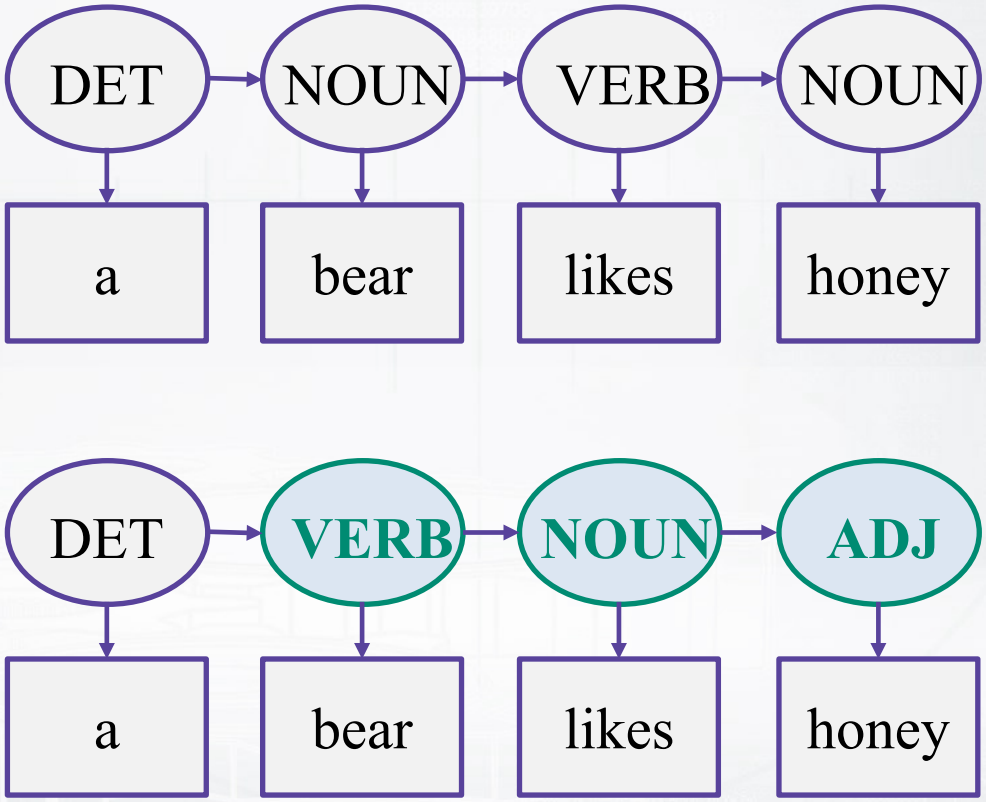
Viterbi algorithm: what are the most probable tags?

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.

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.

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).

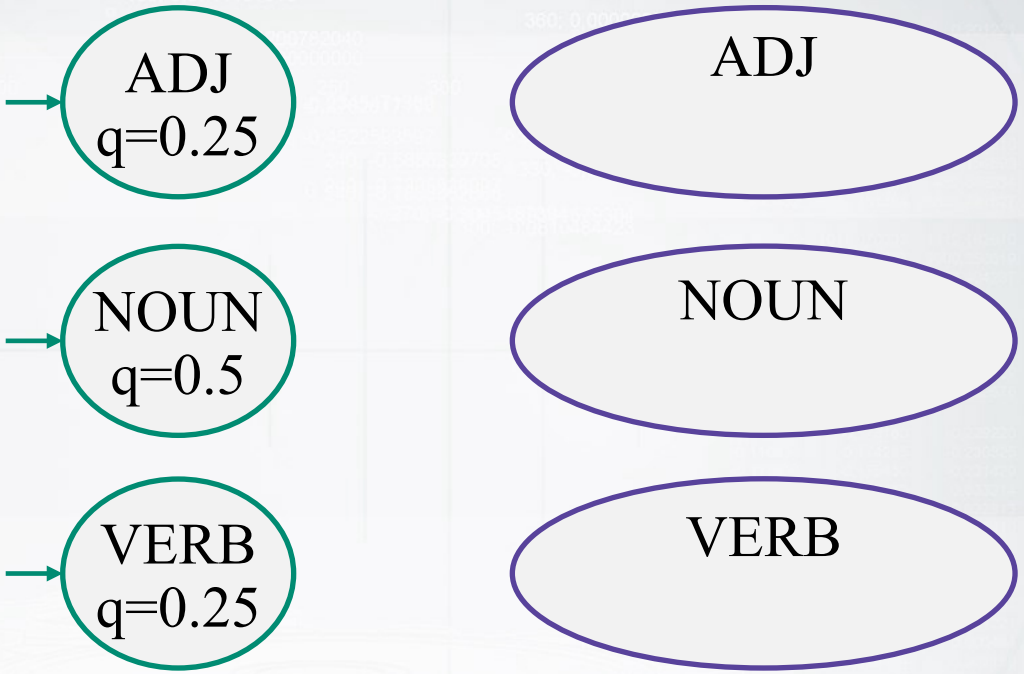
An example of HMM: output probabilities

Consider the following output probabilities for the vocabulary:

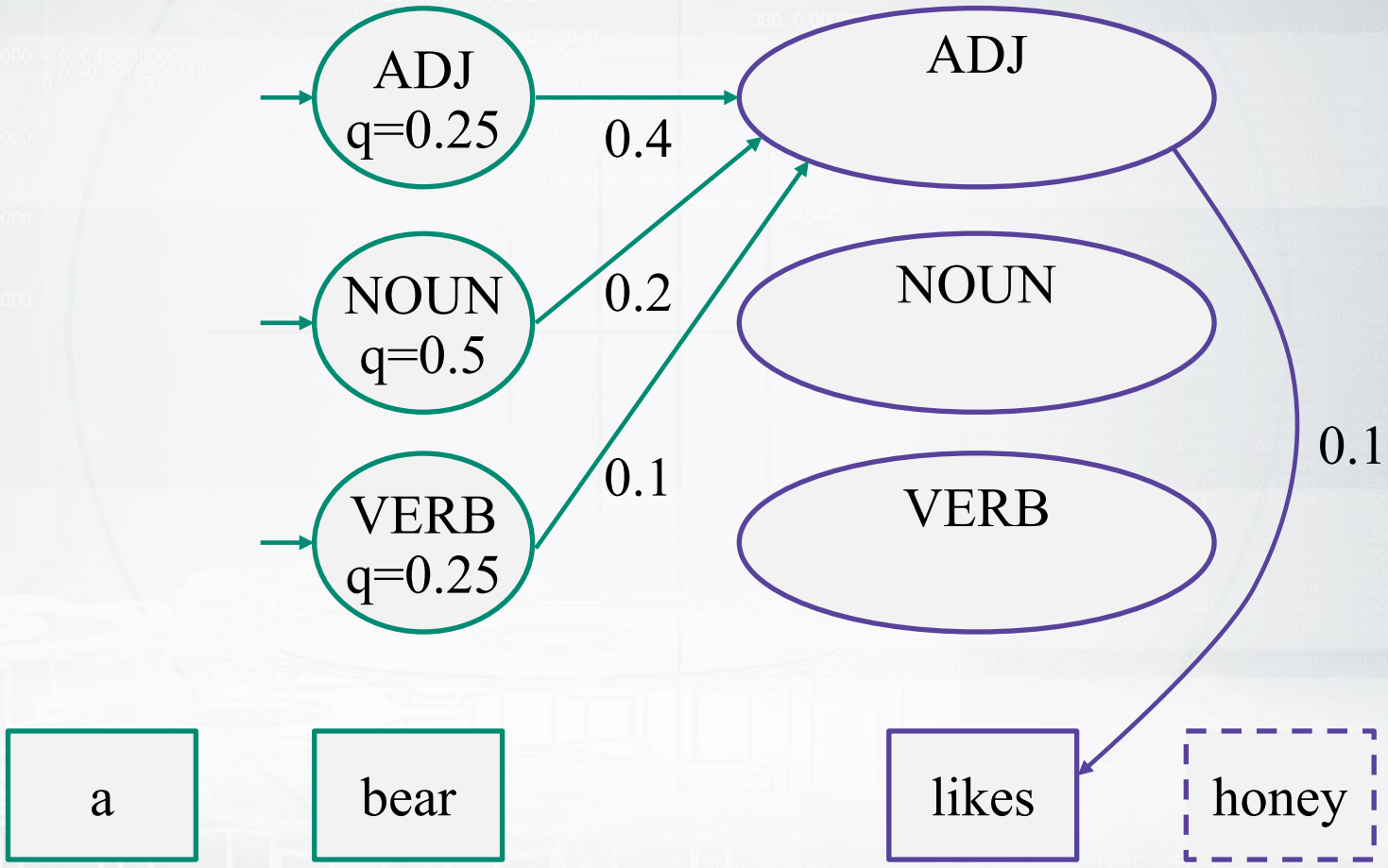
tag\word	a	bear	fly	honey	likes	sweet
ADJ	0.2	0.1	0.1	0.1	0.1	0.4
NOUN	0.1	0.2	0.2	0.2	0.2	0.1
VERB	0.1	0.2	0.2	0.1	0.3	0.1

The probabilities in each row also sum to 1.

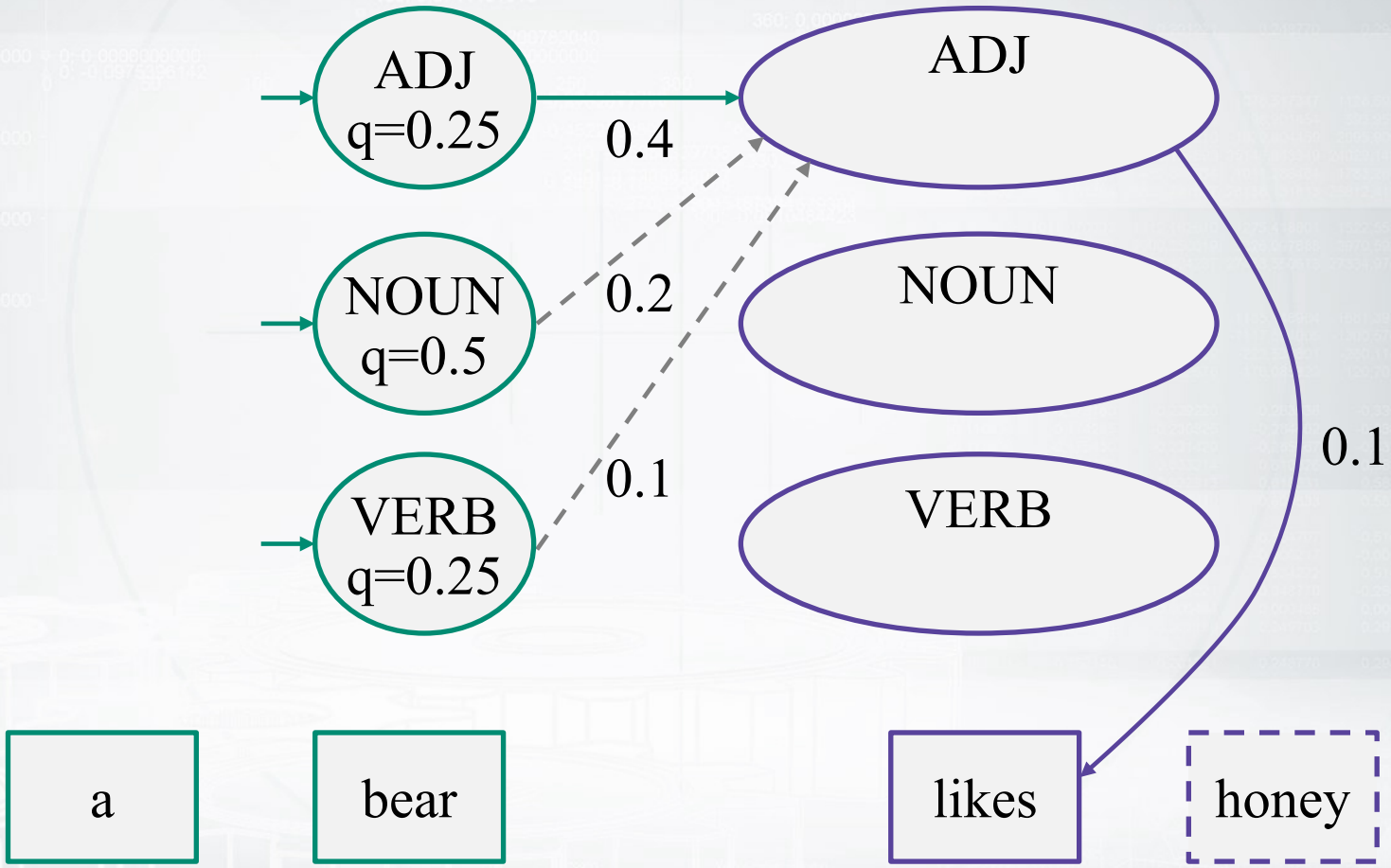
Probabilities of hidden states before “likes”



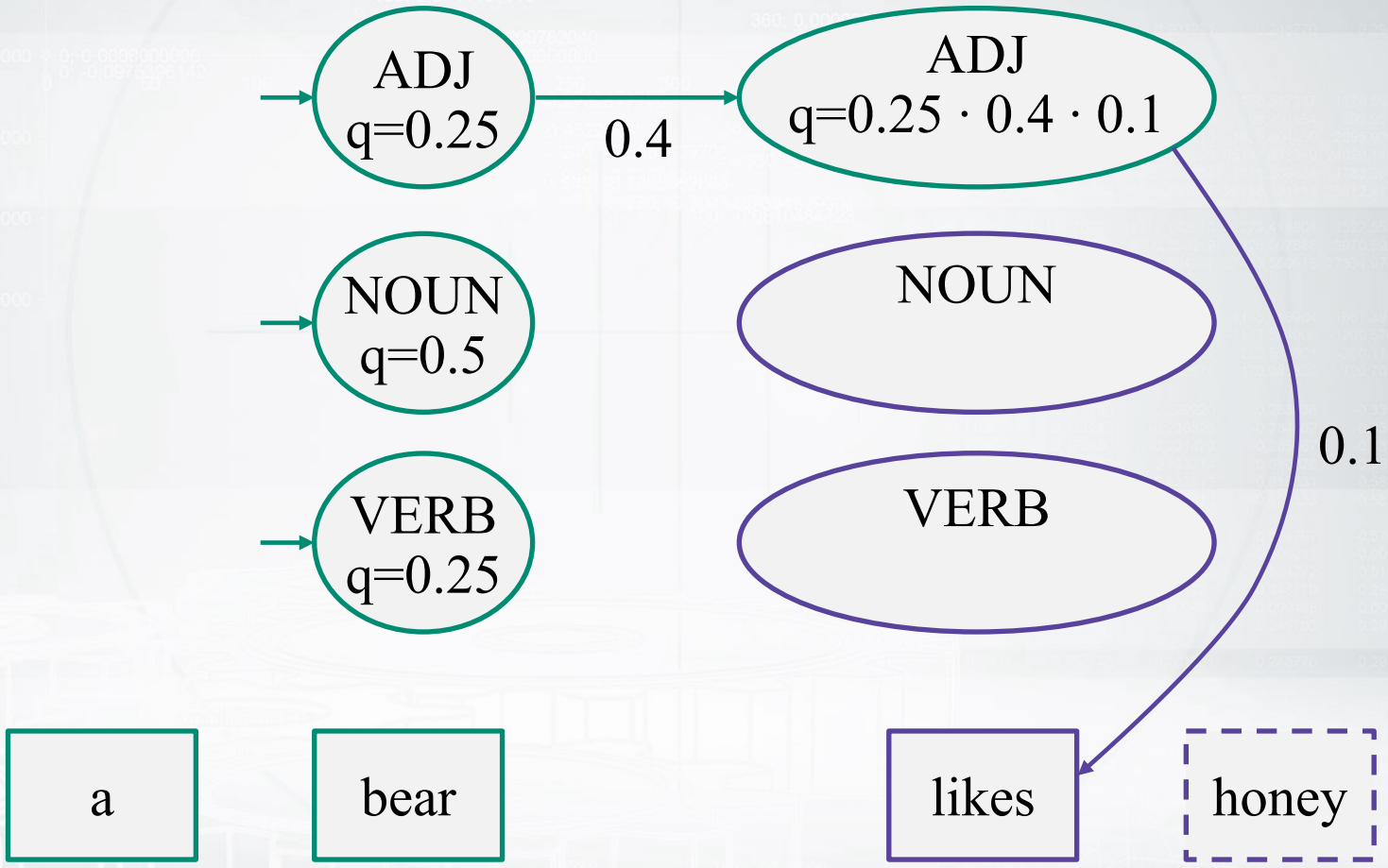
Possible transitions to ADJ state



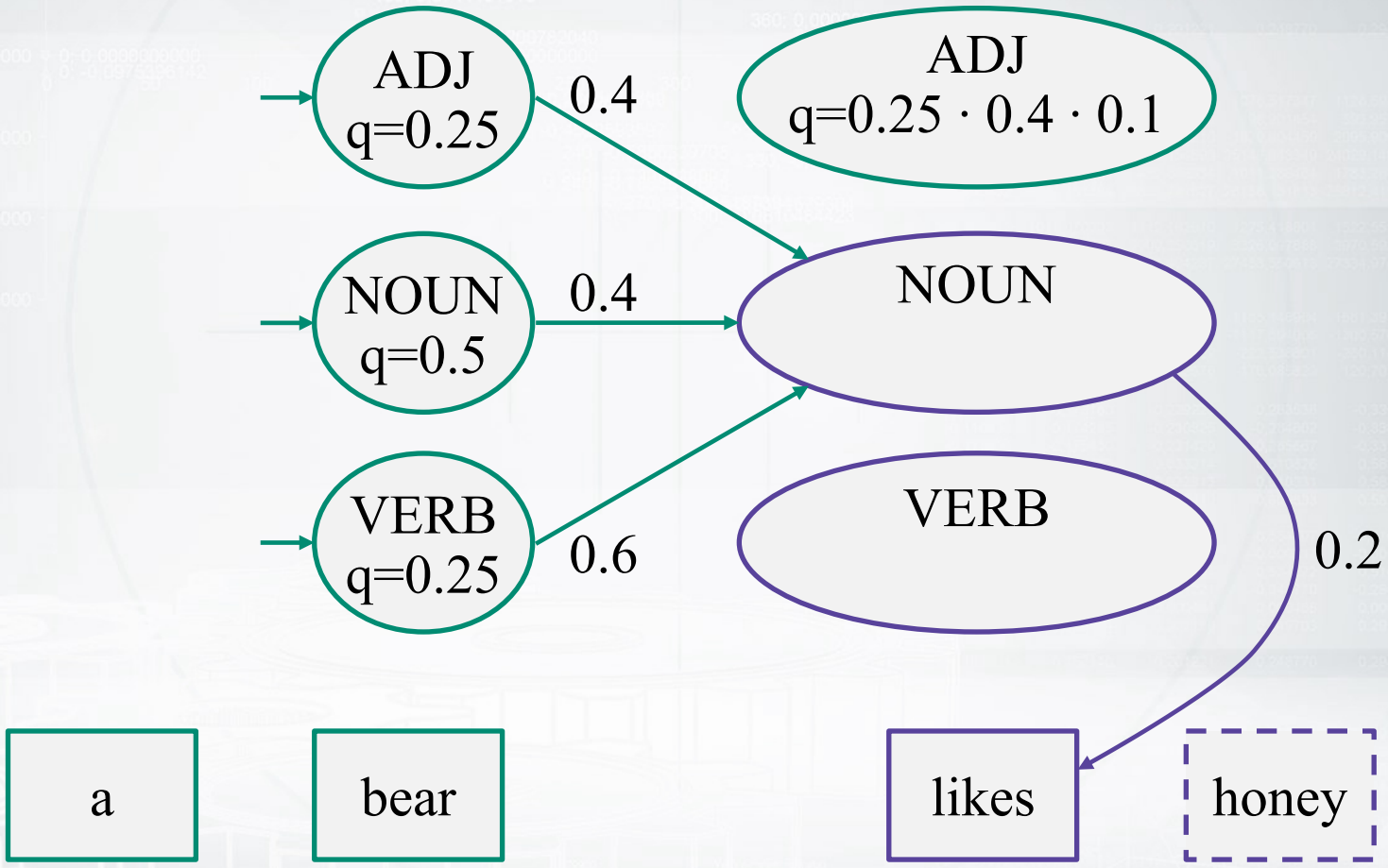
The best transition to ADJ



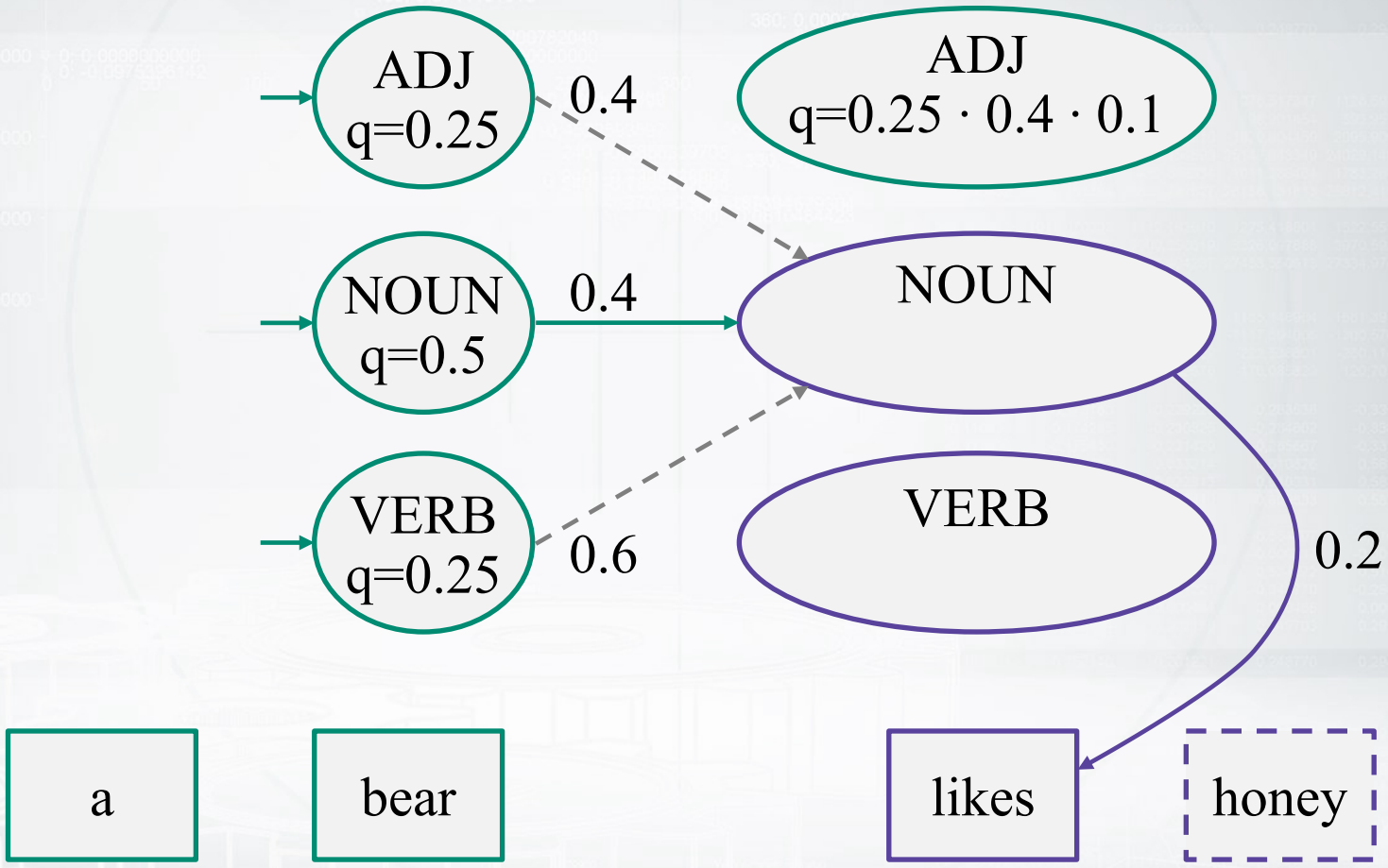
Probability of ADJ state given "likes"



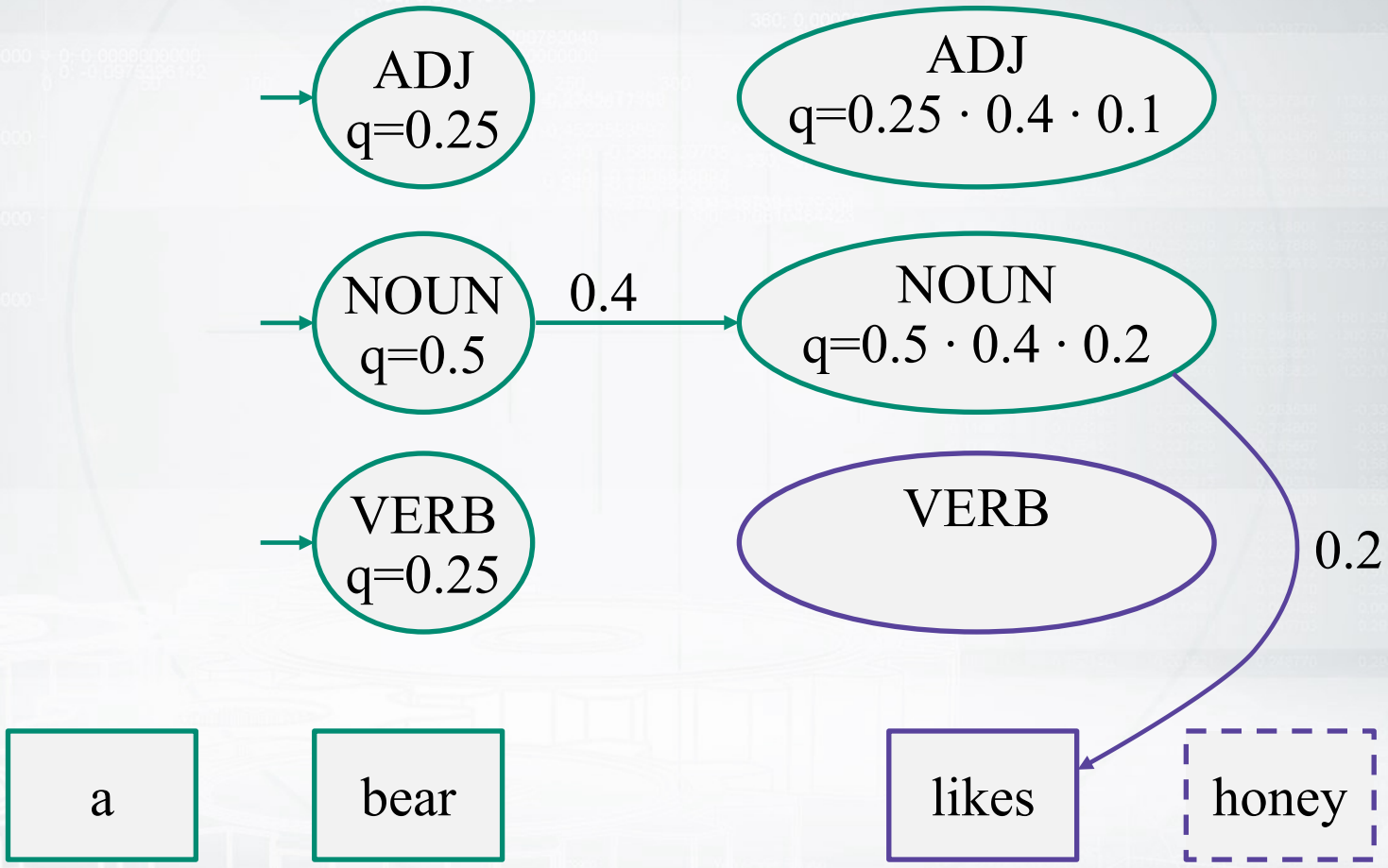
Possible transitions to NOUN state



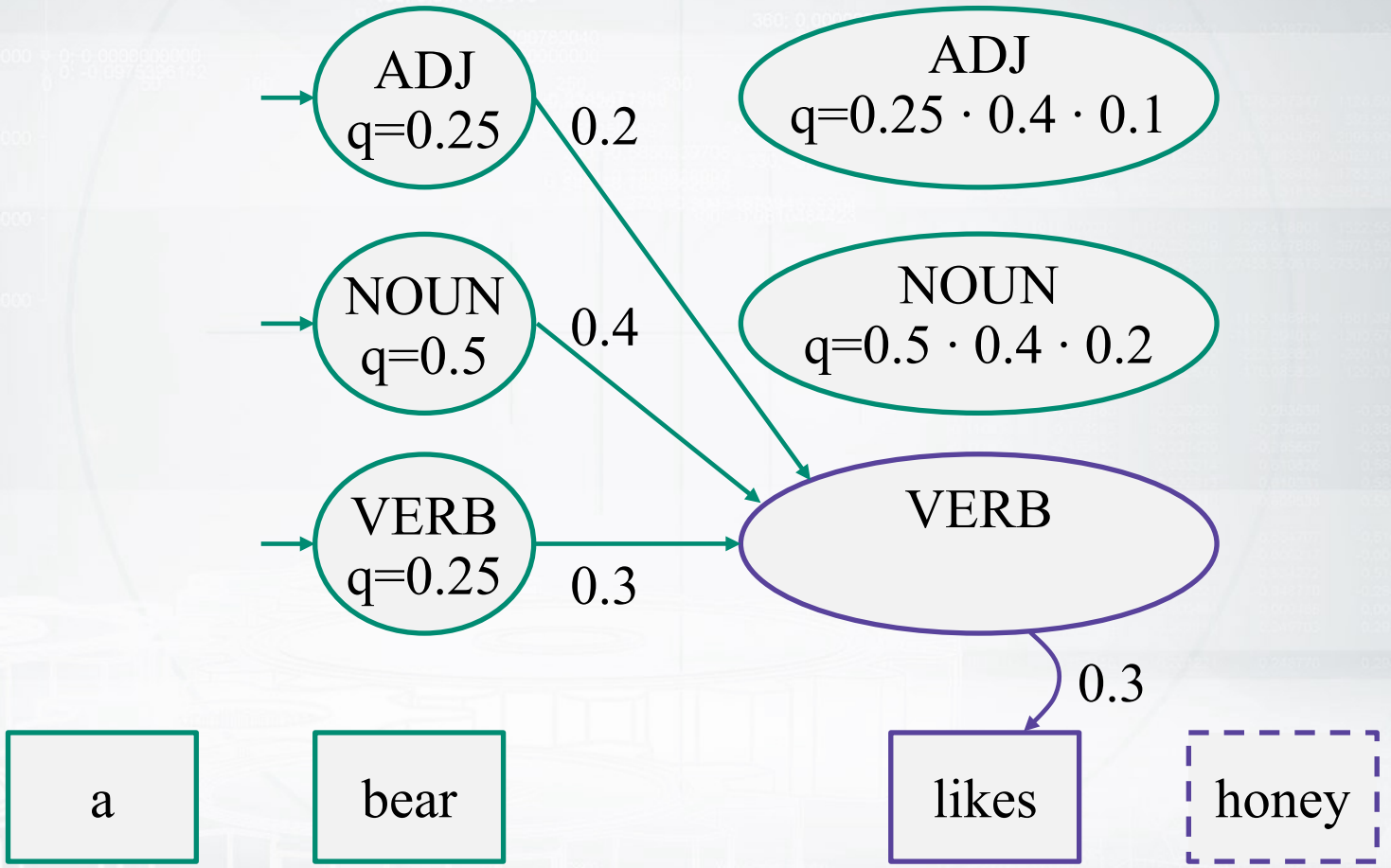
The best transition to NOUN



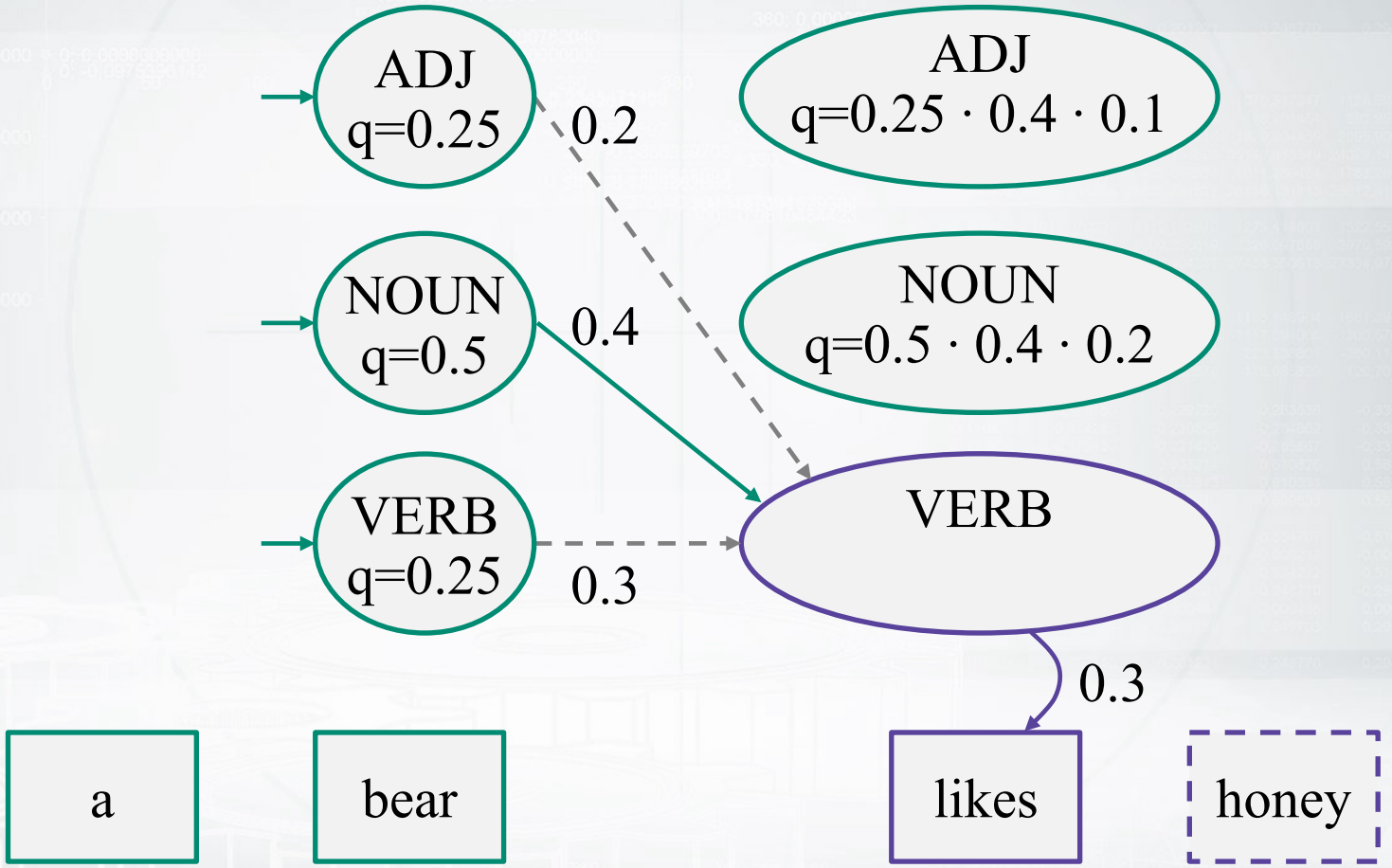
Probability of NOUN state given “likes”



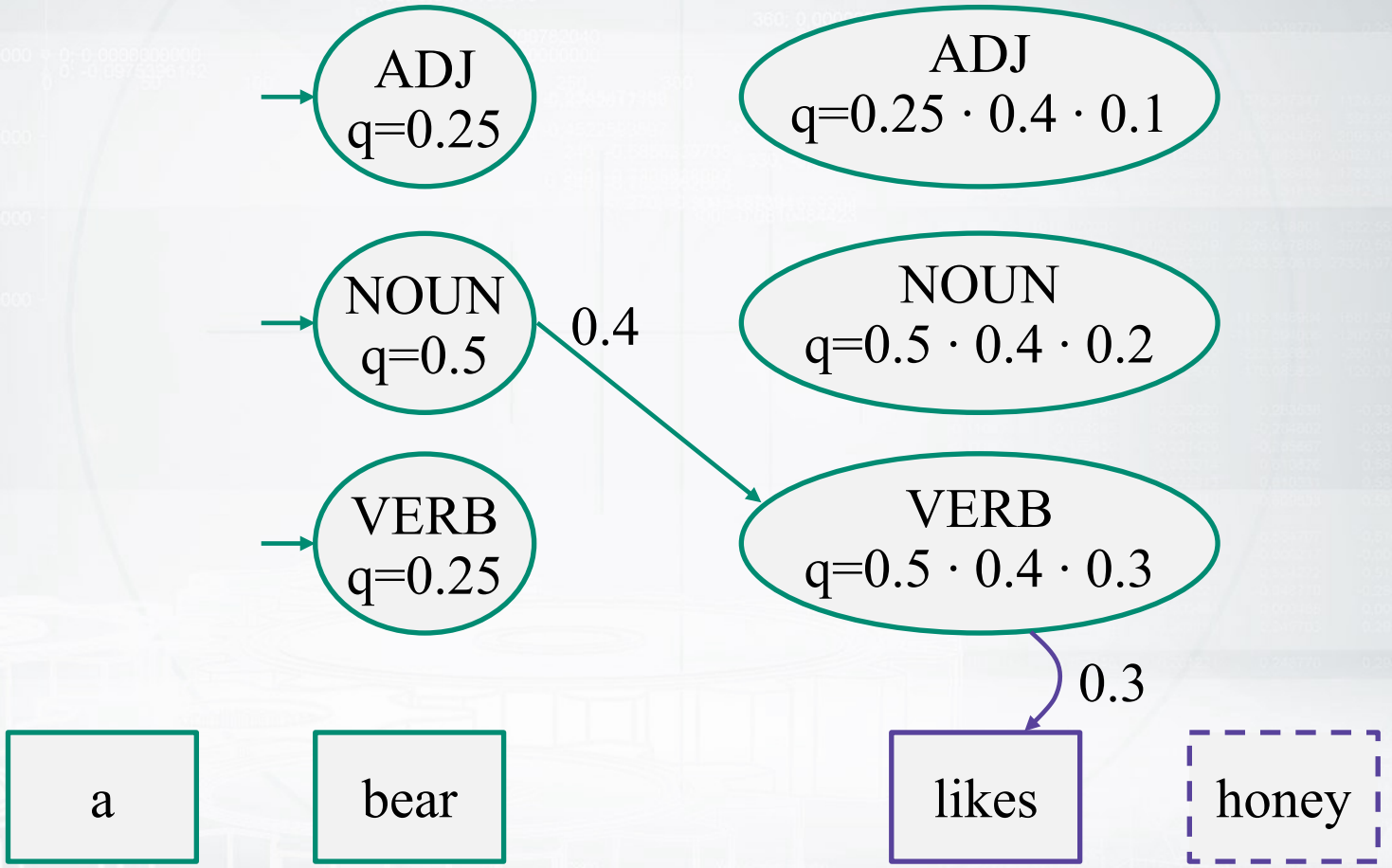
Possible transitions to VERB state



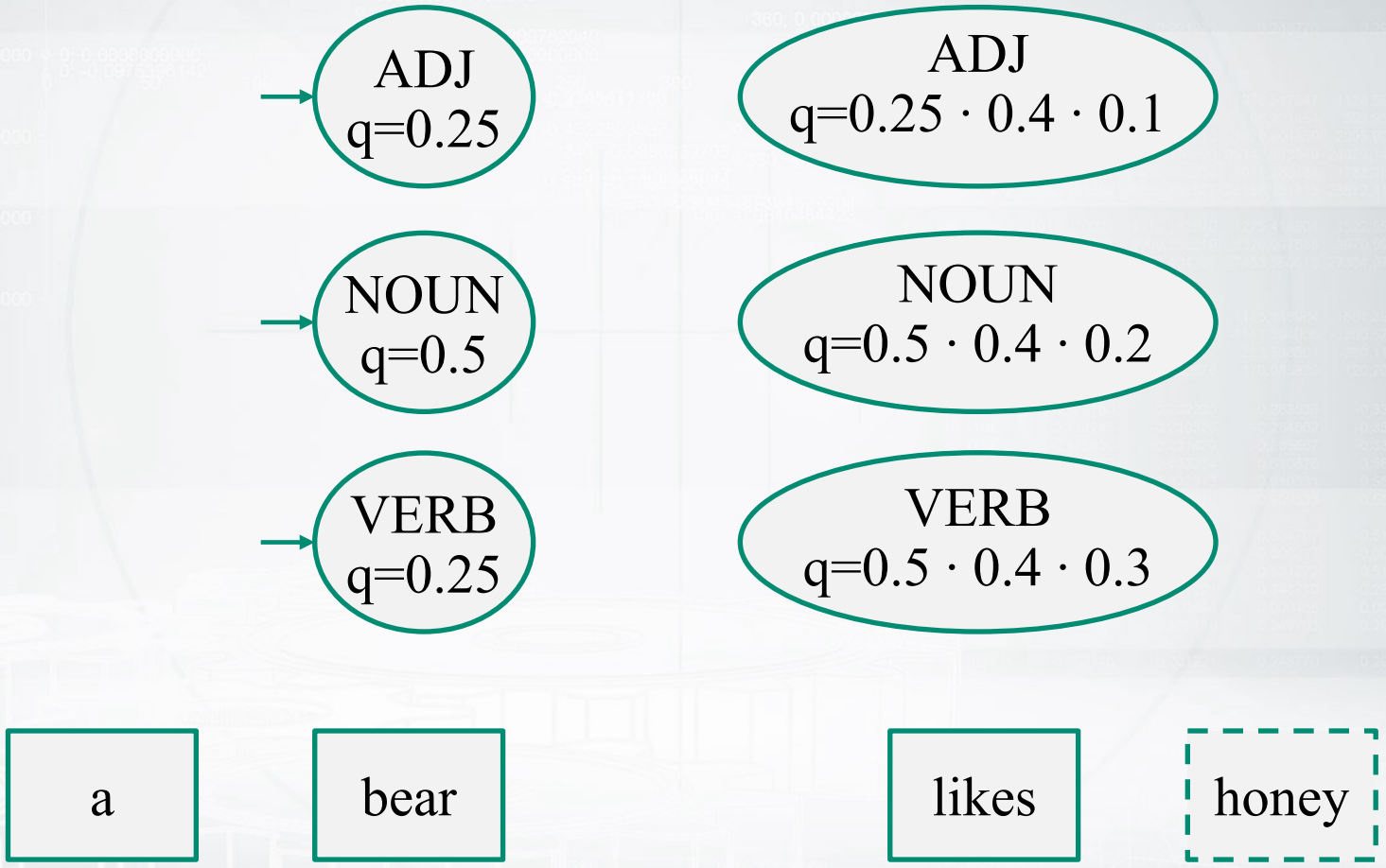
The best transition to VERB



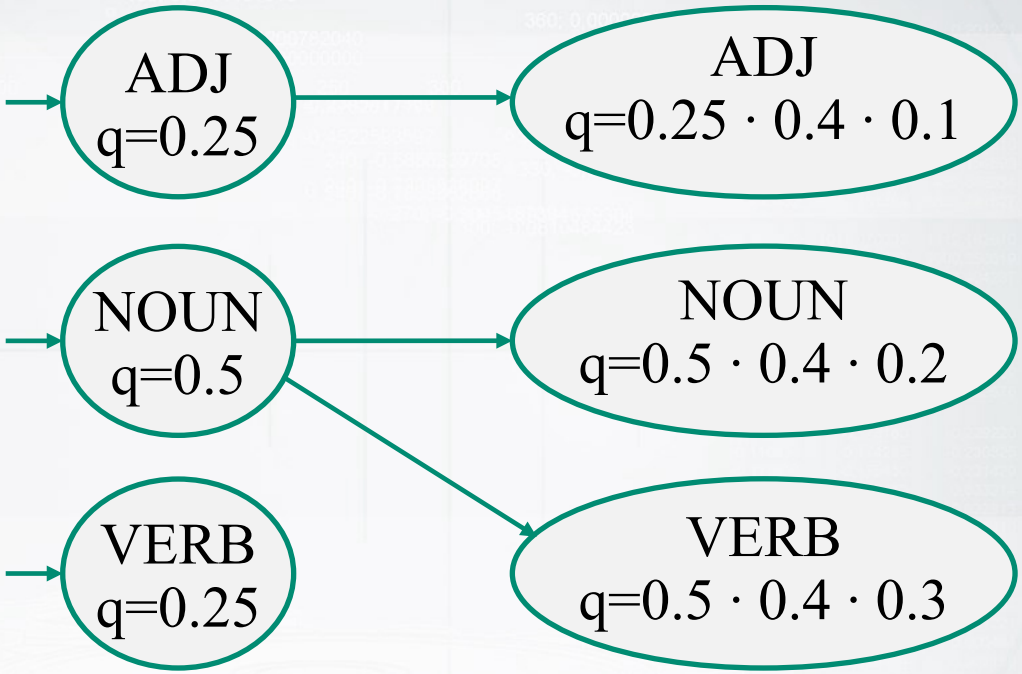
Probability of VERB state given "likes"



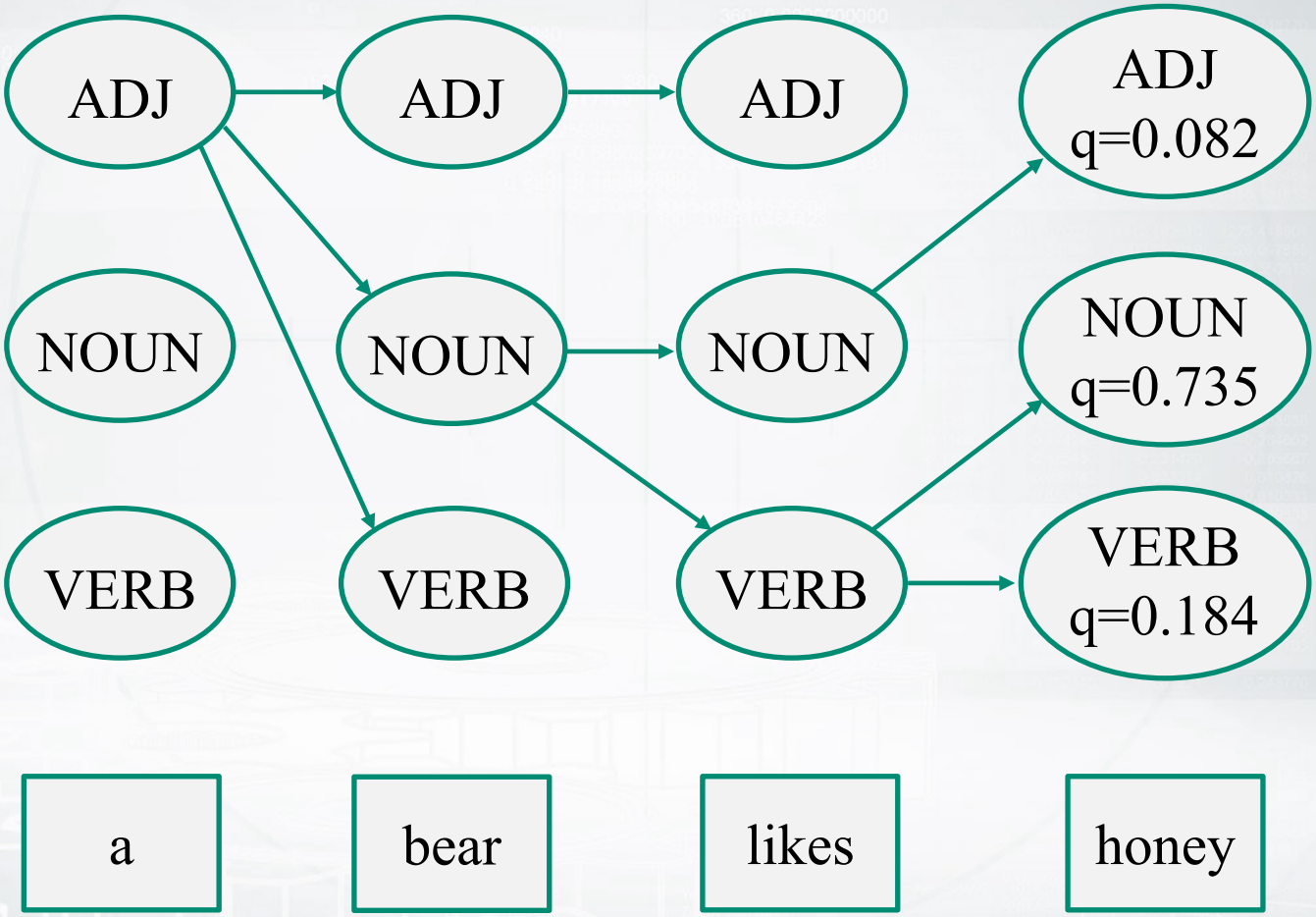
Probabilities of hidden states after “likes”



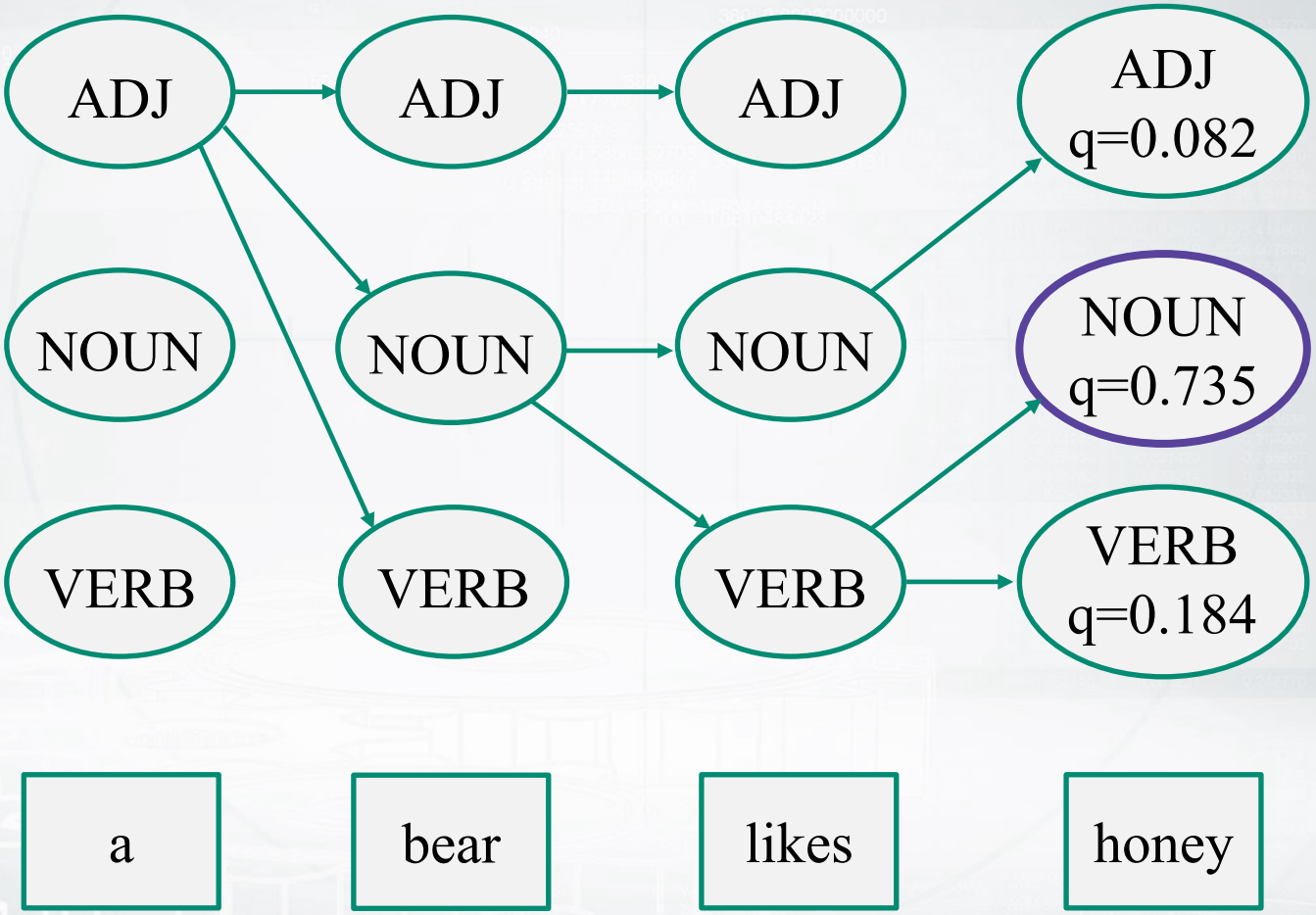
Remember the best transitions!



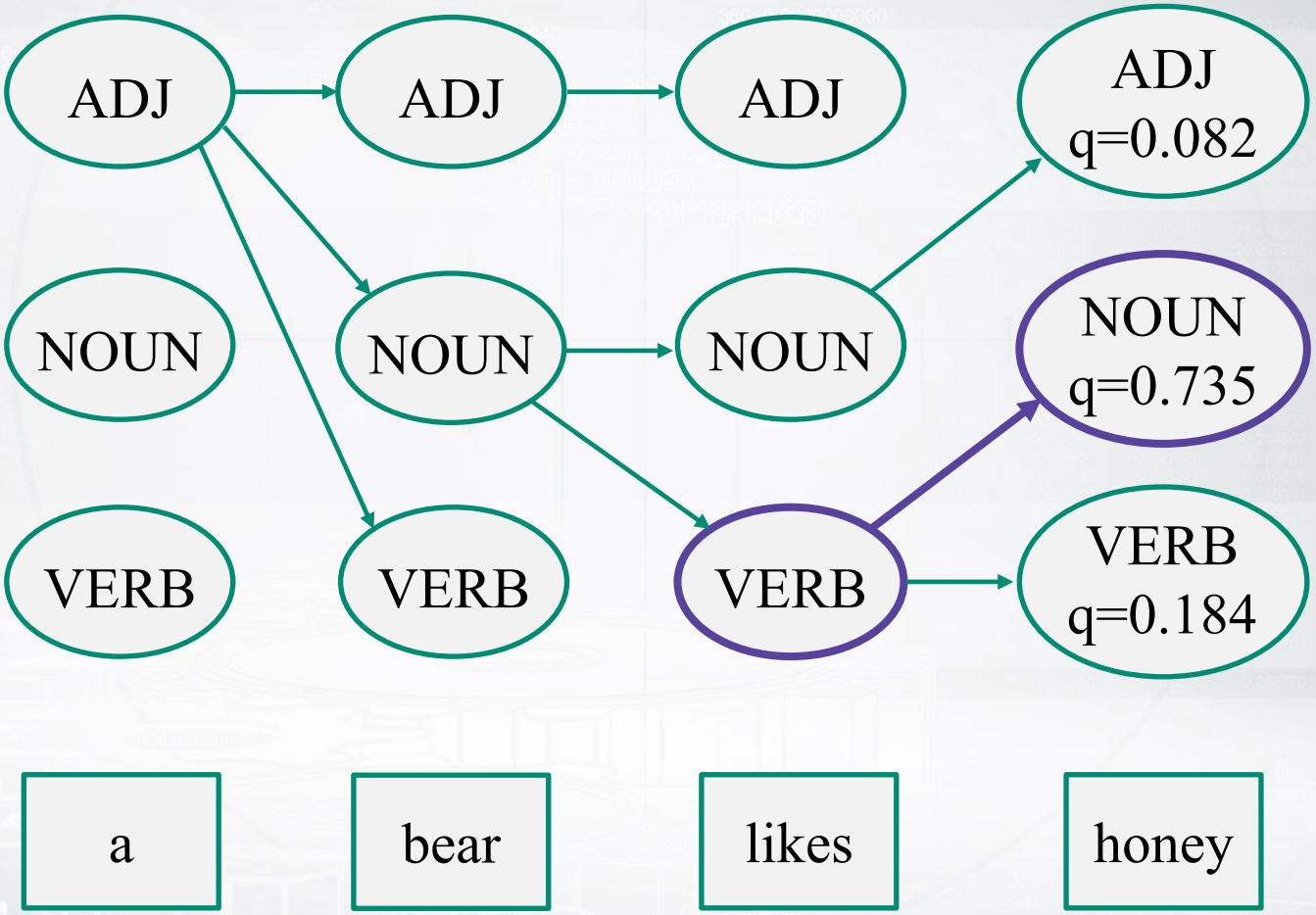
Backtrace


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

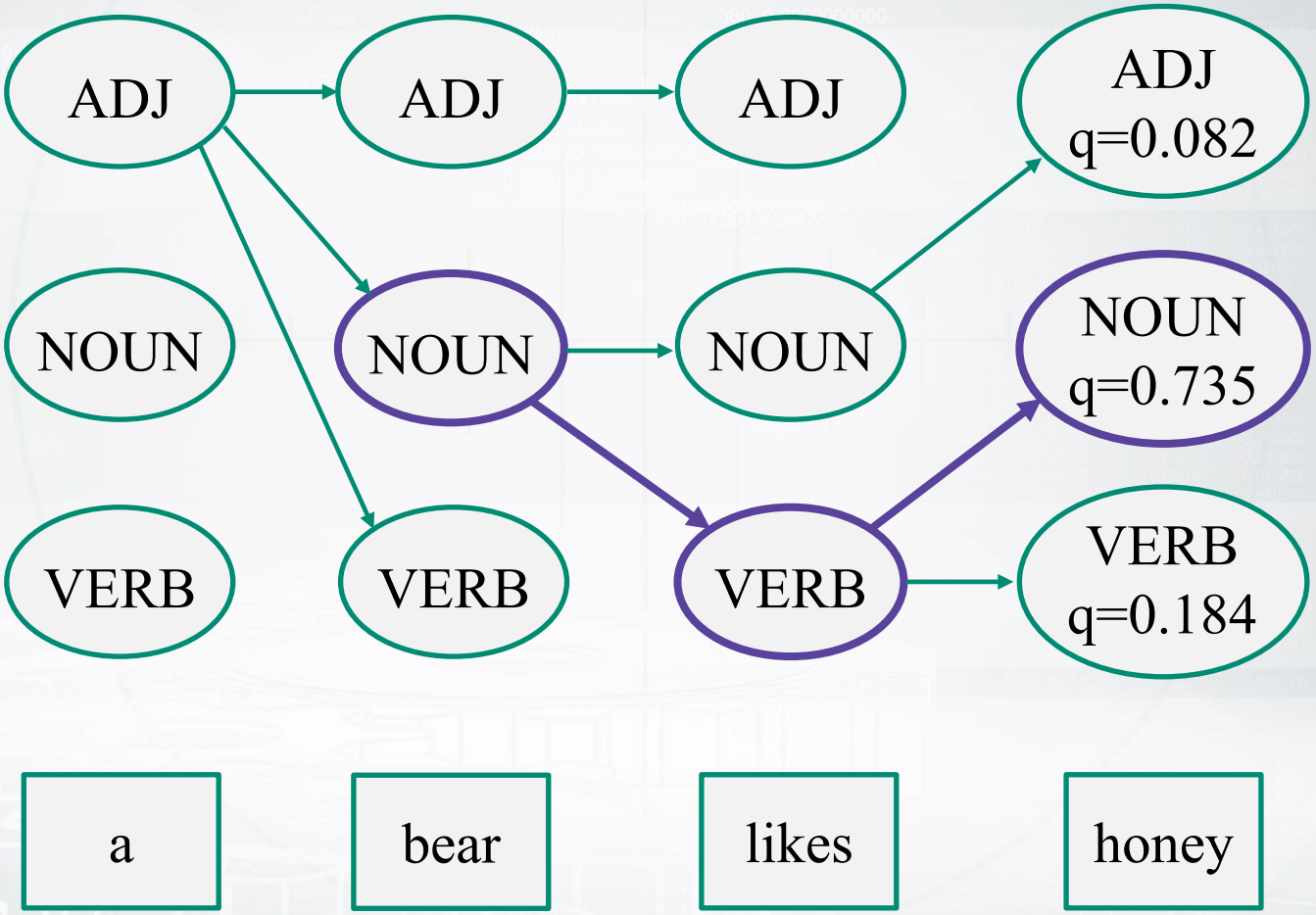
Backtrace


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

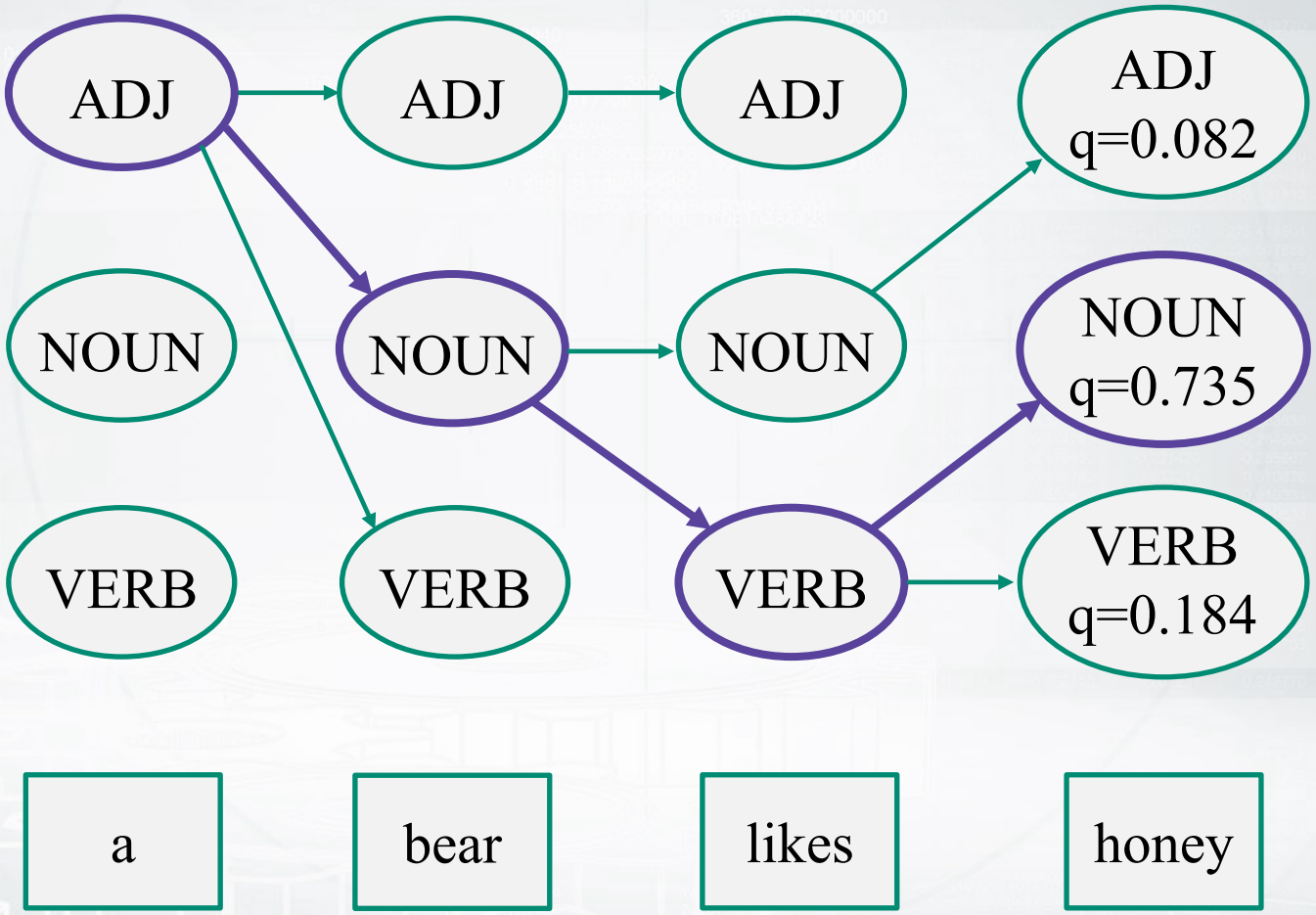
Backtrace


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

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 $[T, s]$