- **Today**
  - Part-of-speech tagging
    - HMM’s for p-o-s tagging

HMM p-o-s Tagger

Given $W = w_1, \ldots, w_n$, find $T = t_1, \ldots, t_n$ that maximizes

$$P(t_1, \ldots, t_n | w_1, \ldots, w_n)$$

Restate using Bayes' rule:

$$(P(t_1, \ldots, t_n) * P(w_1, \ldots, w_n | t_1, \ldots, t_n))/P(w_1, \ldots, w_n)$$

Ignore denominator...

Make independence assumptions...

$P(t_1, \ldots, t_n)$: approximate using **n-gram model**

bigram $\prod_{i=1,n} P(t_i \mid t_{i-1})$

trigram $\prod_{i=1,n} P(t_i \mid t_{i-2} t_{i-1})$

$P(w_1, \ldots, w_n \,|\, t_1, \ldots, t_n)$: approximate by assuming that a word appears in a category independent of its neighbors

$$\prod_{i=1,n} P(w_i \,|\, t_i)$$

Assuming bigram model:

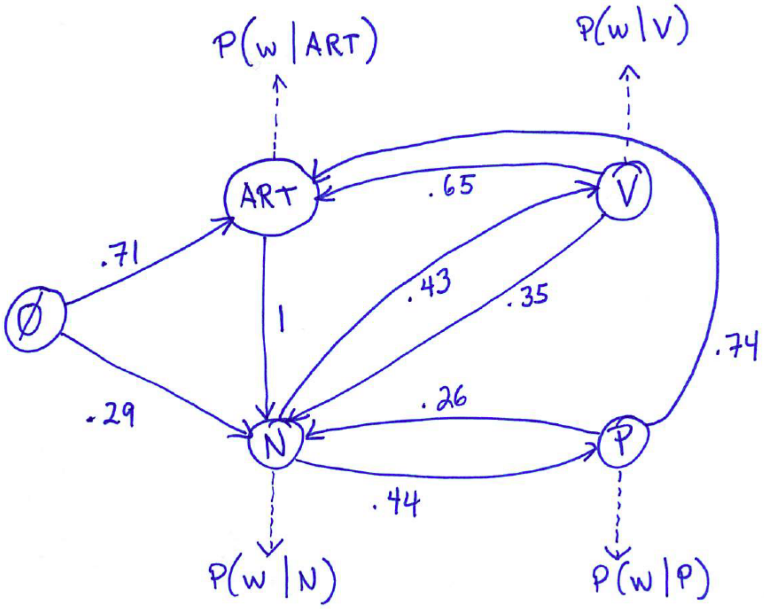
$$P(t_1, \ldots, t_n) * P(w_1, \ldots, w_n | t_1, \ldots, t_n) \approx$$

$$\prod_{i=1,n} \underset{\substack{\uparrow \\ \text{transition} \\ \text{probabilities}}}{P(t_i | t_{i-1})} * \underset{\substack{\nwarrow \\ \text{lexical generation} \\ \text{probabilities}}}{P(w_i | t_i)}$$

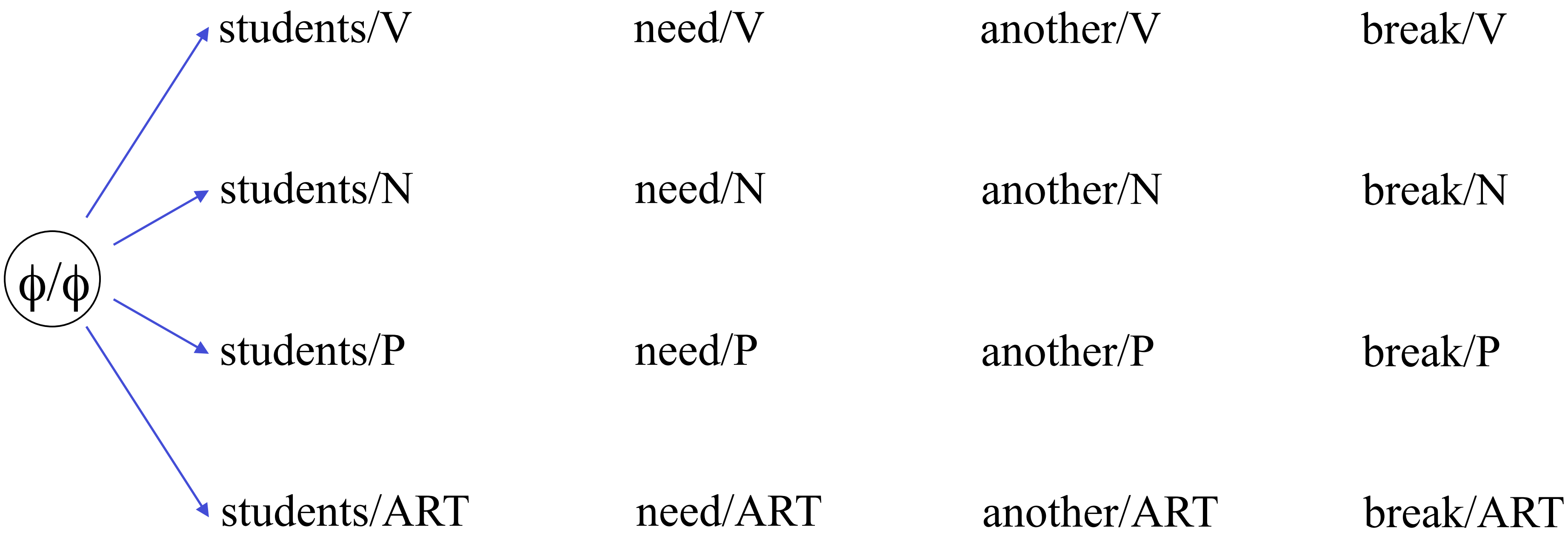
Hidden Markov Models

Equation can be modeled by an HMM.

- **states:** represent a possible lexical category
- **transition probabilities:** bigram probabilities
- **observation probabilities, lexical generation probabilities:** indicate, for each word, how likely that word is to be selected if we randomly select the category associated with the node.



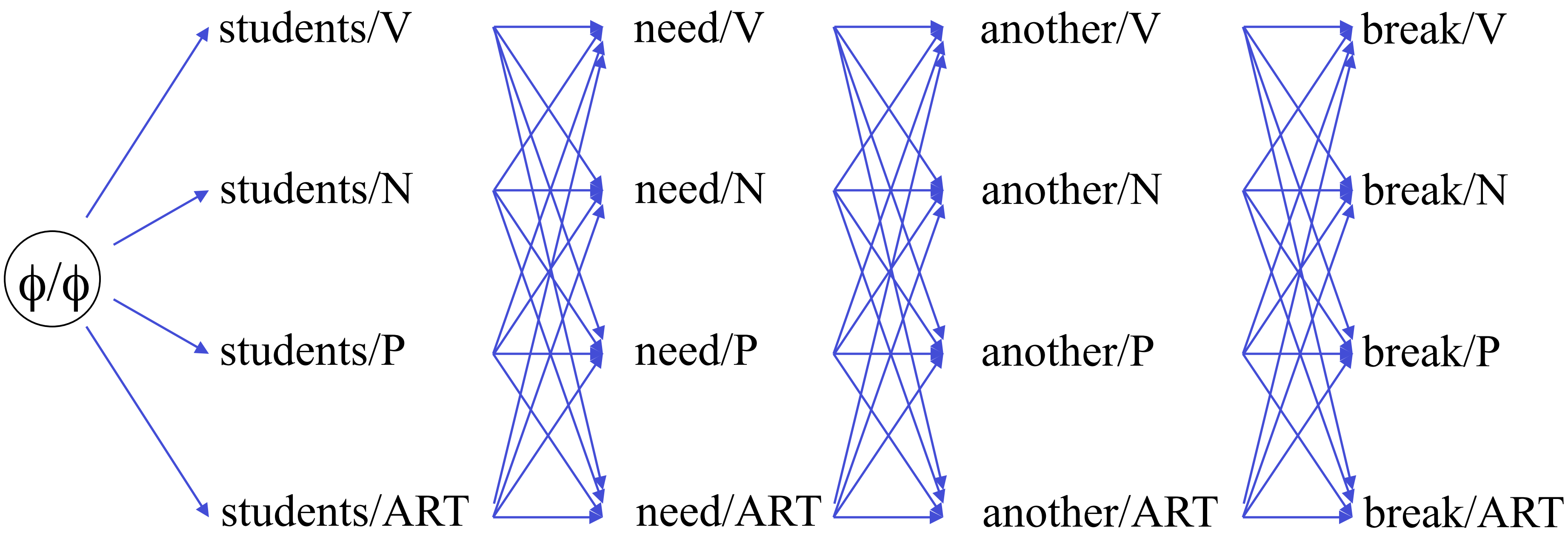
Viterbi Algorithm



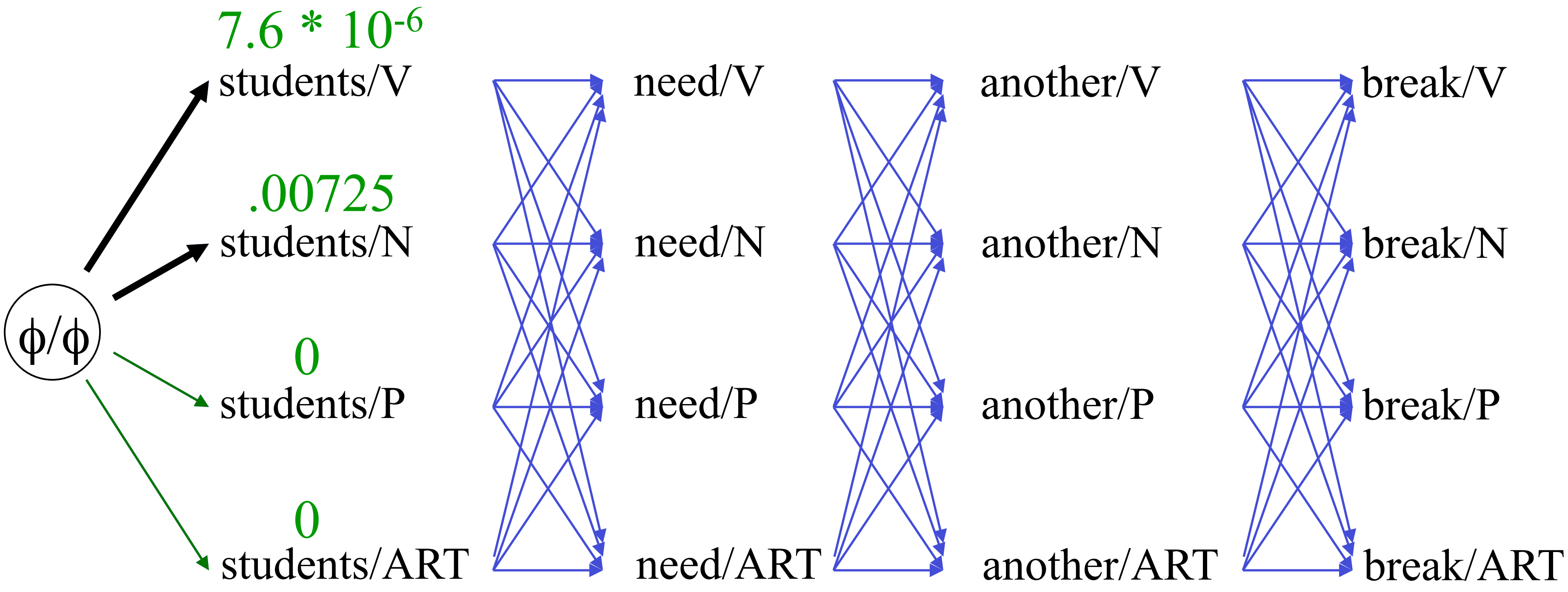
φ/φ

students/V	need/V	another/V	break/V
students/N	need/N	another/N	break/N
students/P	need/P	another/P	break/P
students/ART	need/ART	another/ART	break/ART

Viterbi Algorithm



Viterbi Algorithm



$7.6 * 10^{-6}$

students/V need/V another/V break/V

.00725

students/N need/N another/N break/N

ϕ/ϕ

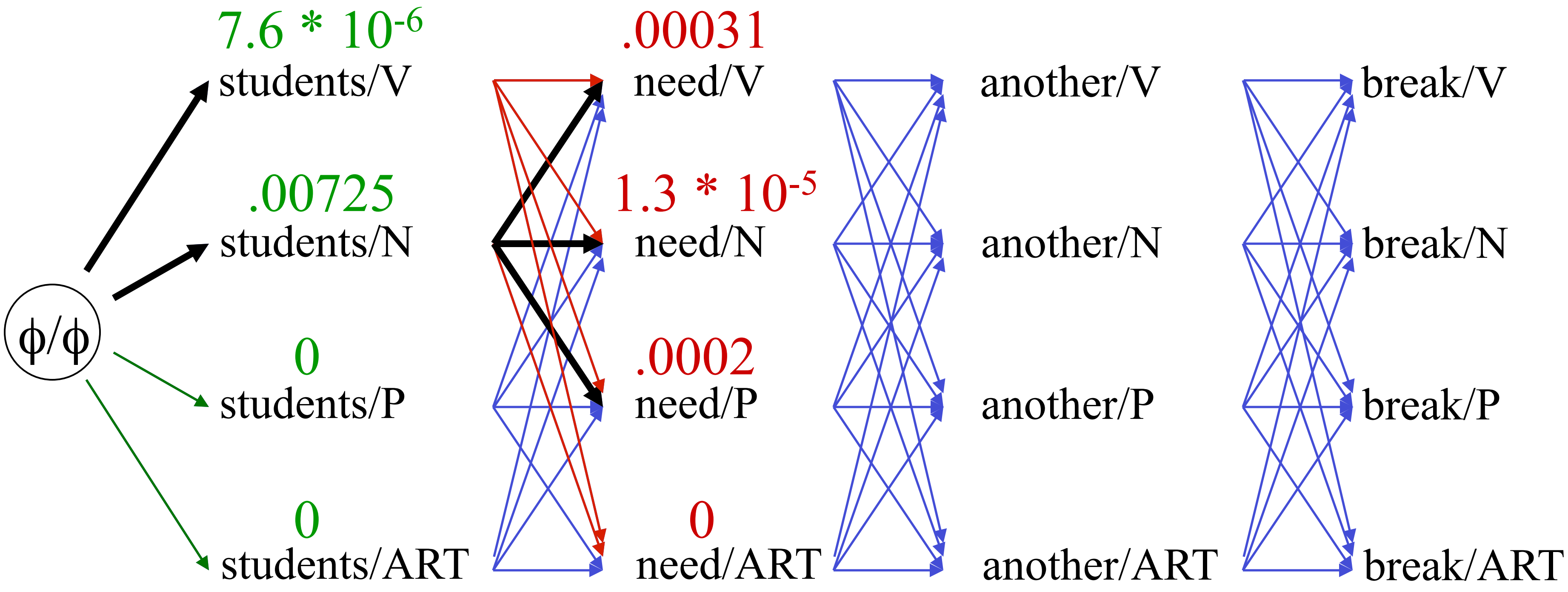
0

students/P need/P another/P break/P

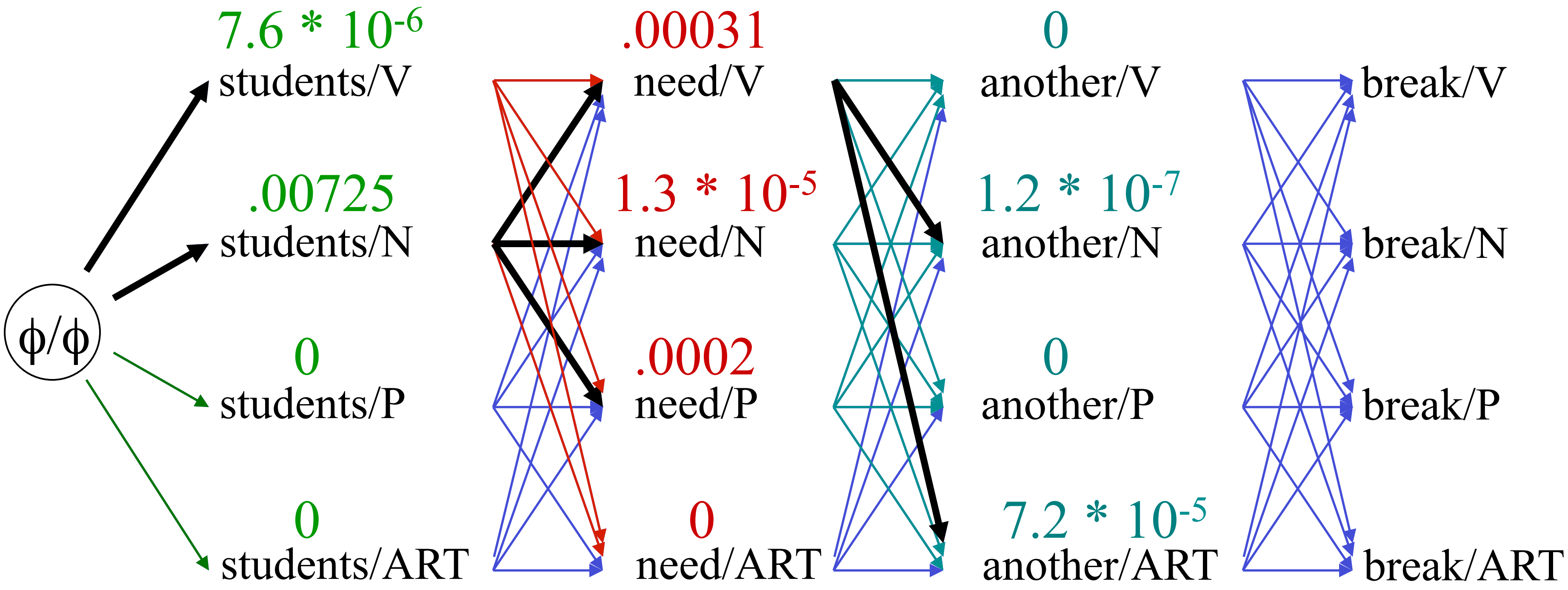
0

students/ART need/ART another/ART break/ART

Viterbi Algorithm



Viterbi Algorithm



$7.6 * 10^{-6}$ .00031 0

students/V need/V another/V break/V

.00725 $1.3 * 10^{-5}$ $1.2 * 10^{-7}$

students/N need/N another/N break/N

ϕ/ϕ

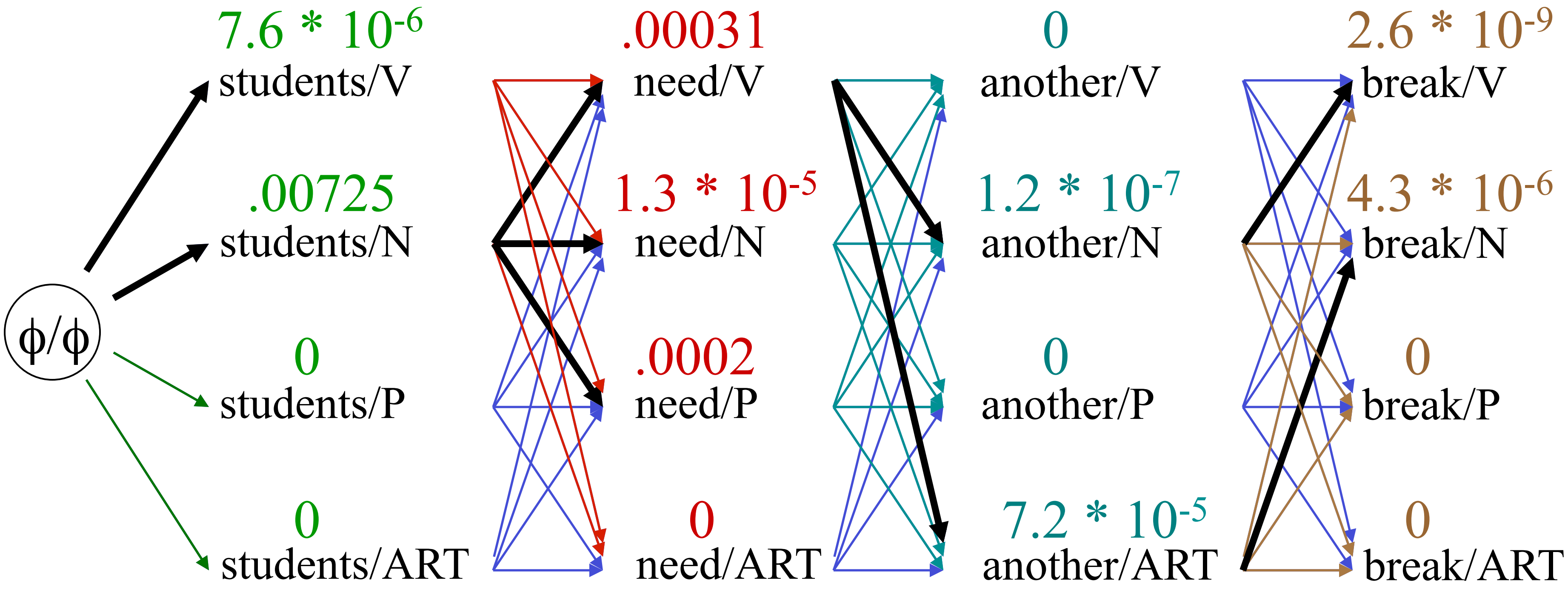
0 .0002 0

students/P need/P another/P break/P

0 0 $7.2 * 10^{-5}$

students/ART need/ART another/ART break/ART

Viterbi Algorithm



ϕ/ϕ

$7.6 * 10^{-6}$
students/V

.00725
students/N

0
students/P

0
students/ART

.00031
need/V

$1.3 * 10^{-5}$
need/N

.0002
need/P

0
need/ART

0
another/V

$1.2 * 10^{-7}$
another/N

0
another/P

$7.2 * 10^{-5}$
another/ART

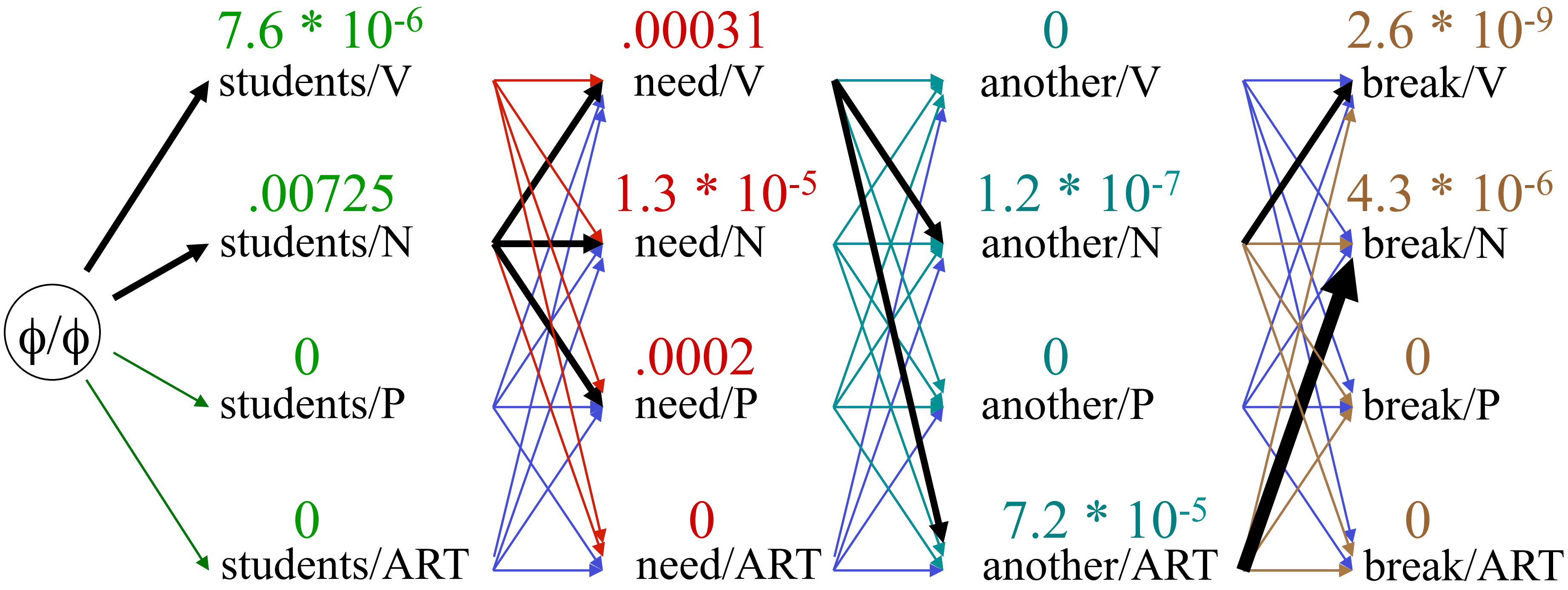
$2.6 * 10^{-9}$
break/V

$4.3 * 10^{-6}$
break/N

0
break/P

0
break/ART

Viterbi Algorithm



φ/φ

$7.6 * 10^{-6}$
students/V

.00725
students/N

0
students/P

0
students/ART

.00031
need/V

$1.3 * 10^{-5}$
need/N

.0002
need/P

0
need/ART

0
another/V

$1.2 * 10^{-7}$
another/N

0
another/P

$7.2 * 10^{-5}$
another/ART

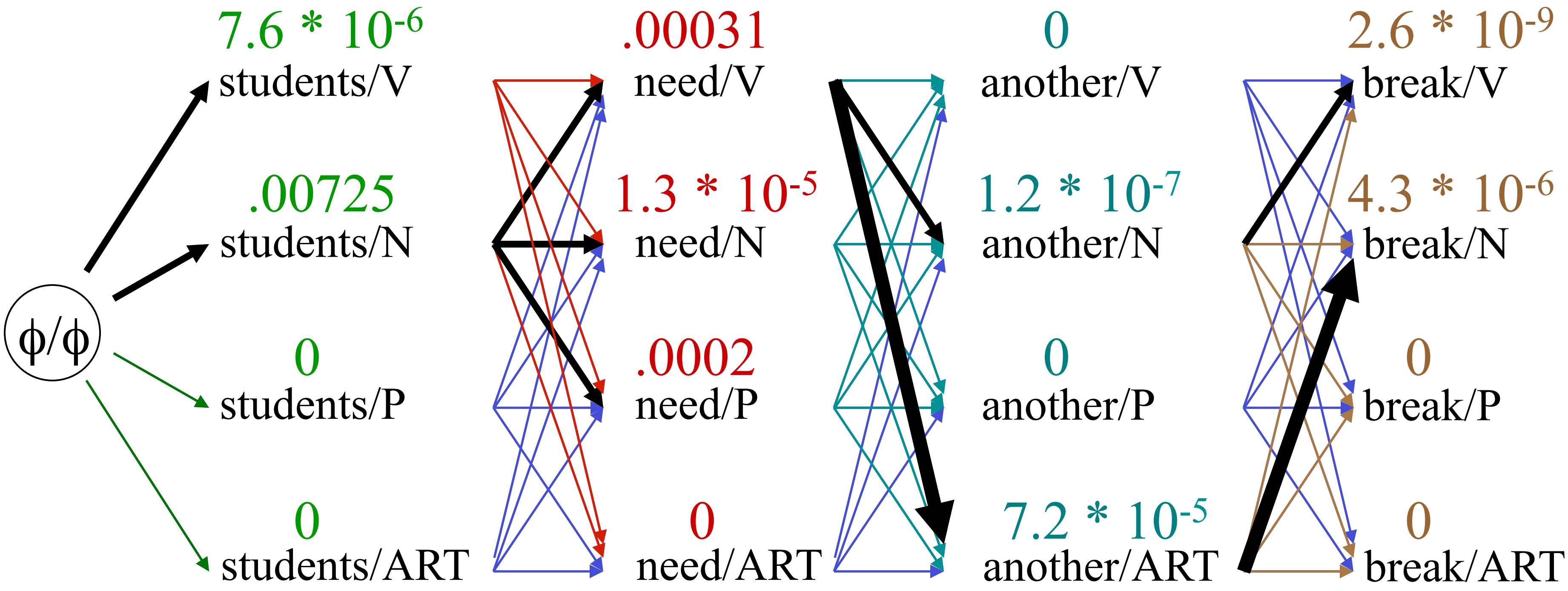
$2.6 * 10^{-9}$
break/V

$4.3 * 10^{-6}$
break/N

0
break/P

0
break/ART

Viterbi Algorithm



φ/φ

$7.6 * 10^{-6}$
students/V

$.00725$
students/N

0
students/P

0
students/ART

$.00031$
need/V

$1.3 * 10^{-5}$
need/N

$.0002$
need/P

0
need/ART

0
another/V

$1.2 * 10^{-7}$
another/N

0
another/P

$7.2 * 10^{-5}$
another/ART

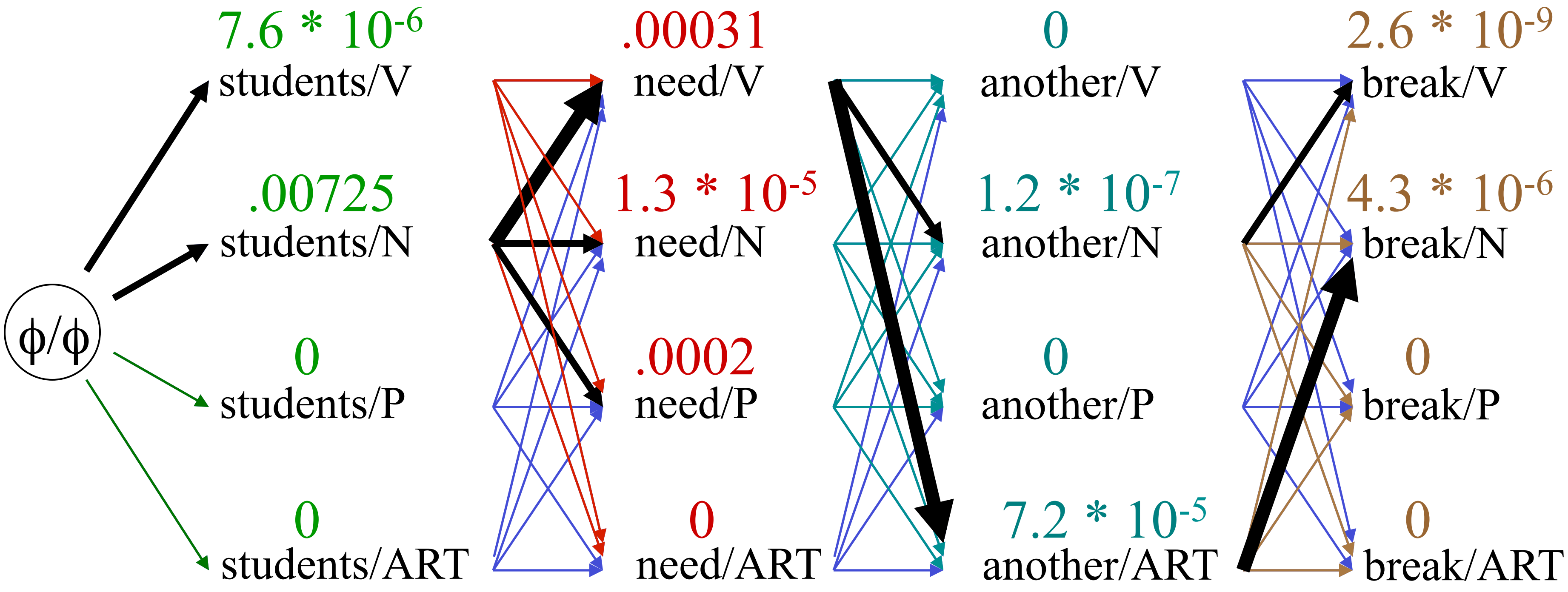
$2.6 * 10^{-9}$
break/V

$4.3 * 10^{-6}$
break/N

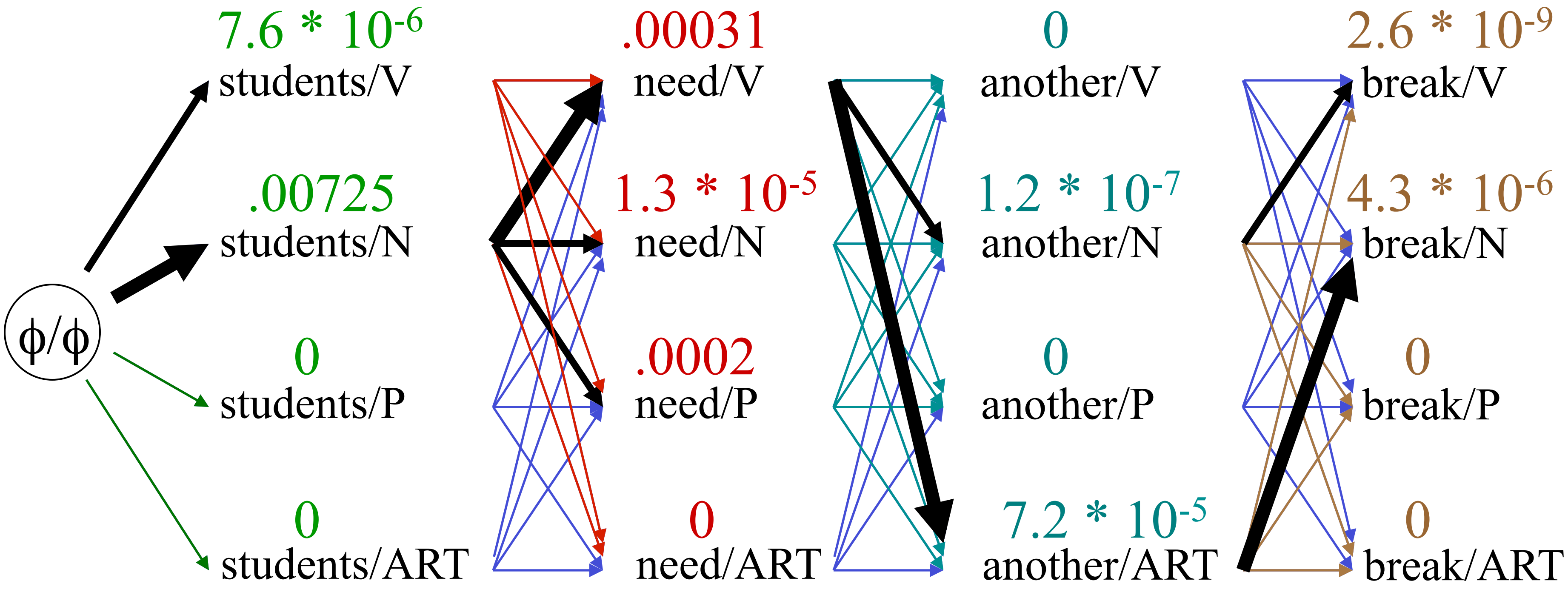
0
break/P

0
break/ART

Viterbi Algorithm



Viterbi Algorithm



ϕ/ϕ

$7.6 * 10^{-6}$
students/V

$.00725$
students/N

0
students/P

0
students/ART

$.00031$
need/V

$1.3 * 10^{-5}$
need/N

$.0002$
need/P

0
need/ART

0
another/V

$1.2 * 10^{-7}$
another/N

0
another/P

$7.2 * 10^{-5}$
another/ART

$2.6 * 10^{-9}$
break/V

$4.3 * 10^{-6}$
break/N

0
break/P

0
break/ART

Viterbi Algorithm

c: number of lexical categories

$P(w_t|t_i)$: lexical generation probabilities

$P(t_i|t_j)$: bigram probabilities

Find most likely sequence of lexical categories $T_1, \ldots, T_n$ for word sequence.

Initialization

```
For i = 1 to c do
  SCORE(i,1) = $P(t_i|\phi) * P(w_1|t_i)$
  BPTR(i,1) = 0
```

Iteration

```
For t = 2 to n
  For i = 1 to c
    SCORE(i,t) =
      $MAX_{j=1..c}(SCORE(j,t-1) * P(t_i|t_j)) * P(w_t|t_i)$
    BPTR(i,t) = index of j that gave max
```

Identify Sequence

```
T(n) = i that maximizes SCORE(i,n)
For i = n-1 to 1 do
  T(i) = BPTR( T(i+1), i+1 )
```

Results

- Effective if probability estmates are computed from a large corpus
- Effective if corpus is of the same style as the input to be classified
- Consistently achieve accuracies of 97% or better using trigram model
- Cuts error rate in half vs. naive algorithm (90% accuracy rate)
- Can be smoothed using backoff or interpolation or discounting...

Extensions

- Can train HMM tagger on unlabeled data using the EM algorithm, starting with a dictionary that lists which tags can be assigned to which words.
- EM then learns the word likelihood function for each tag, and the tag transition probabilities.
- Merialdo (1994) showed, however, that a tagger trained on even a small amount hand-tagged data works better than one trained via EM.

Marseille at Newman Overlook