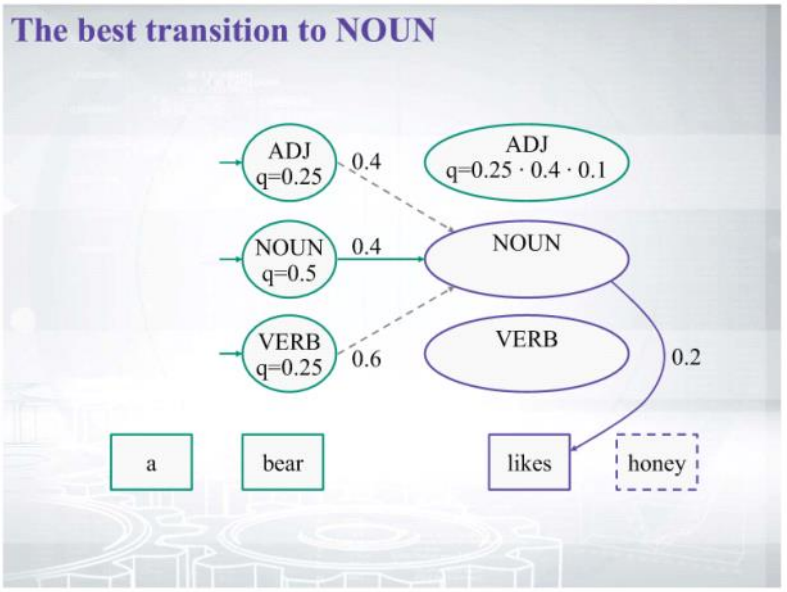


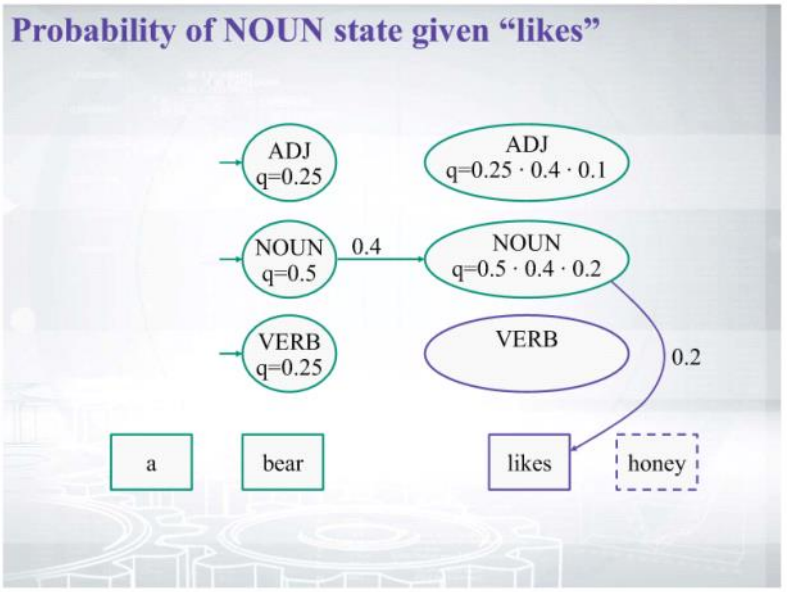

Possible transitions to ADJ state
ADJ
q=0.25
NOUN
q=0.5
VERB
q=0.25
0.4
0.2
0.1
ADJ
NOUN
VERB
0.1
a
bear
likes
honey

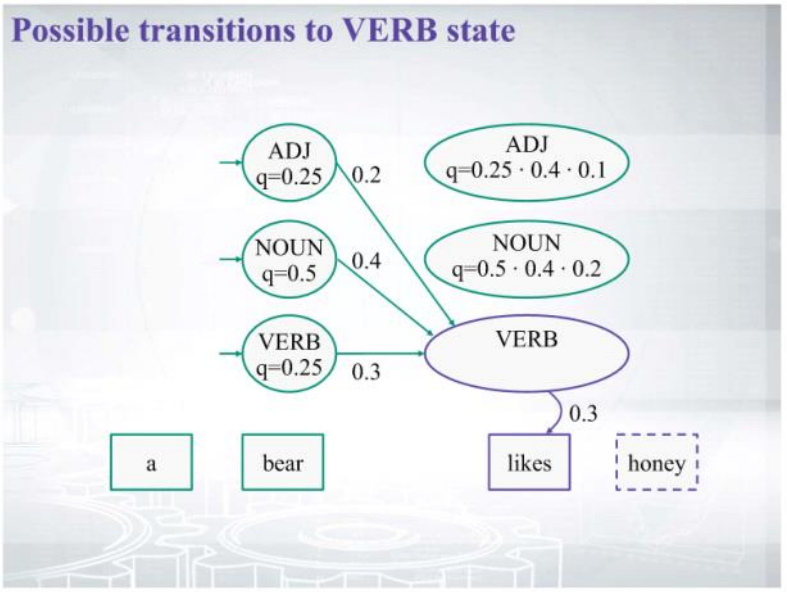

The best transition to ADJ
ADJ
q=0.25
NOUN
q=0.5
VERB
q=0.25
0.4
0.2
0.1
ADJ
NOUN
VERB
0.1
a
bear
likes
honey

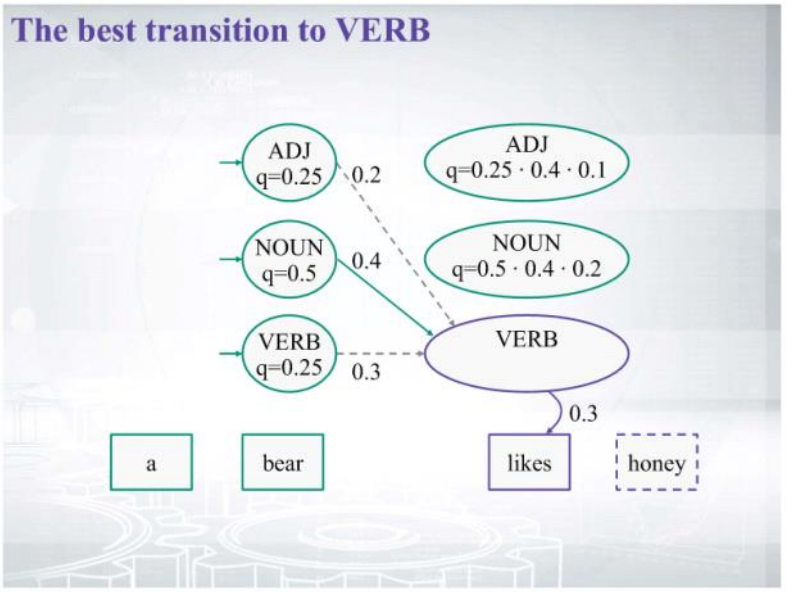

Probability of ADJ state given "likes"
ADJ
q=0.25
0.4
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5
NOUN
0.1
VERB
q=0.25
VERB
a
bear
likes
honey

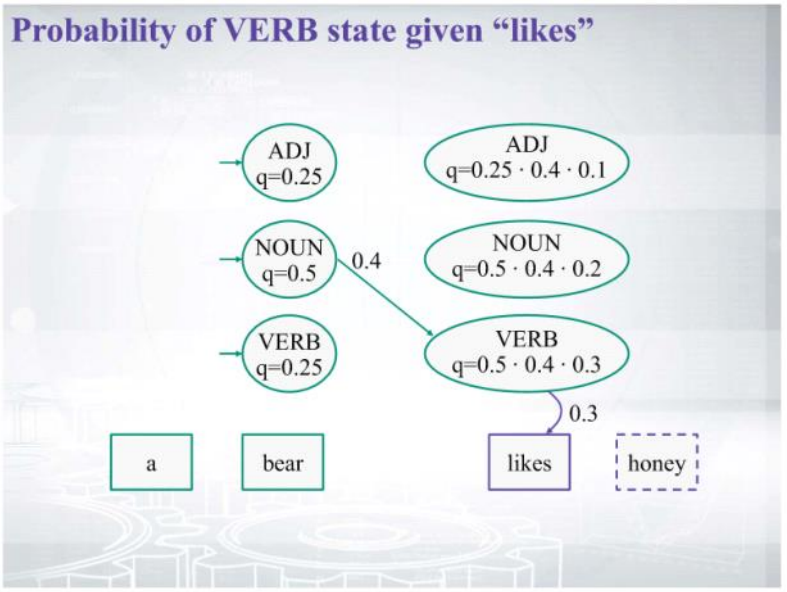

Possible transitions to NOUN state
ADJ
q=0.25
0.4
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5
0.4
NOUN
VERB
q=0.25
0.6
VERB
0.2
a
bear
likes
honey

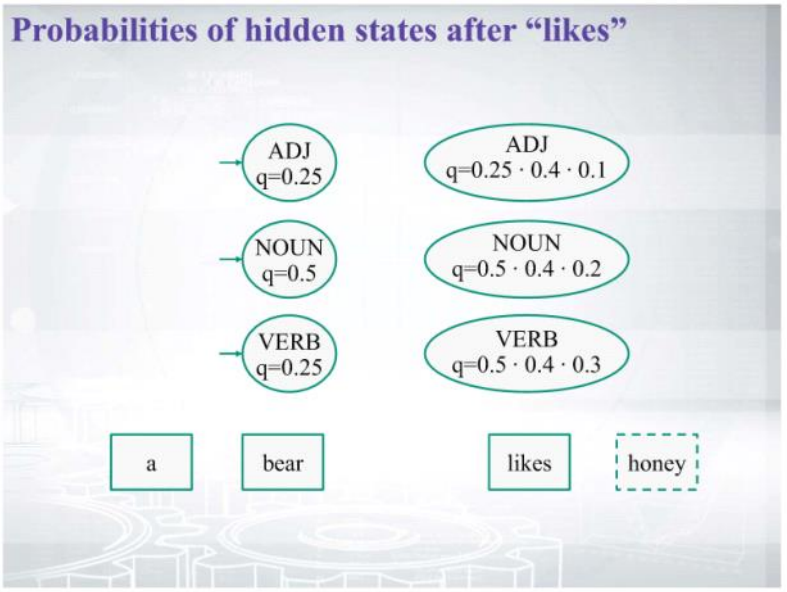

The best transition to NOUN
ADJ
q=0.25
0.4
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5
0.4
NOUN
VERB
q=0.25
0.6
VERB
0.2
a
bear
likes
honey


Probability of NOUN state given "likes"
ADJ
q=0.25
NOUN
q=0.5
VERB
q=0.25
0.4
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5 · 0.4 · 0.2
VERB
0.2
a
bear
likes
honey

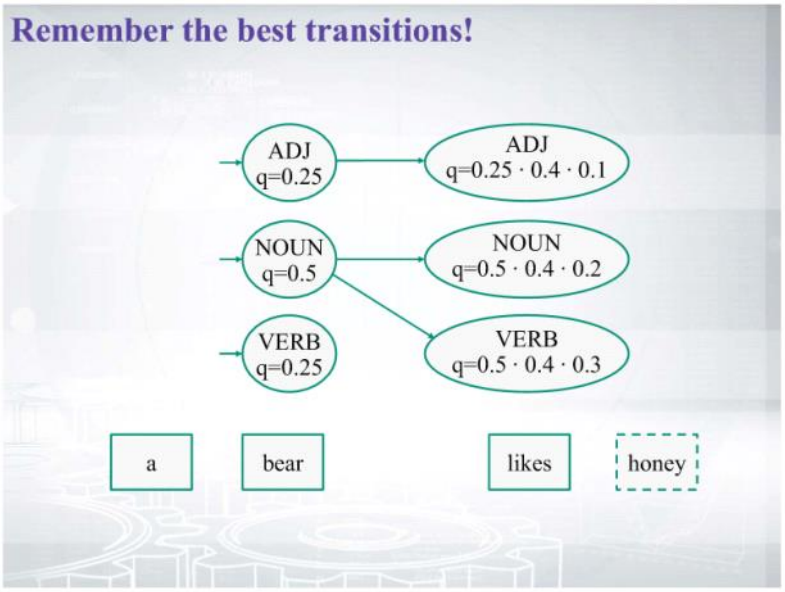

Possible transitions to VERB state
ADJ
q=0.25
0.2
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5
0.4
NOUN
q=0.5 · 0.4 · 0.2
VERB
q=0.25
0.3
VERB
0.3
a
bear
likes
honey


The best transition to VERB
ADJ
q=0.25
0.2
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5
0.4
NOUN
q=0.5 · 0.4 · 0.2
VERB
q=0.25
0.3
VERB
0.3
a
bear
likes
honey

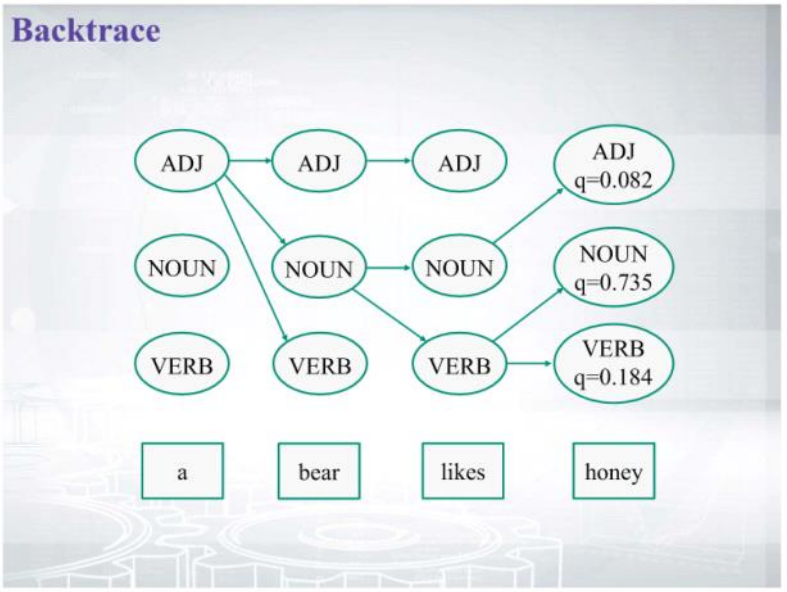

Probability of VERB state given “likes”
ADJ
q=0.25
NOUN
q=0.5
VERB
q=0.25
0.4
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5 · 0.4 · 0.2
VERB
q=0.5 · 0.4 · 0.3
0.3
a
bear
likes
honey


Probabilities of hidden states after “likes”
ADJ
q=0.25
NOUN
q=0.5
VERB
q=0.25
ADJ
q=0.25 · 0.4 · 0.1
NOUN
q=0.5 · 0.4 · 0.2
VERB
q=0.5 · 0.4 · 0.3
a
bear
likes
honey

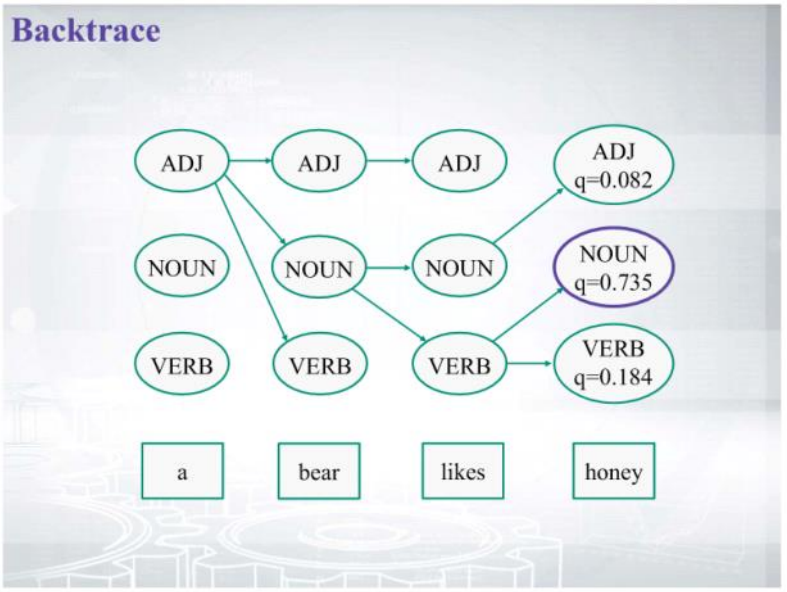
Remember the best transitions!

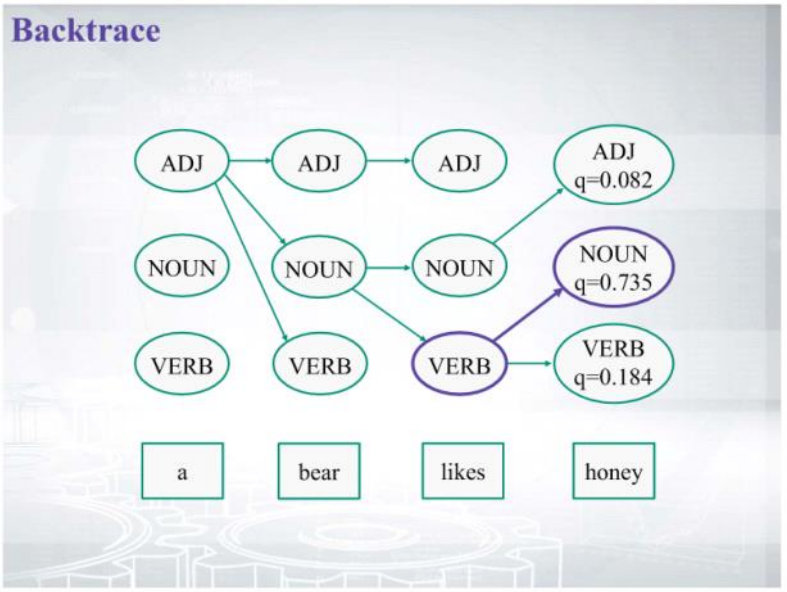


Backtrace

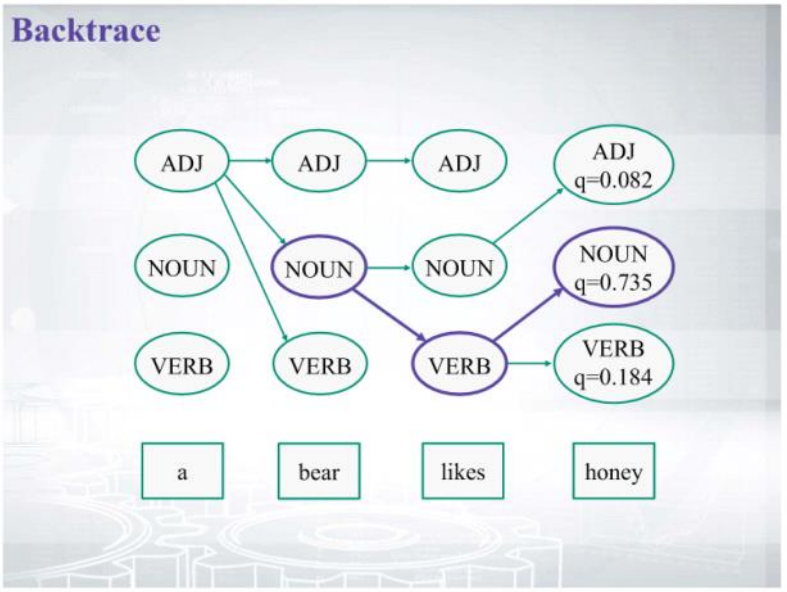

ADJ
ADJ
ADJ
ADJ
q=0.082
NOUN
NOUN
NOUN
NOUN
q=0.735
VERB
VERB
VERB
VERB
q=0.184
a
bear
likes
honey

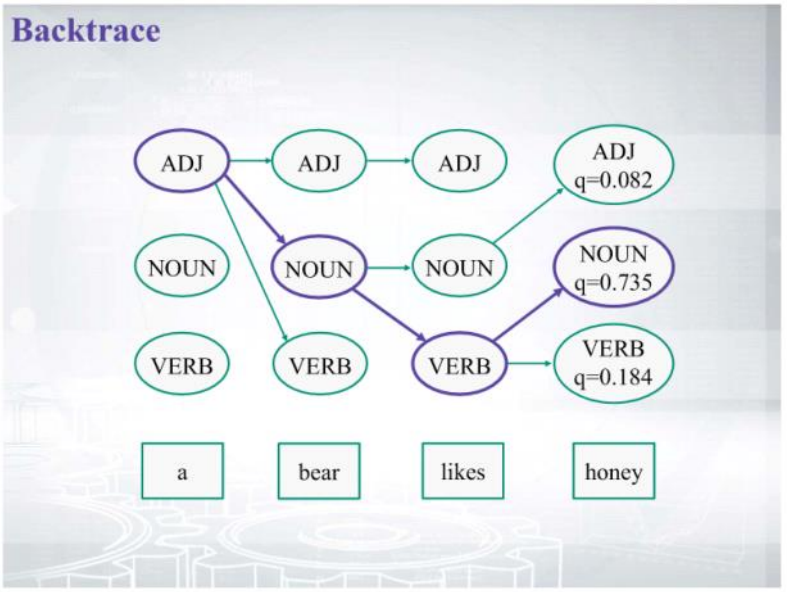
Backtrace


ADJ
ADJ
ADJ
ADJ
q=0.082
NOUN
NOUN
NOUN
NOUN
q=0.735
VERB
VERB
VERB
VERB
q=0.184
a
bear
likes
honey



Hmm... What's this?


Backtrace
ADJ
ADJ
ADJ
ADJ
q=0.082
NOUN
NOUN
NOUN
NOUN
q=0.735
VERB
VERB
VERB
VERB
q=0.184
a
bear
likes
honey


Backtrace
ADJ
ADJ
ADJ
ADJ
q=0.082
NOUN
NOUN
NOUN
NOUN
q=0.735
VERB
VERB
VERB
VERB
q=0.184
a
bear
likes
honey

Viterbi algorithm

Input: observations of length T, state-graph of length N
Output: best-path

for each state s from 1 to N do
$q[1, s] \leftarrow p(s|s_0) \cdot p(o_1|s)$
backpointers$[1, s] \leftarrow 0$

for each time step t from 2 to T do
  for each state s from 1 to N do
    $q[t, s] \leftarrow \max_{s'=1}^{N} q[t-1, s'] \cdot p(s|s') \cdot p(o_t|s)$
    backpointers $[t, s] \leftarrow \text{argmax}_{s'=1}^{N} q[t-1, s'] \cdot p(s|s')$

$s \leftarrow \text{argmax}_{s'=1}^{N} q[T, s']$
return the backtrace path from backpointers $[T, s]$

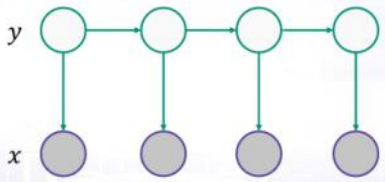
memm


MEMMs, CRFs and other sequential
models for Named Entity Recognition

Hidden Markov Model

$$p(\mathbf{y}, \mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1})p(x_t|y_t)$$

Generative model



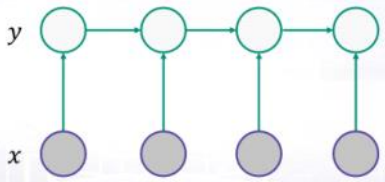
Maximum Entropy Markov Model

$$p(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} p(y_t|y_{t-1}, x_t)$$

Discriminative model

not generative

→ the text is observable only produce y given x



Maximum Entropy Markov Model

$$p(y_t|y_{t-1}, x_t) = \frac{1}{Z_t(y_{t-1}, x_t)} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

Normalization constant

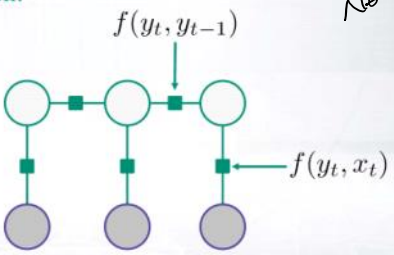
weight

feature

Conditional Random Field (linear chain)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} \prod_{t=1}^{T} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y_t, y_{t-1}, x_t)\right)$$

Undirected graph:



Complicated normalization

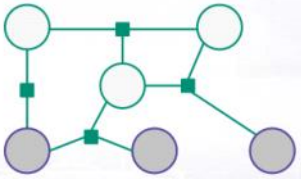
Discriminative model

No direction of links among nodes

Conditional Random Field (general form)

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(x)} \prod_{a=1}^{A} \Psi_a(y_a, x_a)$$

Arbitrary factors



Black-box implementations

CRF++	https://sourceforge.net/projects/crfpp/
MALLET	http://mallet.cs.umass.edu/
GRMM	http://mallet.cs.umass.edu/grmm/
CRFSuite	http://www.chokkan.org/software/crfsuite/
FACTORIE	http://www.factorie.cc

http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf

need to generate features.

Features engineering

Label-observation features:

- $f(y_t, y_{t-1}, x_t) = [y_t = y]\, g_m(x_t)$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']$
- $f(y_t, y_{t-1}, x_t) = [y_t = y][y_{t-1} = y']\, g_m(x_t)$

Observation function examples

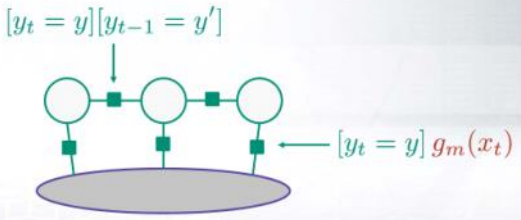
	$w_t = v$	$\forall v \in$
	part-of-speech tag for w_t is j	$\forall$ tags j
	w_t is in a phrase of syntactic type j	$\forall$ tags j
Capitalized	w_t matches [A-Z][a-z]+	
AllCaps	w_t matches [A-Z]+	
EndsInDot	w_t matches [^\.]+.*\.	
	w_t matches a dash	
	w_t appears in a list of stop words	
	w_t appears in list of capitals	

http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf

Dependencies on input

Trick: Pretend the current input x_t contains not only the current word w_t, but also w_{t-1} and w_{t+1} and build observation functions for them as well.

The model is discriminative, so we can use the whole input:



Resume for the lesson

Probabilistic graphical models:

- Hidden Markov Models (generative, directed)
- Maximum Entropy Markov Models (discriminative, directed)
- Conditional Random Field (discriminative, undirected)

Tasks:

- Training – fit parameters (Baum-Welch for HMM)
- Decoding – find the most probable tags (Viterbi for HMM)

Practice:

- Features engineering + black-box implementation

4. Consider a Hidden Markov Model with three hidden states: N (noun), V (verb) and O (other). Let all transitions between states be equiprobable. Consider the following possible outputs:

   N: *mimsy* | *borogoves*

   V: *were* | *borogoves*

   O: *All* | *mimsy* | *the*

   Let all these outputs be also equiprobable.

   Consider the sentence "*All mimsy were the borogoves*" and choose the correct statement.

   - There are four possible best tag sequences: ONVON, ONVOV, OOVON, OOVOV. All of them are equiprobable.
   - The best tag sequence is OOVON.
   - The best tag sequence is ONVOV.
   - There are two possible best tag sequences: ONVON and ONVOV. They are equiprobable.
   - The best tag sequence is OOVOV.
   - The best tag sequence is ONVON.