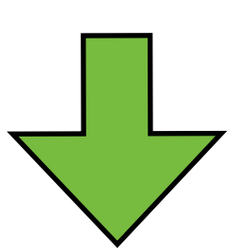
Unsupervised Learning on an Approximate Corpus

Jason Smith, Jason Eisner

Learning from n-grams

Sentences:

					Counts
time	flies	like	an	arrow	20
fruit	flies	like	an	orange	3
your plane	flies	like	an	ostrich	2



n-grams:

time	flies	like			20
fruit	flies	like			3
	flies	like	an		25
		like	an	arrow	20
your plane	flies				2

...

Contributions

- Learning from a finite-state distribution over sentences
  - e.g. an n-gram language model over sentences, instead of individual sentences
- Why?
  - Original corpus unavailable
  - Speed (learning from compressed data)
  - (Fundamental question about weighted grammars)
- Exact and approximate solutions

Task

- HMM POS tagging (Merialdo 94)
- Many approaches build off of EM

Previous Work

Gender

- Count the possessive pronouns following a word after connectives or verbs. [Bergsma 2005]
- Pattern: Word CC/V* PRP$

		its	his	her	their	my
	star and ___	2805	2409	1594	250	247
Hollywood	star and ___	0	29	13	0	0
variable	star and ___	57	0	0	0	0

- (slide taken from Lin et al., 2009)

Previous Work

Gender

- Count the possessive pronouns following a word after connectives or verbs. [Bergsma 2005]
- Pattern: Word CC/V* PRP$

	its	his	her	their	my
star and ___	2805	2409	1594	250	247
Hollywood star and ___	0	29	13	0	0
variable star and ___	57	0	0	0	0

- (slide taken from Lin et al., 2009)

Previous Work

Gender

- Count the possessive pronouns following a word after connectives or verbs. [Bergsma 2005]
- Pattern: Word CC/V* PRP$

	its	his	her	their	my
star and ___	2805	2409	1594	250	247
Hollywood star and ___	0	29	13	0	0
variable star and ___	57	0	0	0	0

- (slide taken from Lin et al., 2009)

Motivation: learning from n-grams

Full context:

N	V	Adv	Det	N
time	flies	like	an	arrow
Adj	N	V	Det	N
fruit	flies	like	an	orange

Motivation: learning from n-grams

Full context:

N	V	Adv	Det	N
time	flies	like	an	arrow
Adj	N	V	Det	N
fruit	flies	like	an	orange

Local n-gram context:

?	V? N?	Adv? V?	Det	?
?	flies	like	an	?

Motivation: learning from n-grams

Local n-gram context:

?	V? N?	Adv? V?	Det	?
?	flies	like	an	?

Overlapping n-grams:

			Counts
time	flies	like	20
fruit	flies	like	3
plane	flies	like	2

Motivation: learning from n-grams

Local n-gram context:

?	V? N?	Adv? V?	Det	?
time 80% fruit 12% plane 8%	flies	like	an	?

Overlapping n-grams:

			Counts
time	flies	like	20
fruit	flies	like	3
plane	flies	like	2

Motivation: learning from n-grams

Local n-gram context:

N	V	Adv	Det	?
Adj	N	V		
time 80%				
fruit 12%	flies	like	an	?
plane 8%				

Overlapping n-grams:

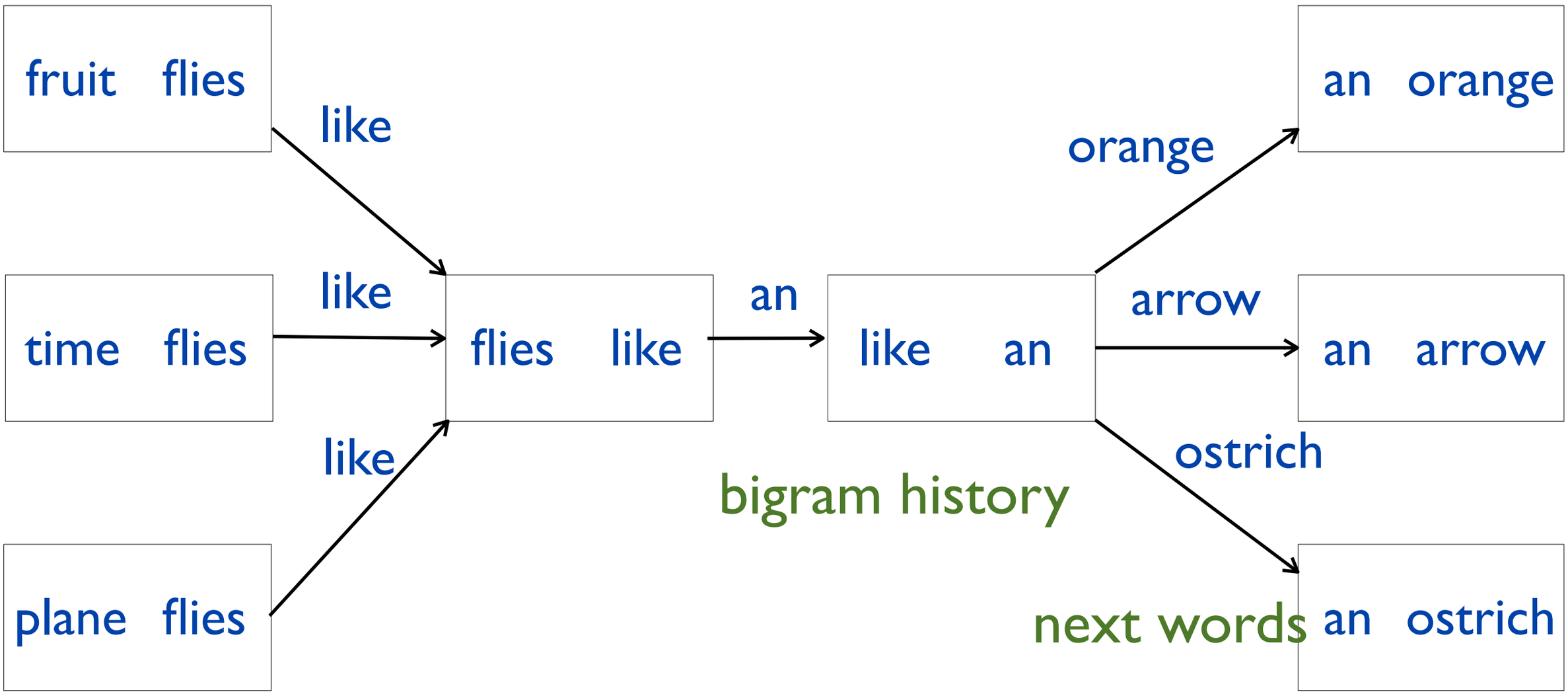
			Counts
time	flies	like	20
fruit	flies	like	3
plane	flies	like	2

Exploit Overlapping n-grams

					Counts
time	flies	like			20
	flies	like	an		25
		like	an	arrow	20
	fruit	flies	like		3
		like	an	orange	3
your	plane	flies			2
	plane	flies	like		2
		like	an	ostrich	2

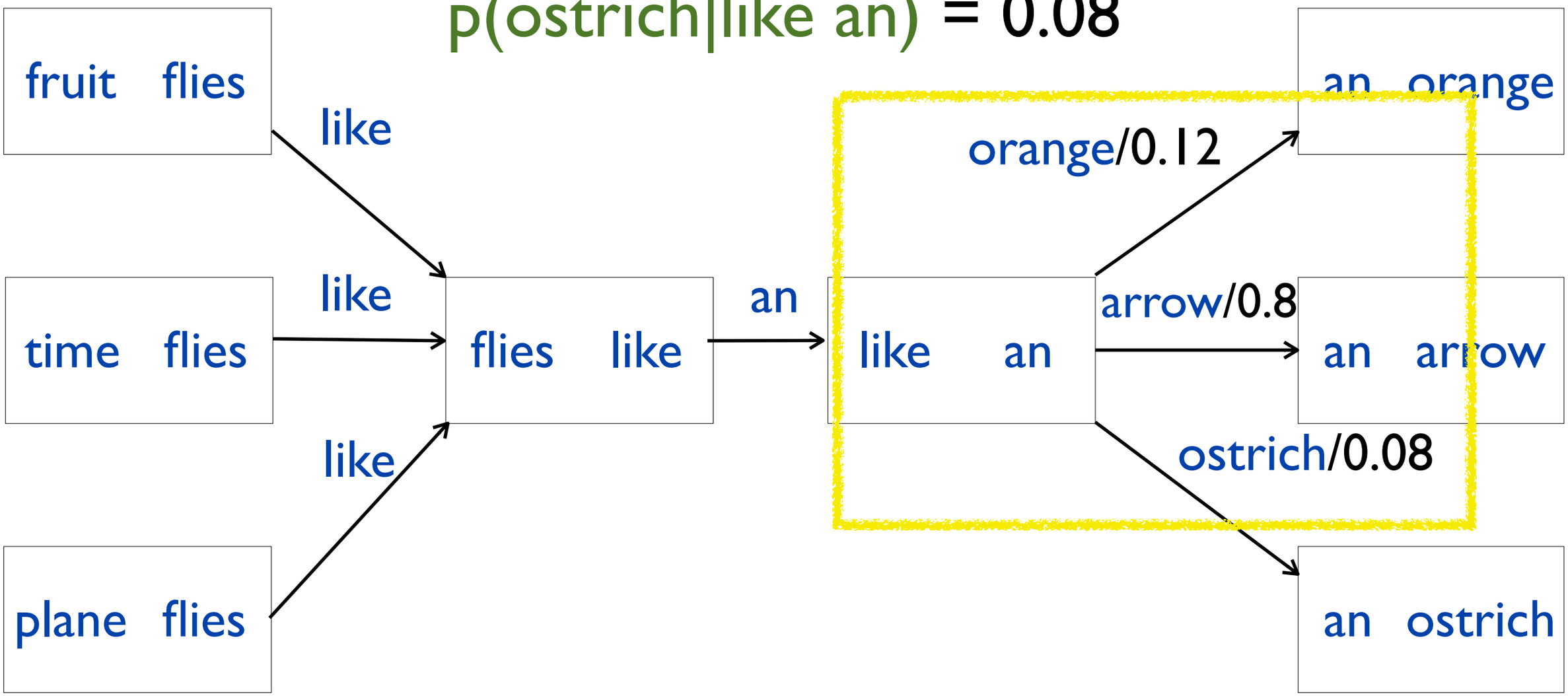
Exploit Overlapping n-grams

n-gram language model!



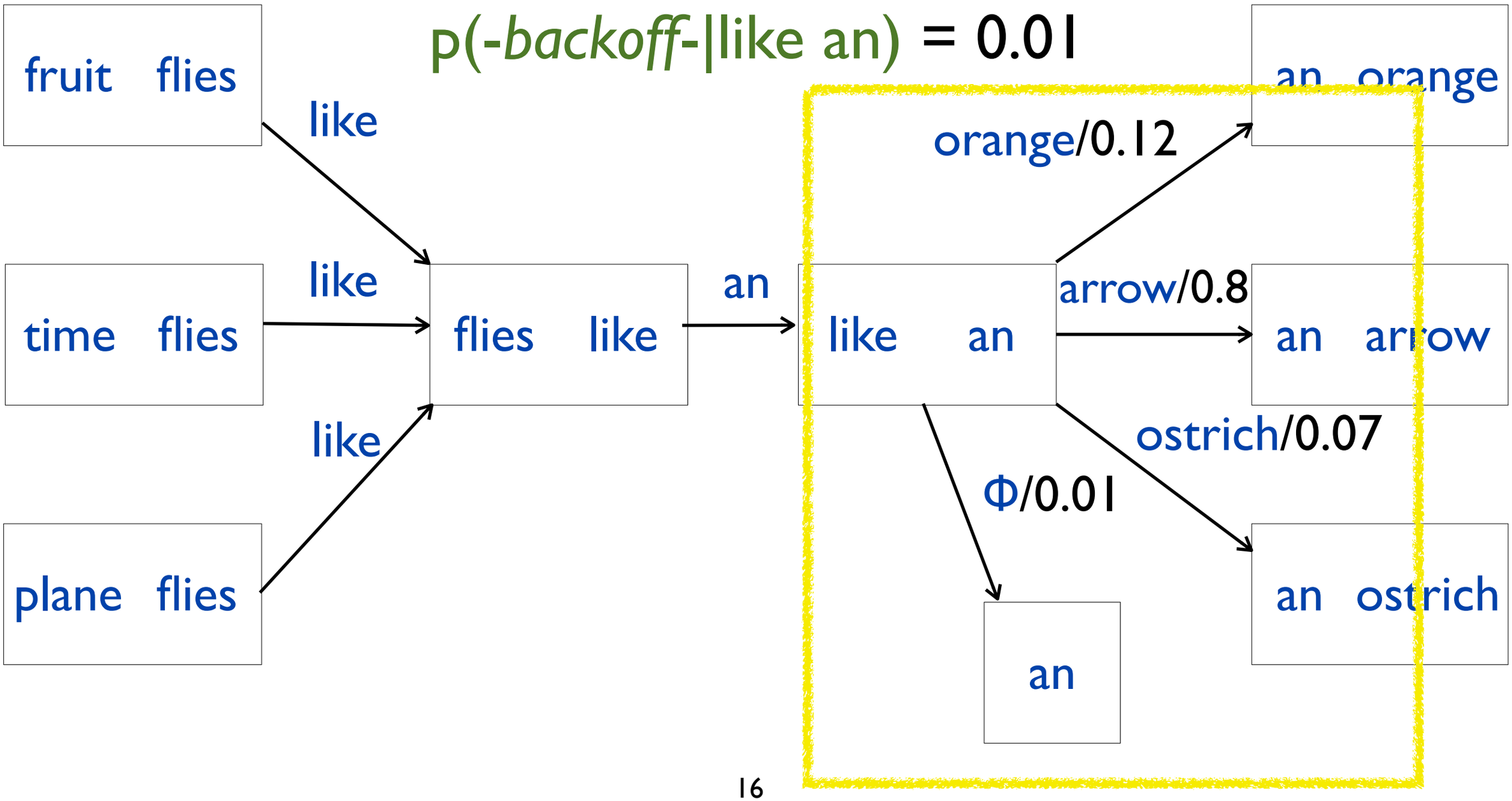
N-gram language models

p(orange|like an) = 0.12
p(arrow|like an) = 0.80
p(ostrich|like an) = 0.08

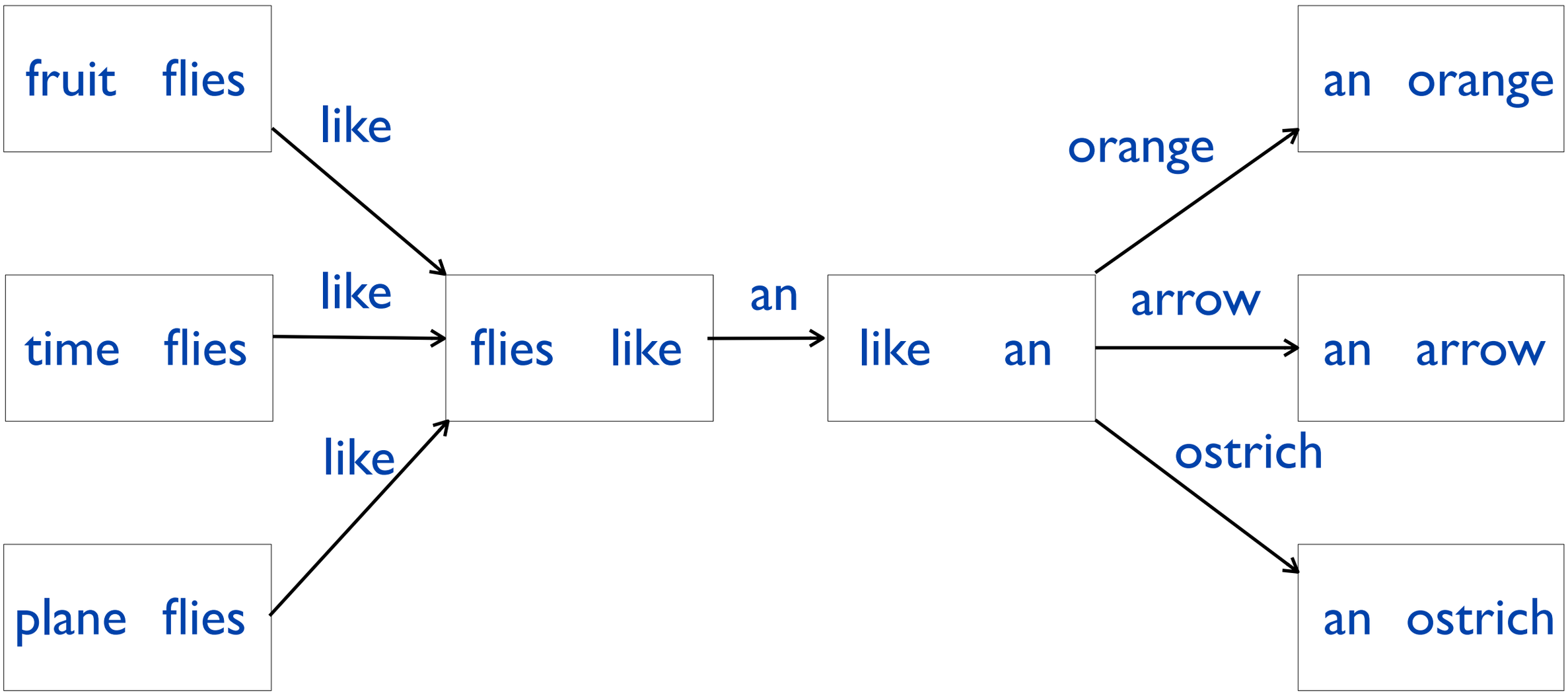


N-gram language models

p(orange|like an) = 0.12
p(arrow|like an) = 0.80
p(ostrich|like an) = 0.07
p(-*backoff*-|like an) = 0.01



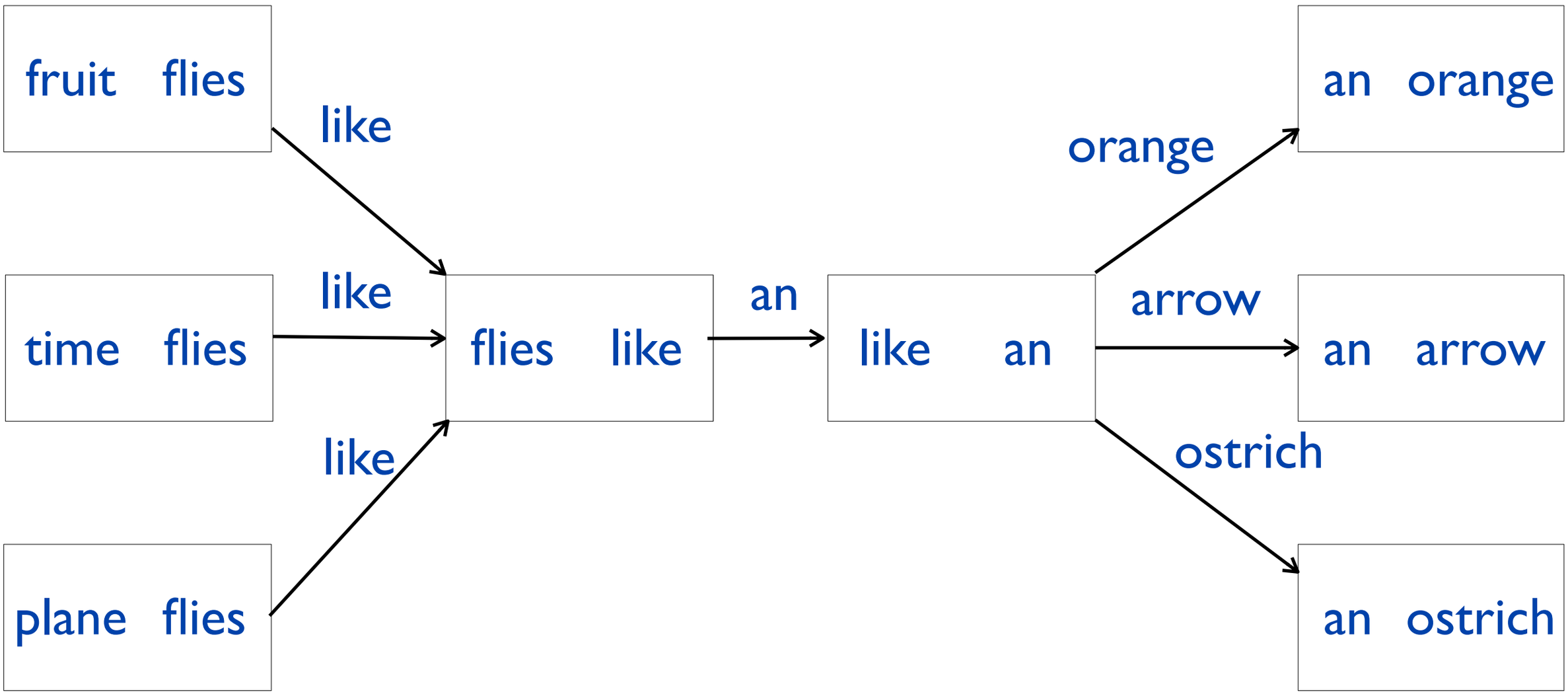
N-gram language models



N-gram language models

c(w):

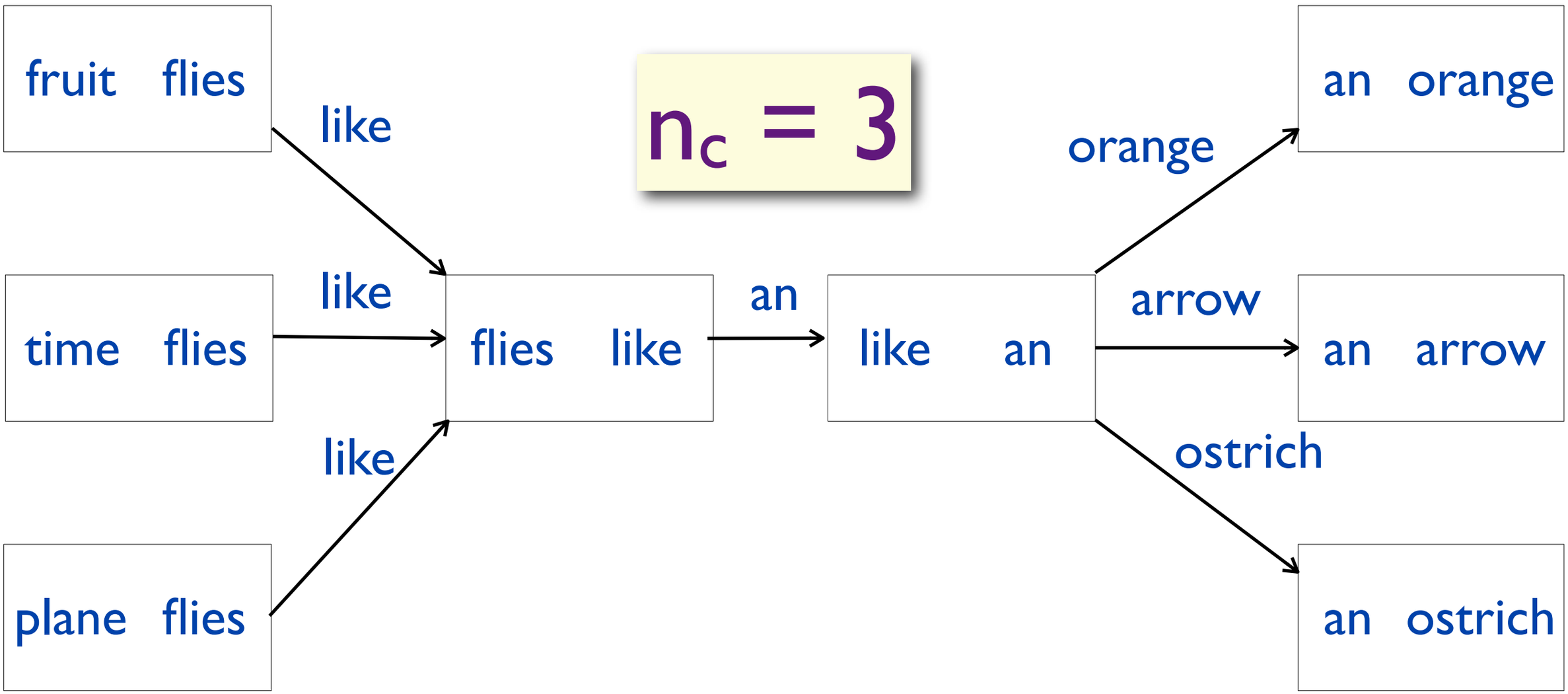
- probability distribution over sentences
- "approximate corpus"



N-gram language models

c(w):

- probability distribution over sentences
- "approximate corpus"



$n_c = 3$

Task

- HMM POS tagging (Merialdo 94)
- Many approaches build off of EM

$$\max \sum_{w \in \text{corpus}} \frac{1}{n} \log \sum_{t} p(t, w)$$

$$\max \sum_{w \in \text{corpus}} c(w) \log \sum_{t} p(t, w)$$

HMM Tagging

Sentence: time flies like an arrow

HMM Tagging

Sentence: time flies like an arrow

p(Tag|Last Tag)

	Det	N	V	...
Det	0.1	0.1	0.5	
N	0.8	0.3	0.4	...
V	0.1	0.6	0.1	
...		...		

HMM Tagging

Sentence: time flies like an arrow

p(Tag|Last Tag)

	Det	N	V	...
Det	0.1	0.1	0.5	
N	0.8	0.3	0.4	...
V	0.1	0.6	0.1	
...	...			

p(Word|Tag)

	Det	N	V	...
time	0.01	0.3	0.1	
flies	0.01	0.2	0.2	...
an	0.33	0.01	0.01	
...	...			

HMM Tagging

Sentence: time flies like an arrow

p(Tag|Last Tag)

	Det	N	
Det	0.1	0.1	0.3

p(Word|Tag)

		N	V	...
time	0.01	0.3	0.1	

$p(t,w) : n_p = 2$

n_c : c(w)'s word context window

n_p : p(t,w)'s tag context window

Supervised learning: HMM

N	V	Adv	Det	N
time	flies	like	an	arrow

Supervised learning: HMM

transition counts: estimating p(Tag|Last Tag)

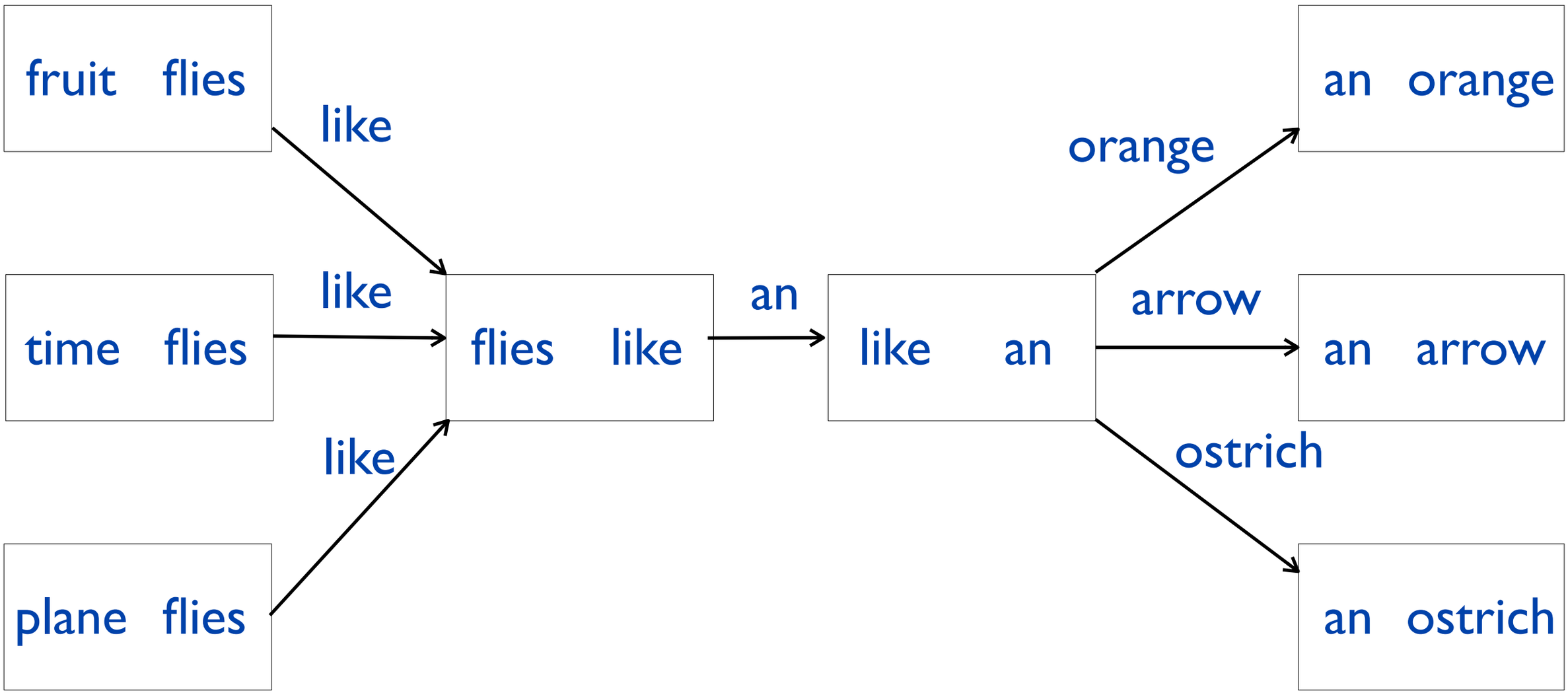
Supervised learning: HMM

emission counts: estimating p(Word|Tag)

N	V	Adv	Det	N
time	flies	like	an	arrow

What if someone tagged our n-grams?

c(w):

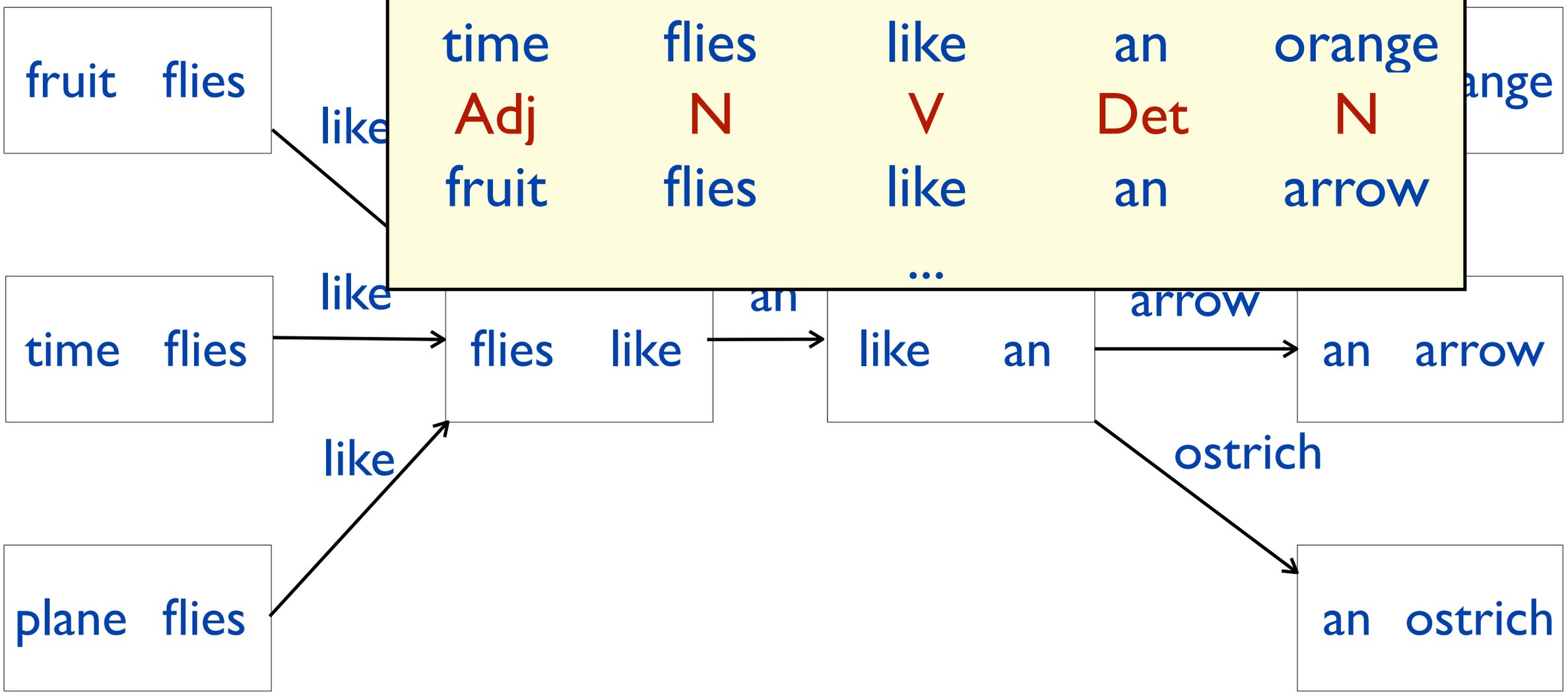


What if someone tagged our n-grams?

c(w):

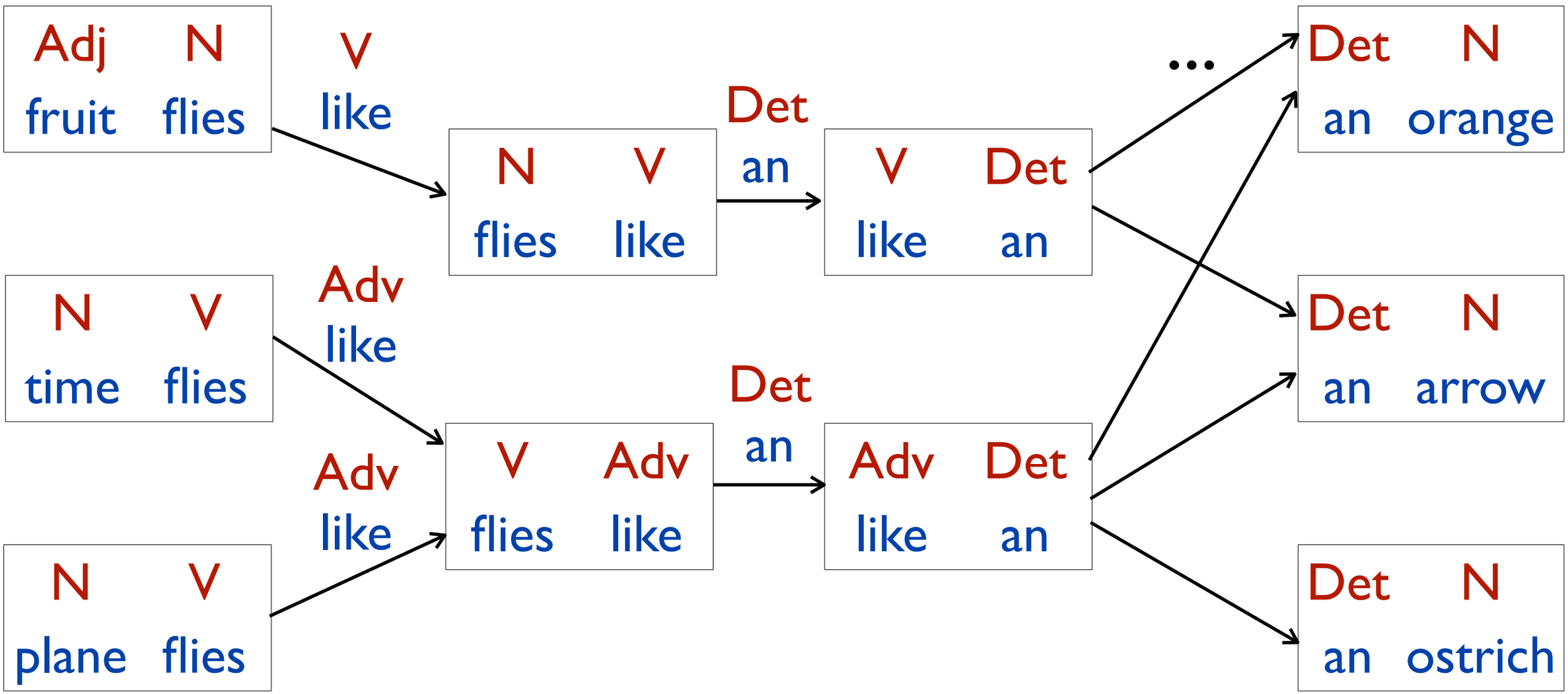
N	V	Adv	Det	N
time	flies	like	an	orange
Adj	N	V	Det	N
fruit	flies	like	an	arrow

...



What if someone tagged our n-grams?

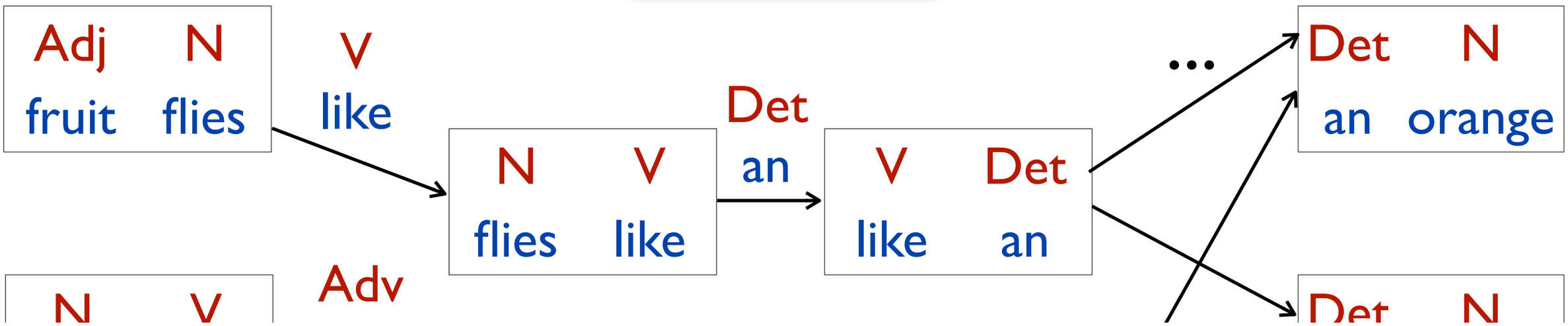
c(~~w~~) c(t,w):



What if someone tagged our n-grams?

c(~~w~~) c(t,w):

$n_q = 3$



n_c : c(w)'s word context window

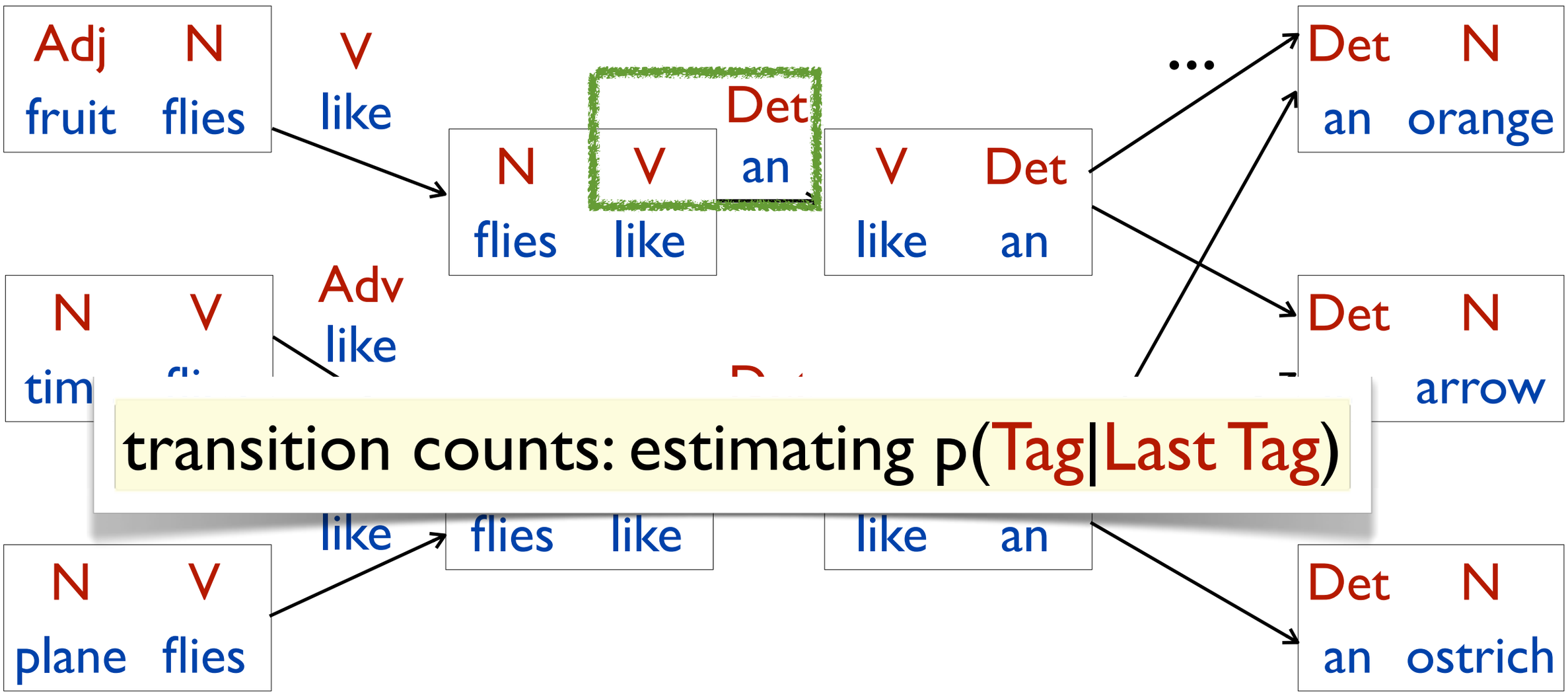
n_p : p(t,w)'s tag context window

n_q : c(t,w)'s tag context window

What if someone tagged our n-grams?

$n_q = 3$

c(~~w~~) c(t,w):

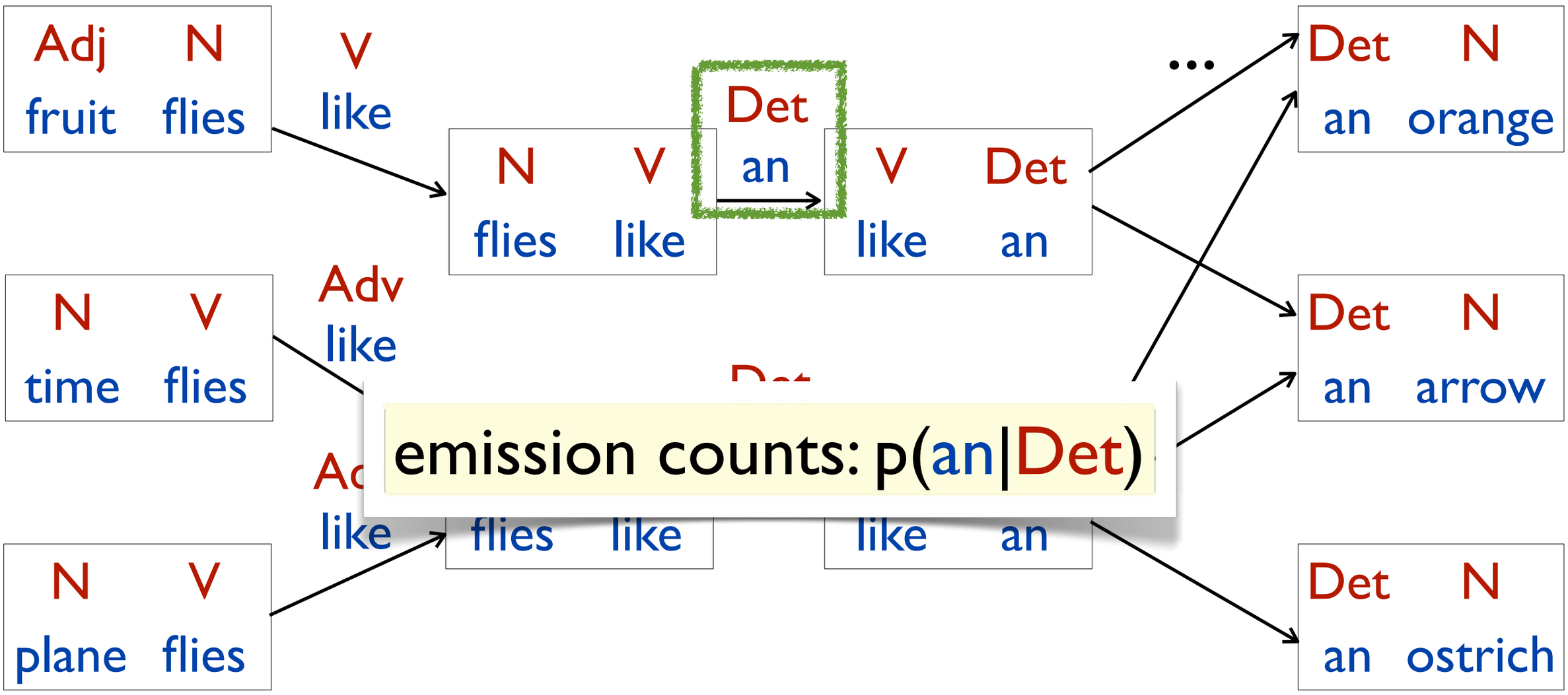


transition counts: estimating p(Tag|Last Tag)

What if someone tagged our n-grams?

$n_q = 3$

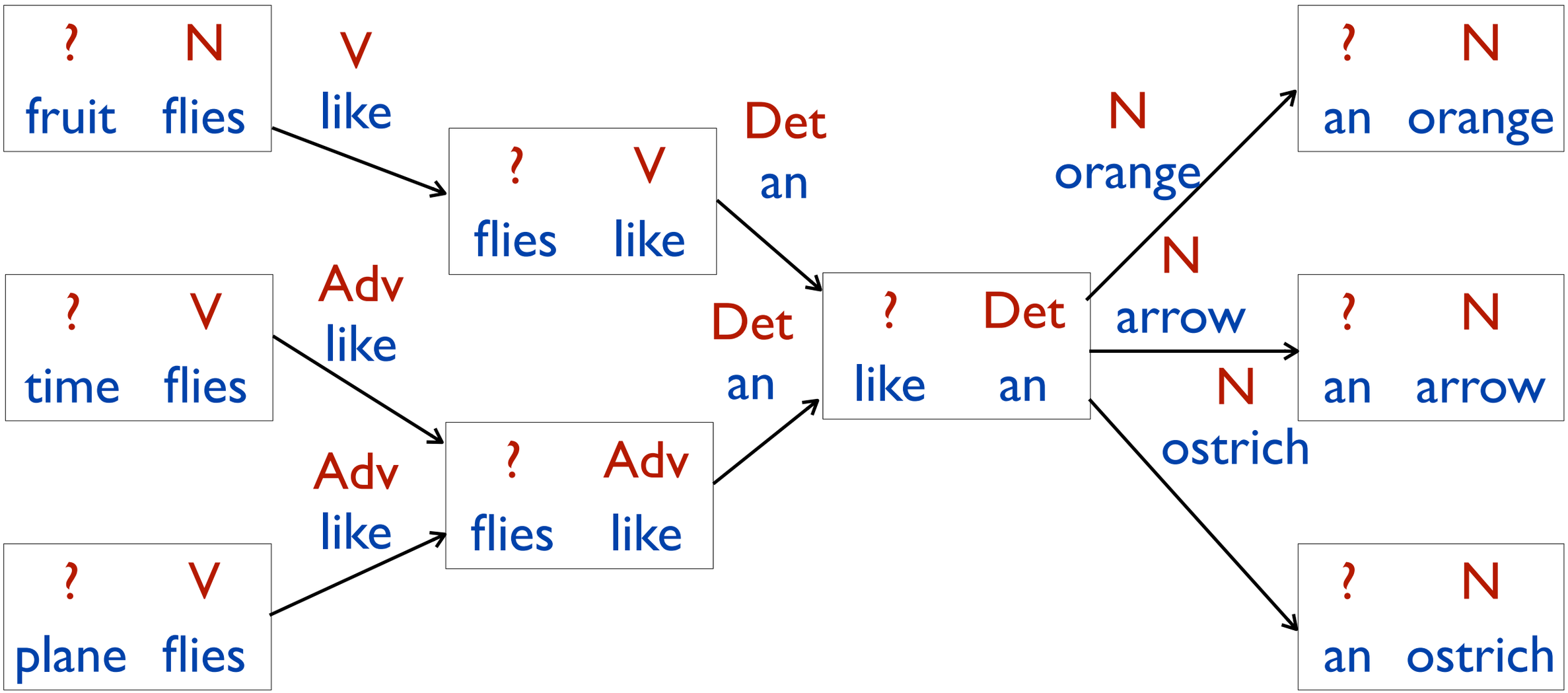
~~c(w)~~ c(t,w):



What if someone tagged our n-grams?

$n_q = 2$

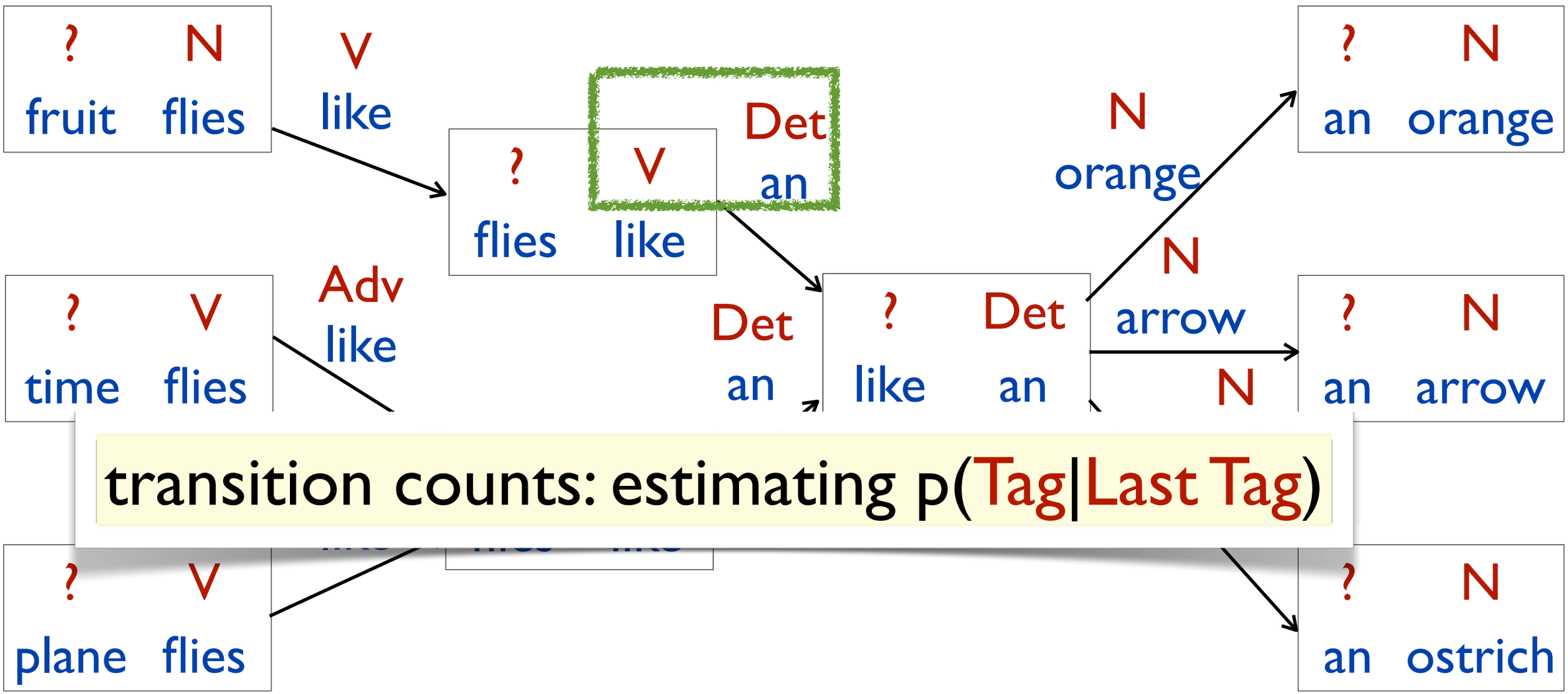
c(~~w~~) c(t,w):



What if someone tagged our n-grams?

$n_q = 2$

c(~~w~~) c(t,w):

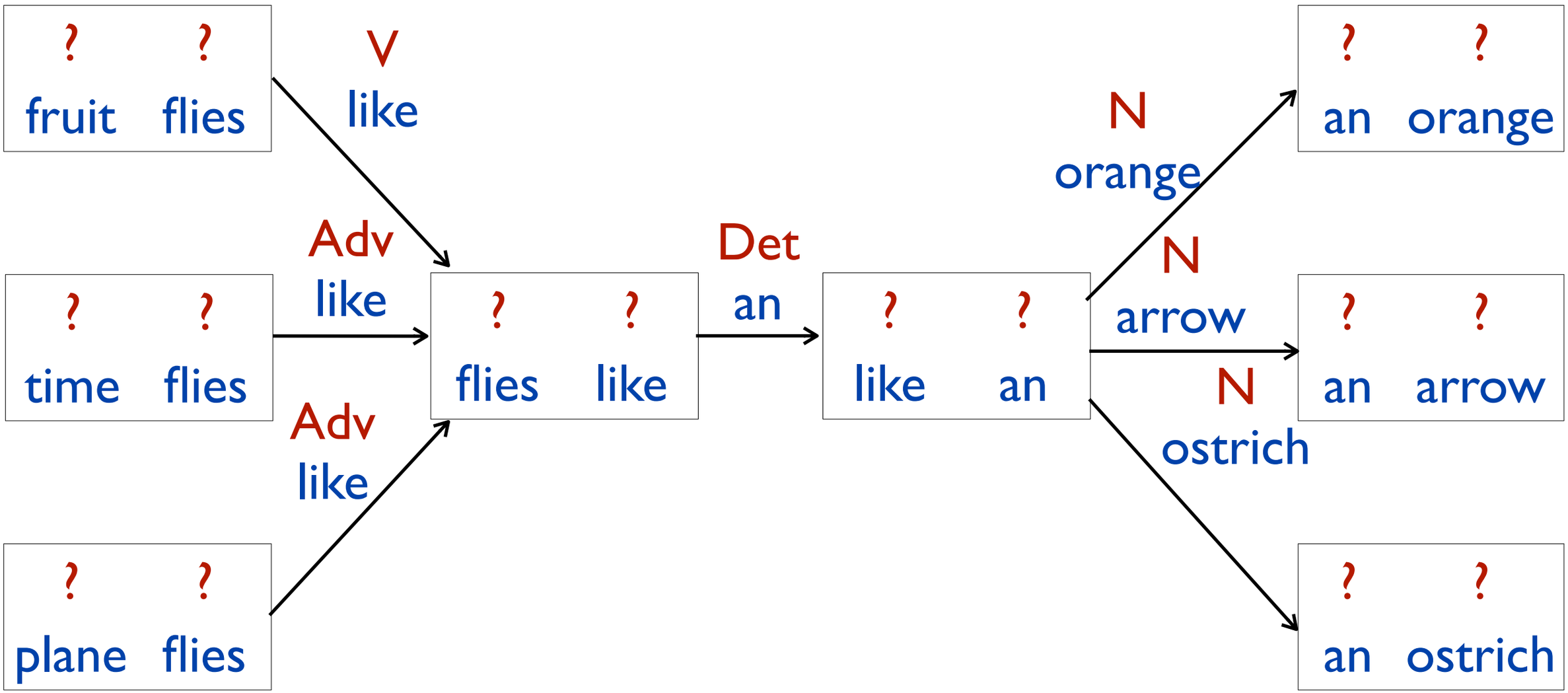


transition counts: estimating p(Tag|Last Tag)

What if someone tagged our n-grams?

$n_q = 1$

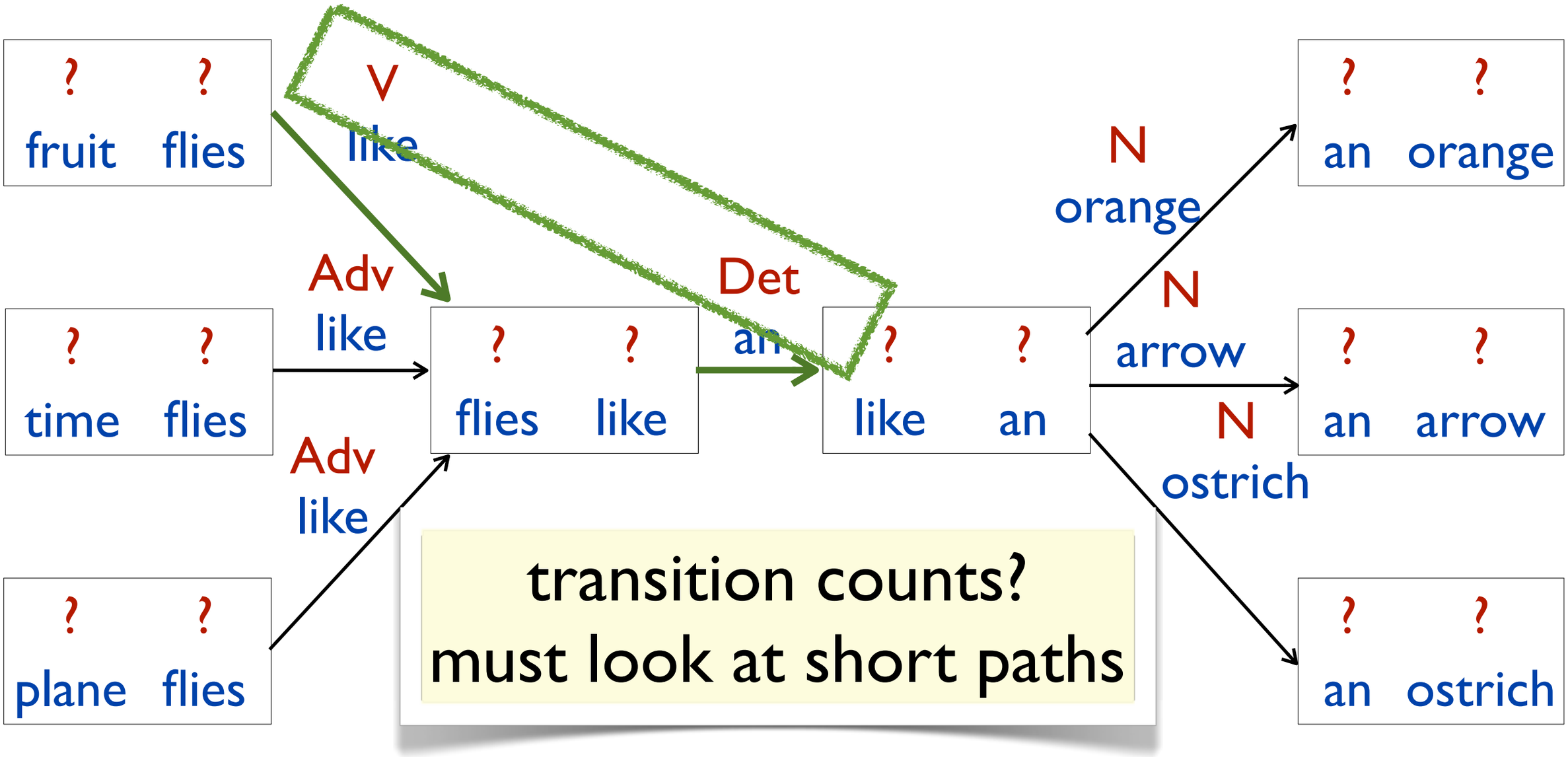
~~c(w)~~ c(t,w):



What if someone tagged our n-grams?

$n_q = 1$

c(~~w~~) c(t,w):



transition counts?
must look at short paths

Unsupervised learning

Sentence: time flies like an arrow

p(Tag|Last Tag)

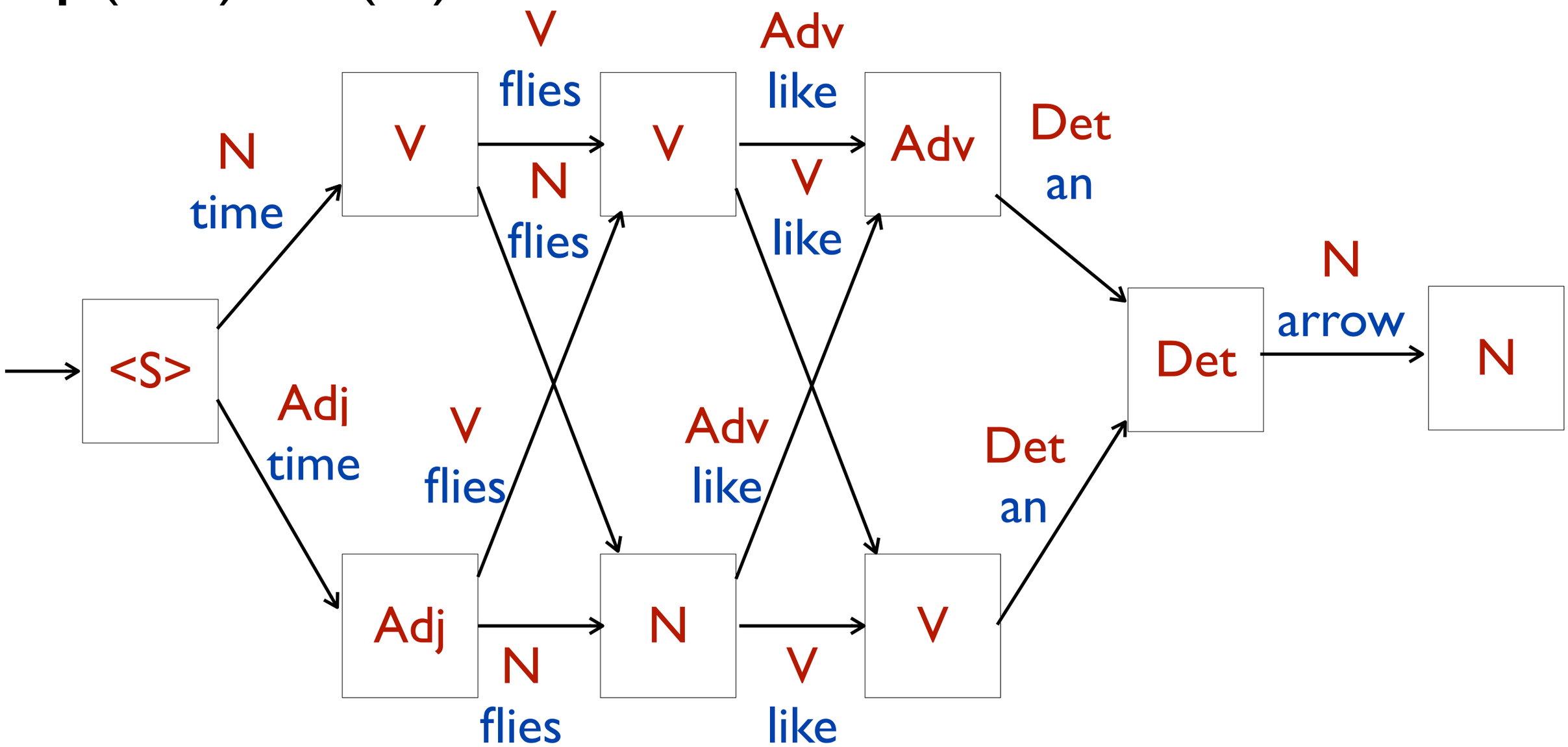
	Det	N	V	...
Det	0.1	0.1	0.5	
N	0.8	0.3	0.4	...
V	0.1	0.6	0.1	
...	...			

p(Word|Tag)

	Det	N	V	...
time	0.01	0.3	0.1	
flies	0.01	0.2	0.2	...
an	0.33	0.01	0.01	
...	...			

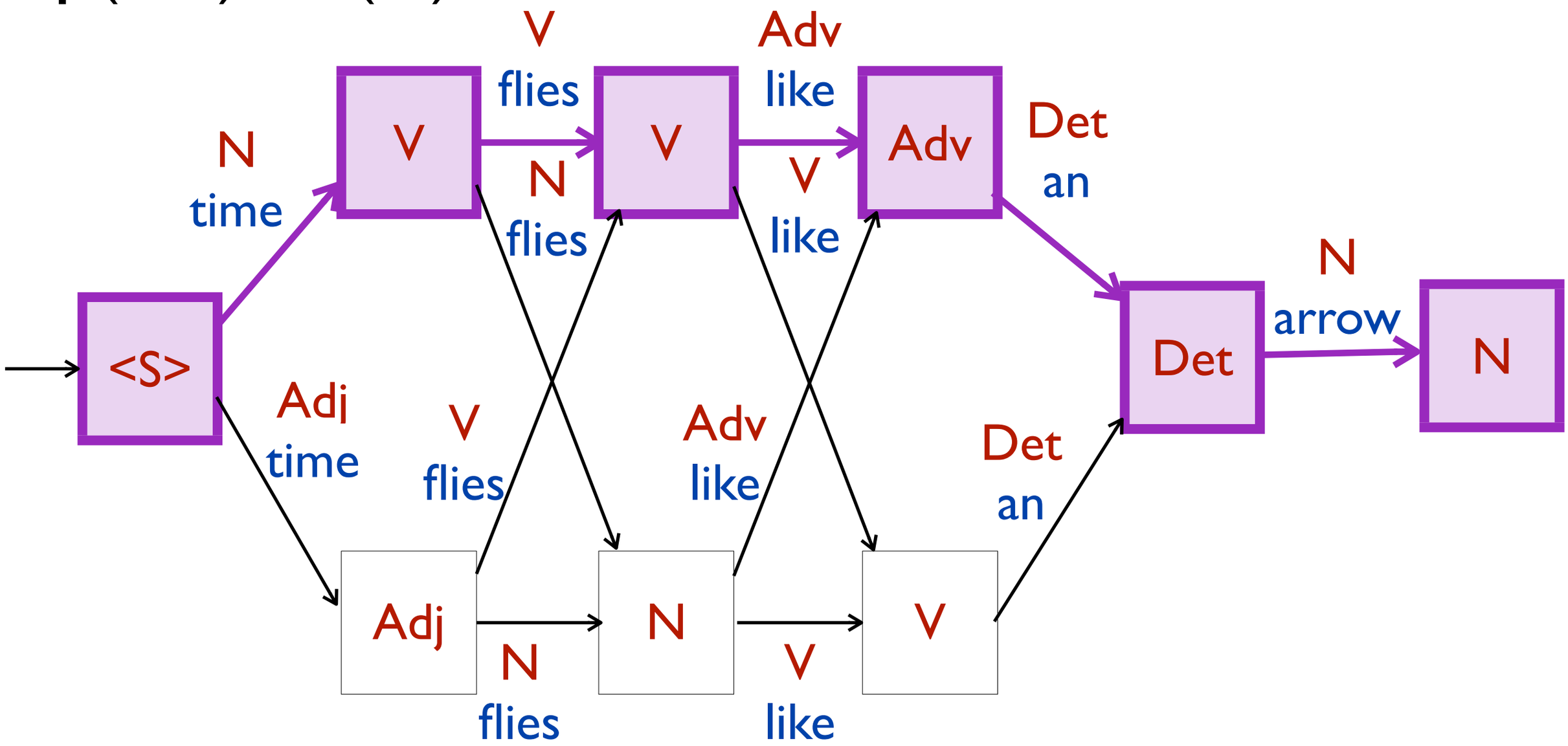
HMM Tagging Trellis

p(t,w) ∘ c(w):



HMM Tagging Trellis

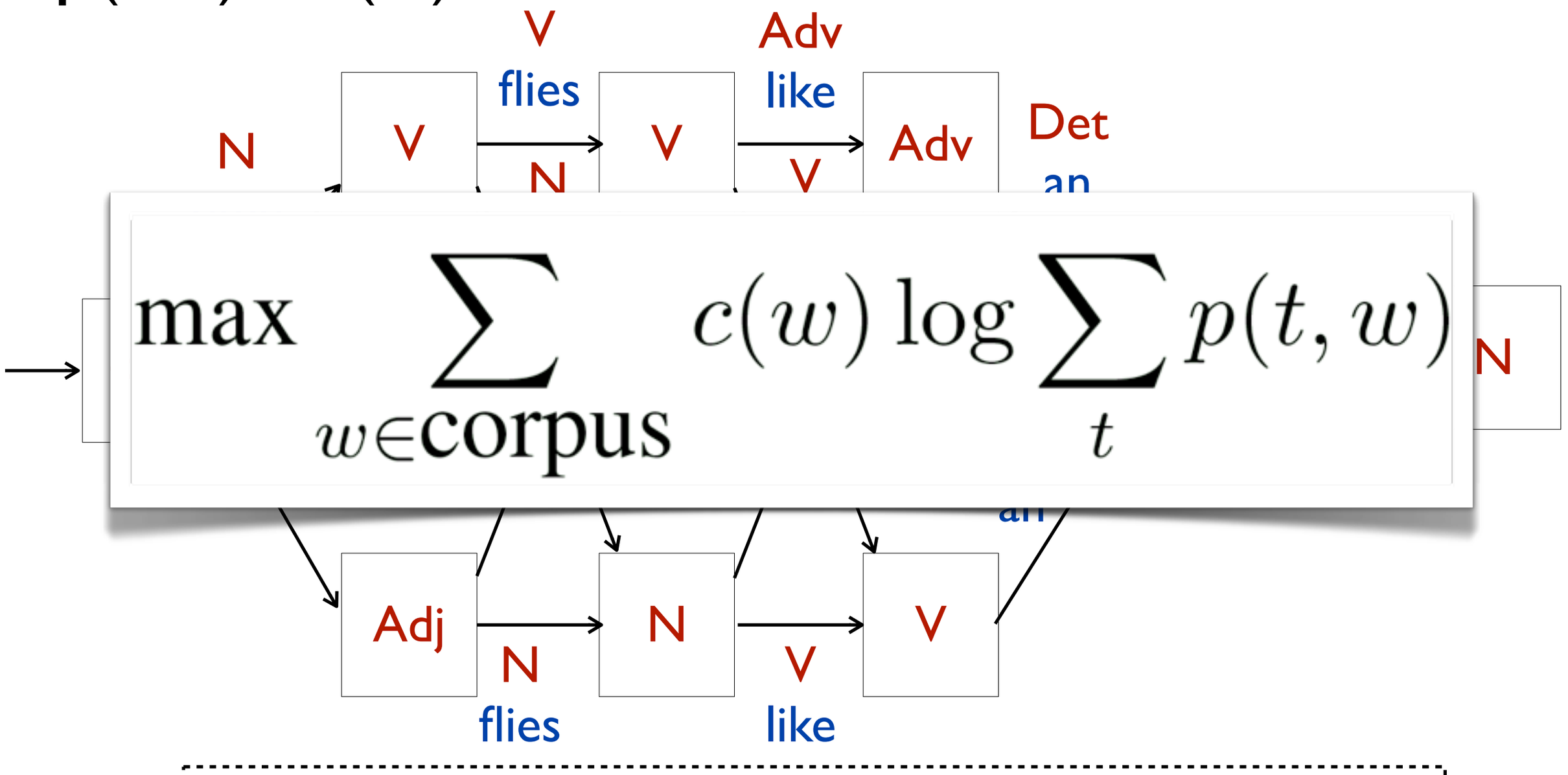
p(t,w) ∘ c(w):



time flies like an arrow

HMM Tagging Trellis

p(t,w) ∘ c(w):

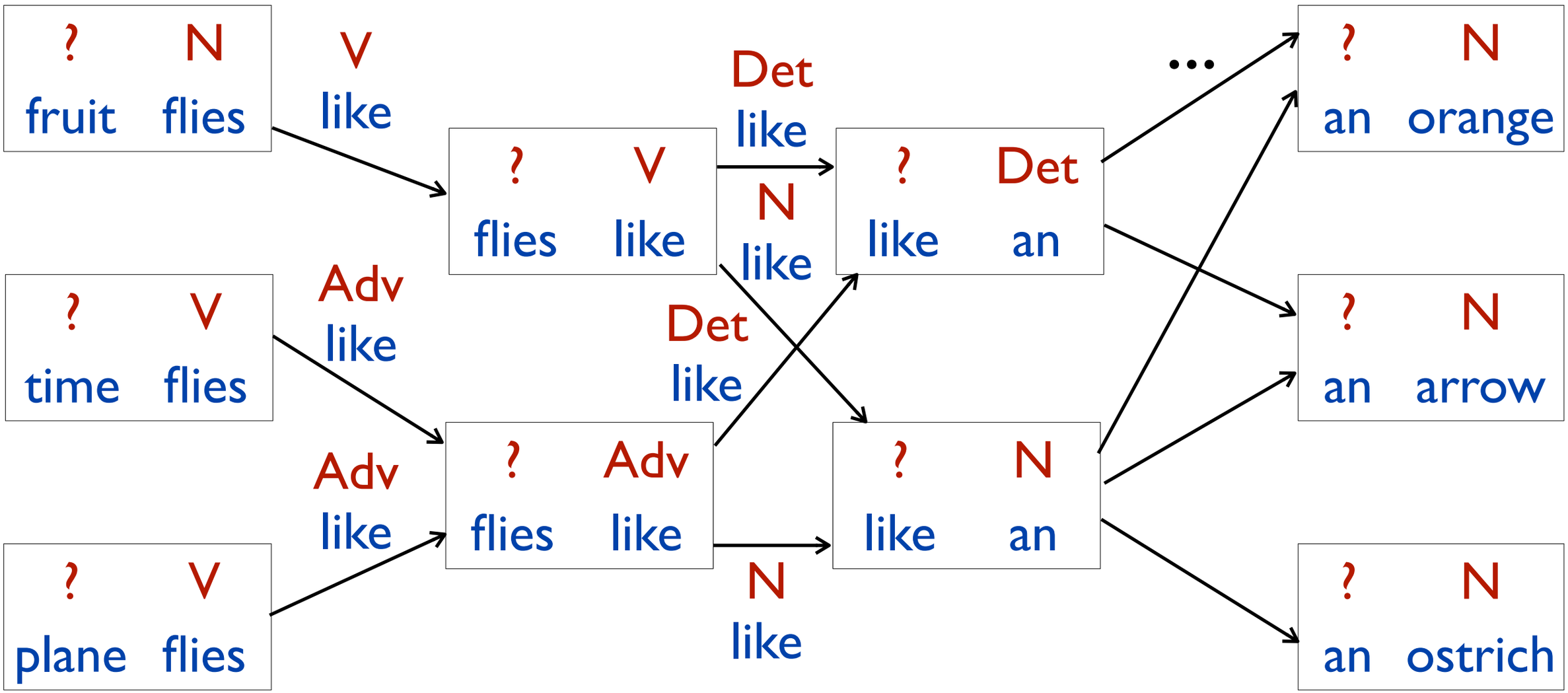


time flies like an arrow

Let's tag our own n-grams (EM)

$n_q = 2$

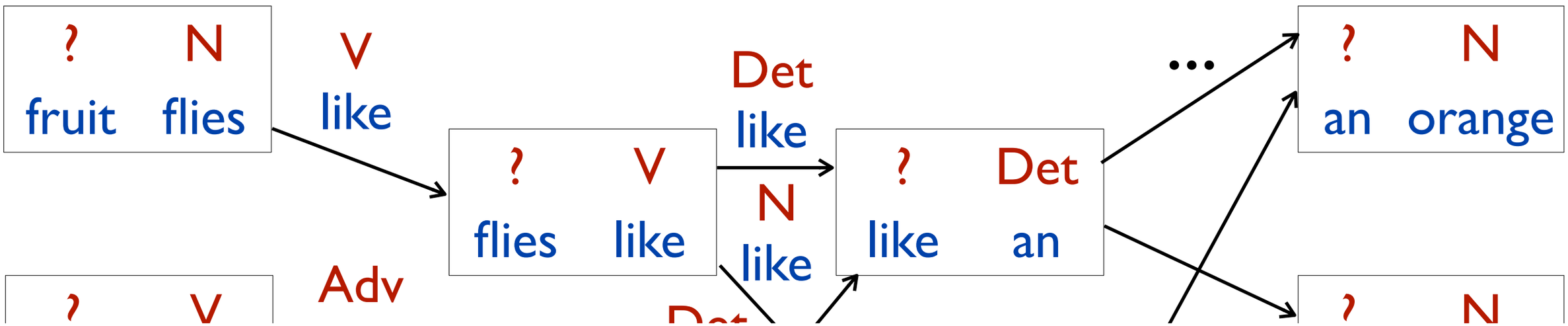
~~c(w) c(t,w)~~ c(w)q(t|w):



Let's tag our own n-grams (EM)

$n_q = 2$

~~c(w) c(t,w)~~ c(w)q(t|w):



n_c : c(w)'s word context window

n_p : p(t,w)'s tag context window

n_q : q(t|w)'s tag context window

Variational EM

$$\log \sum_t p_\theta(t, w) = \log \sum_t q(t|w)(\frac{p_\theta(t, w)}{q(t|w)})$$

$$\geq \sum_t q(t|w) \log(\frac{p_\theta(t, w)}{q(t|w)})$$

$$= \mathbb{E}_{q(t|w)}[\log p_\theta(t, w) - \log q(t|w)]$$

Variational EM

$$\log \sum_t p_\theta(t, w) = \log \sum_t q(t|w)(\frac{p_\theta(t, w)}{q(t|w)})$$

Jensen's inequality

$$\geq \sum_t q(t|w) \log(\frac{p_\theta(t, w)}{q(t|w)})$$

$$= \mathbb{E}_{q(t|w)}[\log p_\theta(t, w) - \log q(t|w)]$$

Variational EM

$$\mathbb{E}_c(w) \log \sum_t p_\theta(t, w) = \mathbb{E}_c(w) \log \sum_t q(t|w) \left(\frac{p_\theta(t, w)}{q(t|w)}\right)$$

$$\geq E_c(w) \sum_t q(t|w) \log \left(\frac{p_\theta(t, w)}{q(t|w)}\right)$$

$$= \mathbb{E}_{c(w)q(t|w)} [\log p_\theta(t, w) - \log q(t|w)]$$

Variational EM

$n_q = n_p$: variational bound is "tight"

$n_q < n_p$: we are approximating

n_c : c(w)'s word context window

n_p : p(t,w)'s tag context window

n_q : q(t|w)'s tag context window

How to maximize this bound

$$\mathbb{E}_{c(w)q(t|w)}[\log p_\theta(t,w) - \log q(t|w)]$$

- Updating p(t,w) (M-step): shown earlier
- Updating q(t|w) (E-step): more complex, but has a dynamic programming solution which makes use of finite-state machines
- Expectation semirings (Eisner 2002), details in paper

Experiments: EM vs. n-gram EM

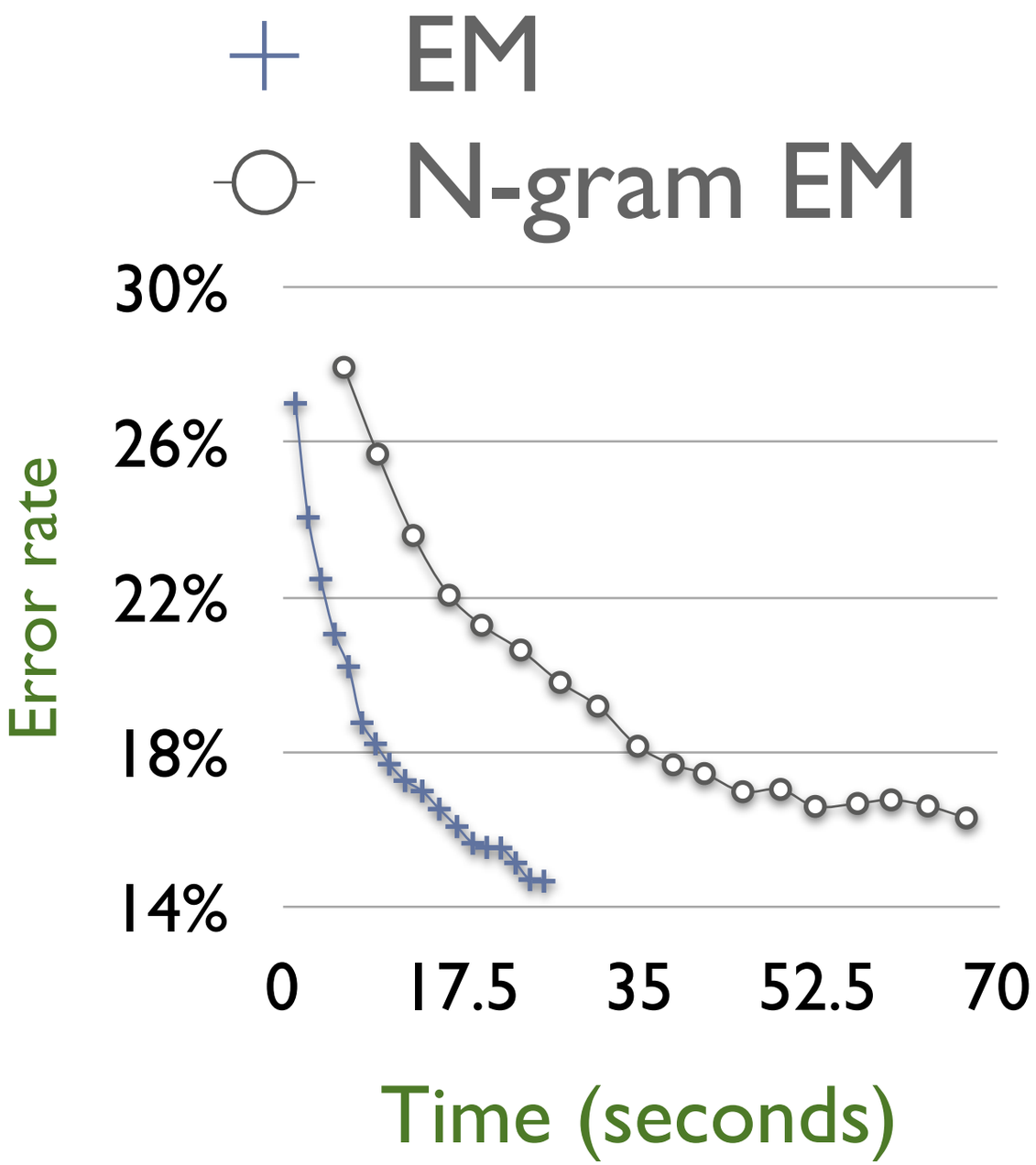
- How does EM on a full corpus compare to n-gram EM on an approximate corpus?
  - POS tagging accuracy and likelihood
- Standard setup for unsupervised POS tagging with a dictionary
- Reduced tag set (17 tags)
- Limited tag dictionary from WSJ (words must appear 5 times, otherwise all tags are possible)

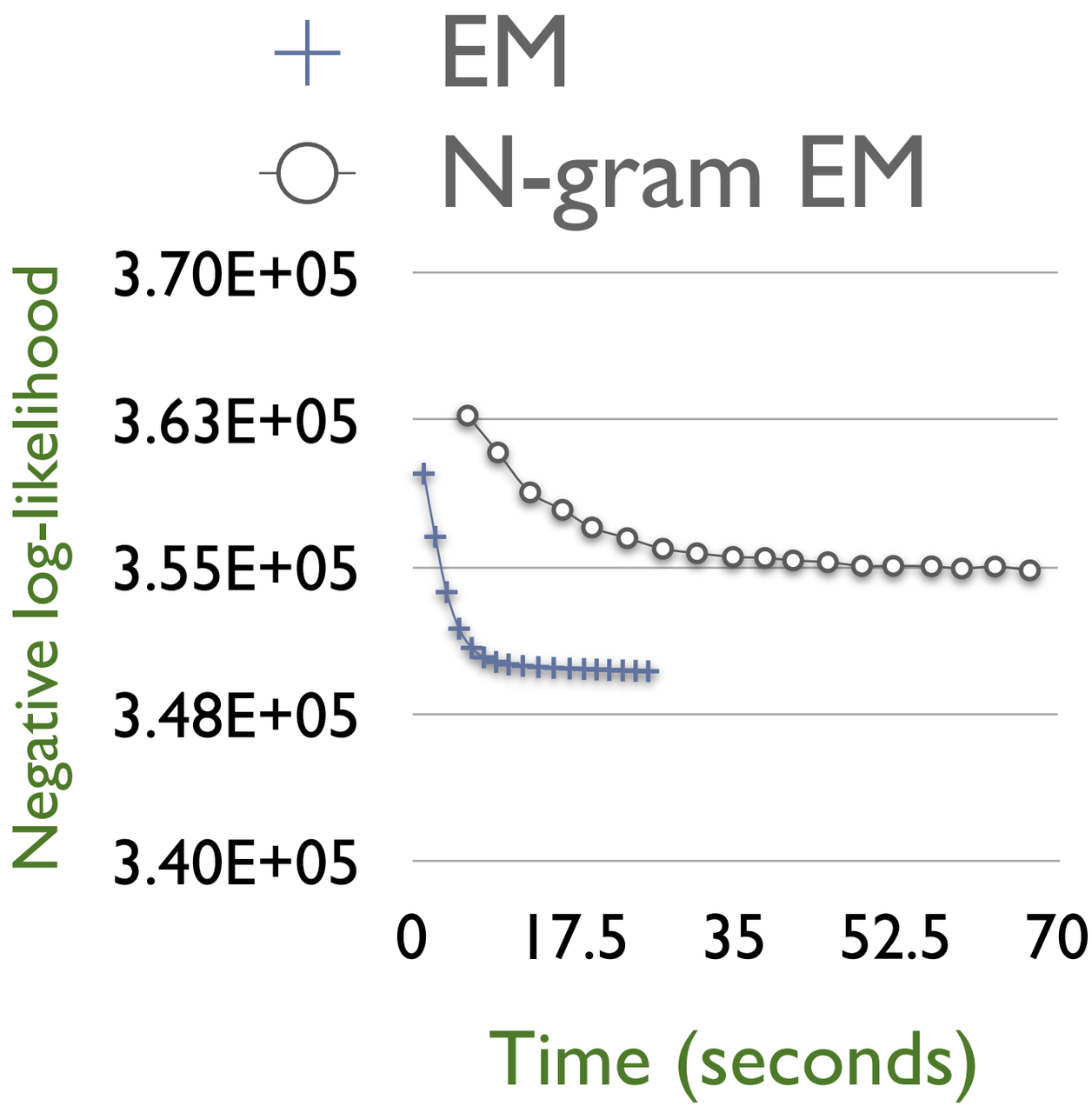
Experiments: EM vs. n-gram EM

- n-gram EM parameter choices:
  - n_c=5 - c(w) uses up to 5-grams
  - n_p=2 - p(t,w) is a bigram HMM
  - n_q=1 - q(t|w) conditions tag only on n-gram word context
    (approximate, but saves space)

Results: WSJ

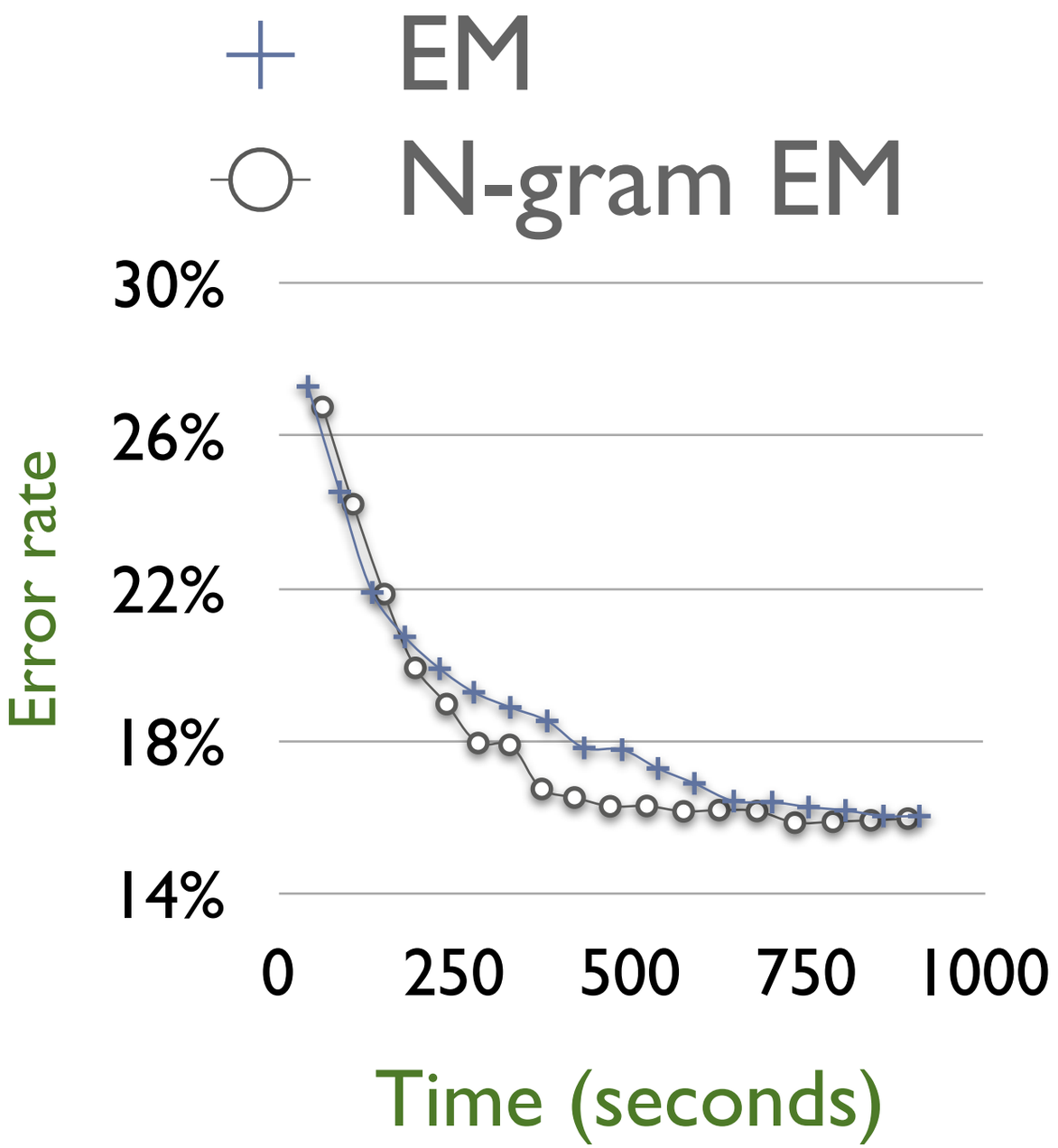
1 million words, count cutoff of 3, 430k n-grams

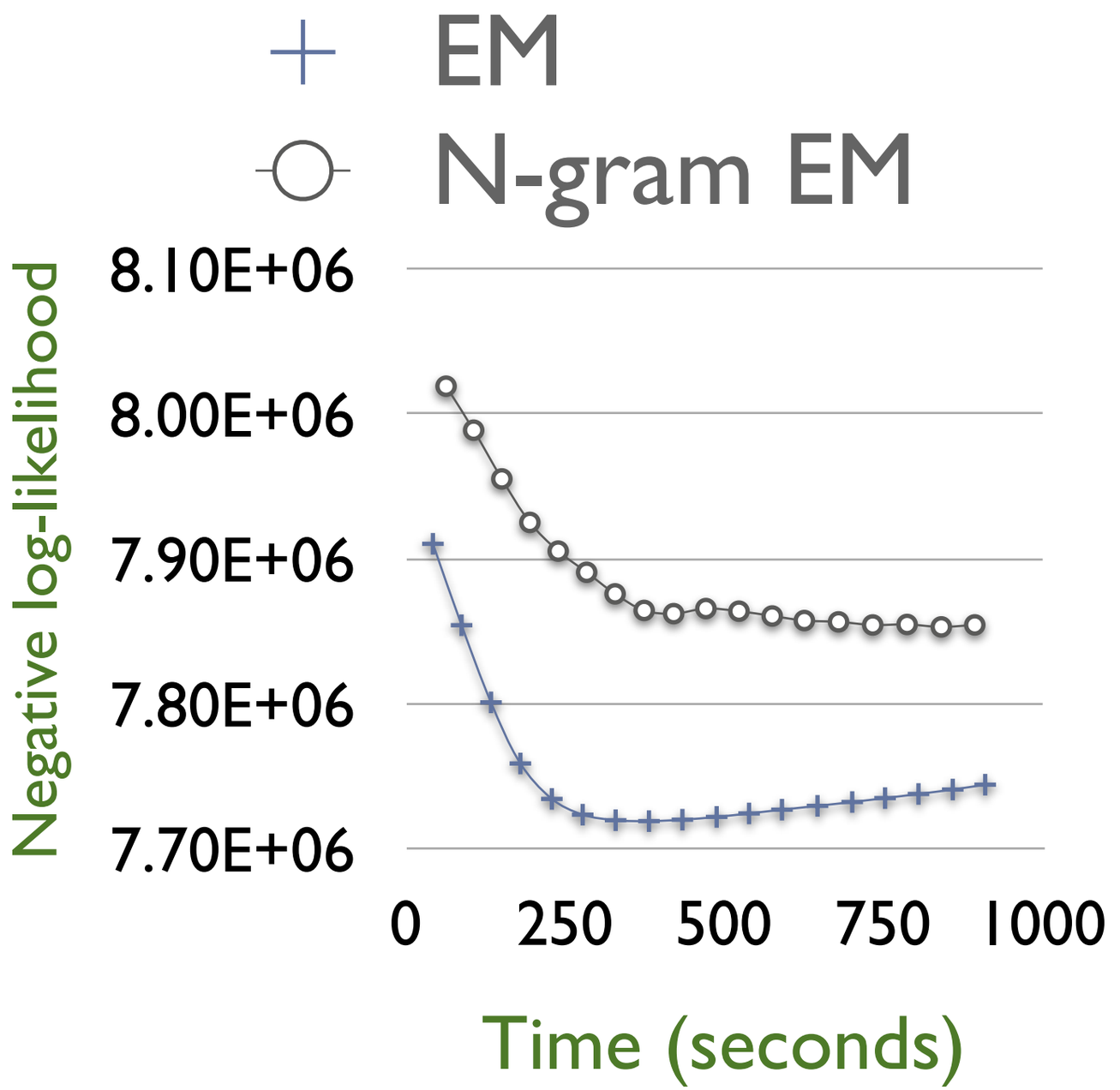




Results: 20m Gigaword

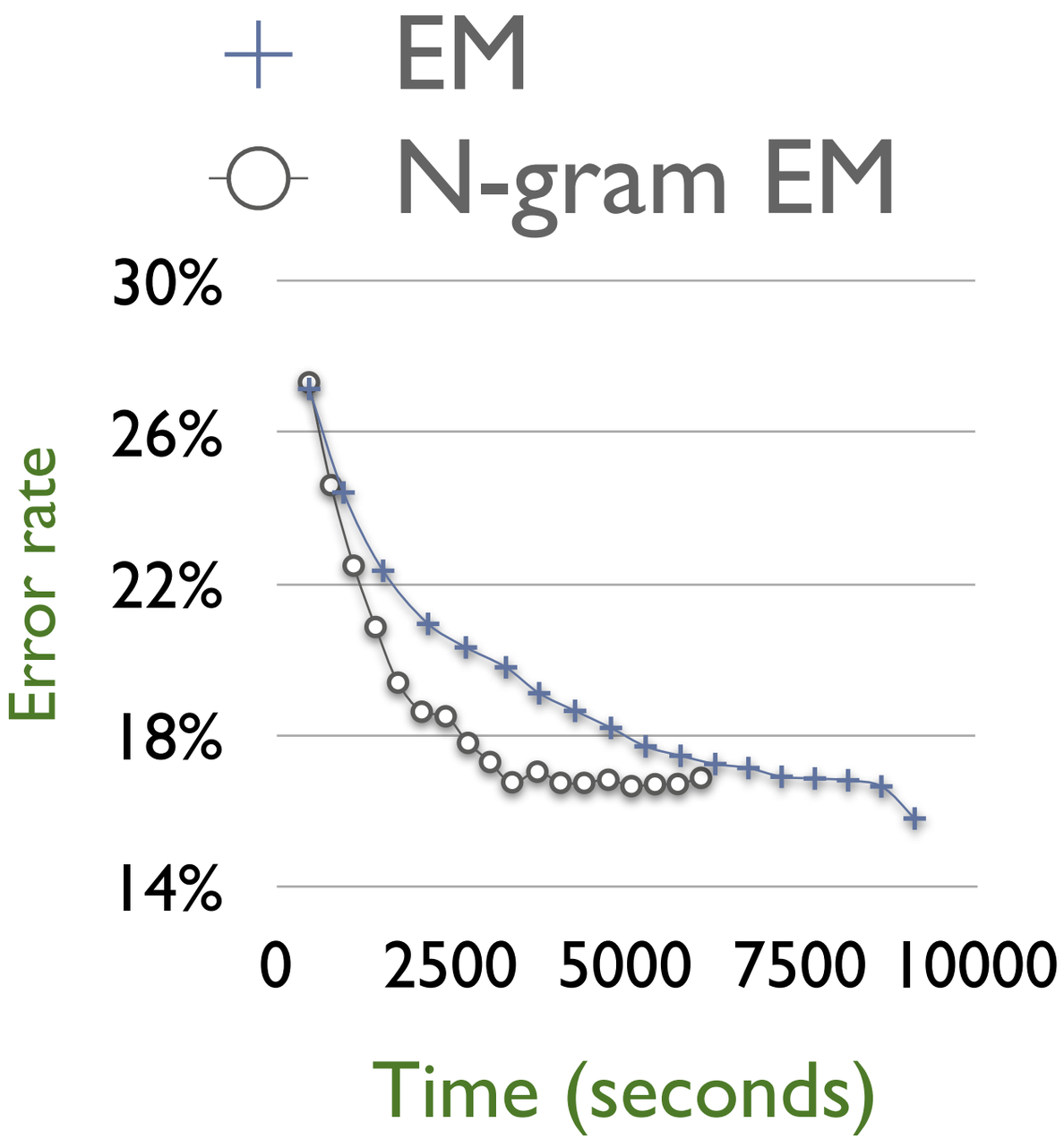
20 million words, count cutoff of 10, 2.8m n-grams

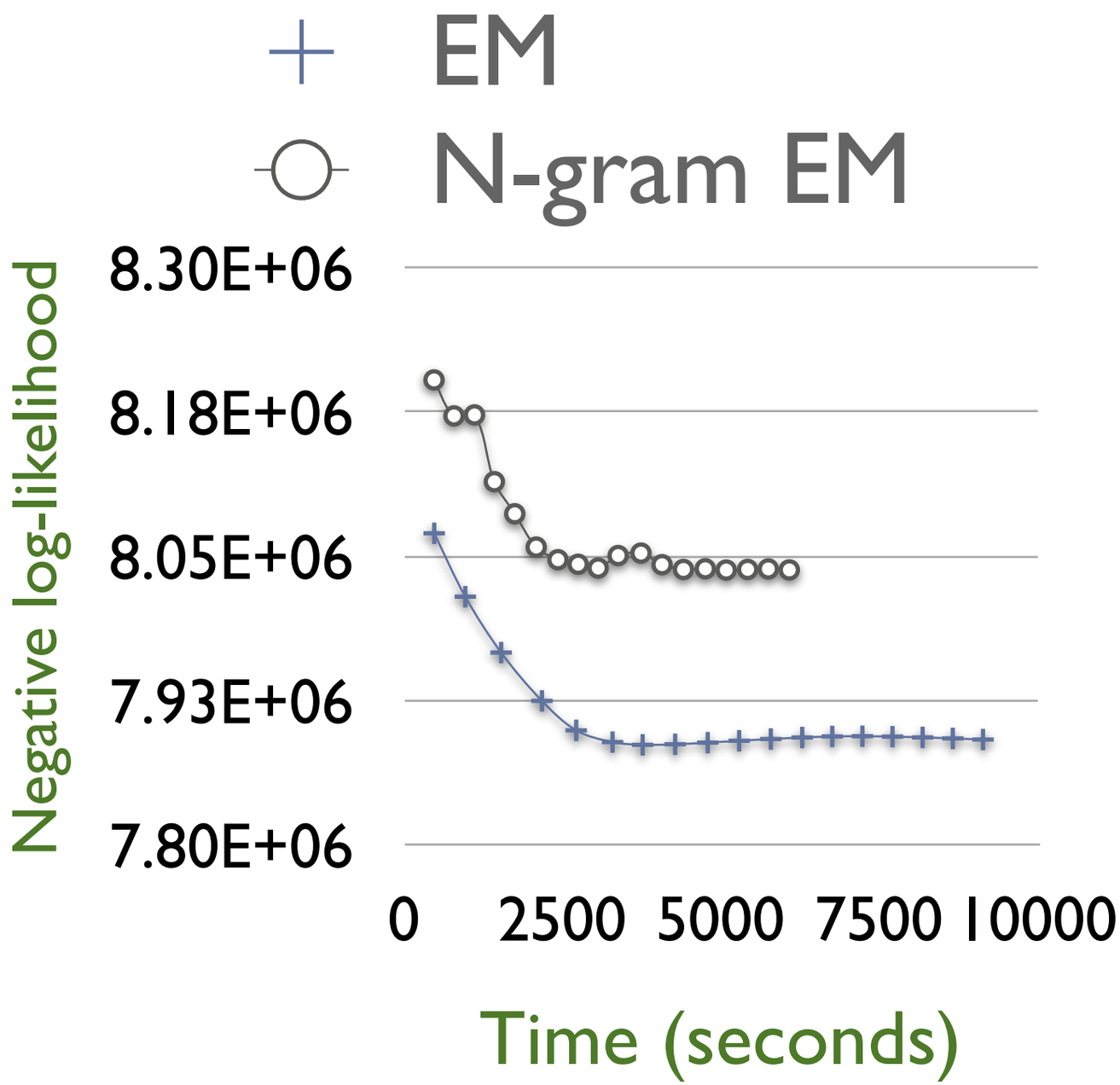




Results: 200m Gigaword

200 million words, count cutoff of 20, 14m n-grams





Conclusions

- **New problem**: train on an infinite corpus (distribution over sentences)
- **New algorithms**: exact and approximate likelihood maximization
- **New results**: faster (sublinear) training by compressing corpus into n-gram model