Noisy Channel and Hidden Markov Models

Natural Language Processing
CS 4120/6120—Spring 2017
Northeastern University

David Smith
with material from Jason Eisner & Andrew McCallum

Warren Weaver
to Norbert Wiener
4 March 1947

One thing I wanted to ask you about is this. A most serious problem, for UNESCO and for the constructive and peaceful future of the planet, is the problem of translation, as it unavoidably affects the communication between peoples. Huxley has recently told me that they are appalled by the magnitude and the importance of the translation job.

Recognizing fully, even though necessarily vaguely, the semantic difficulties because of multiple meanings, etc., I have wondered if it were unthinkable to design a computer which would translate. Even if it would translate only scientific material (where the semantic difficulties are very notably less), and even if it did produce an inelegant (but intelligible) result, it would seem to me worth while.

Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography—methods which I believe succeed even when one does not know what language has been coded—one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: "**This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.**"

Word Segmentation

theprophetsaidtothecity

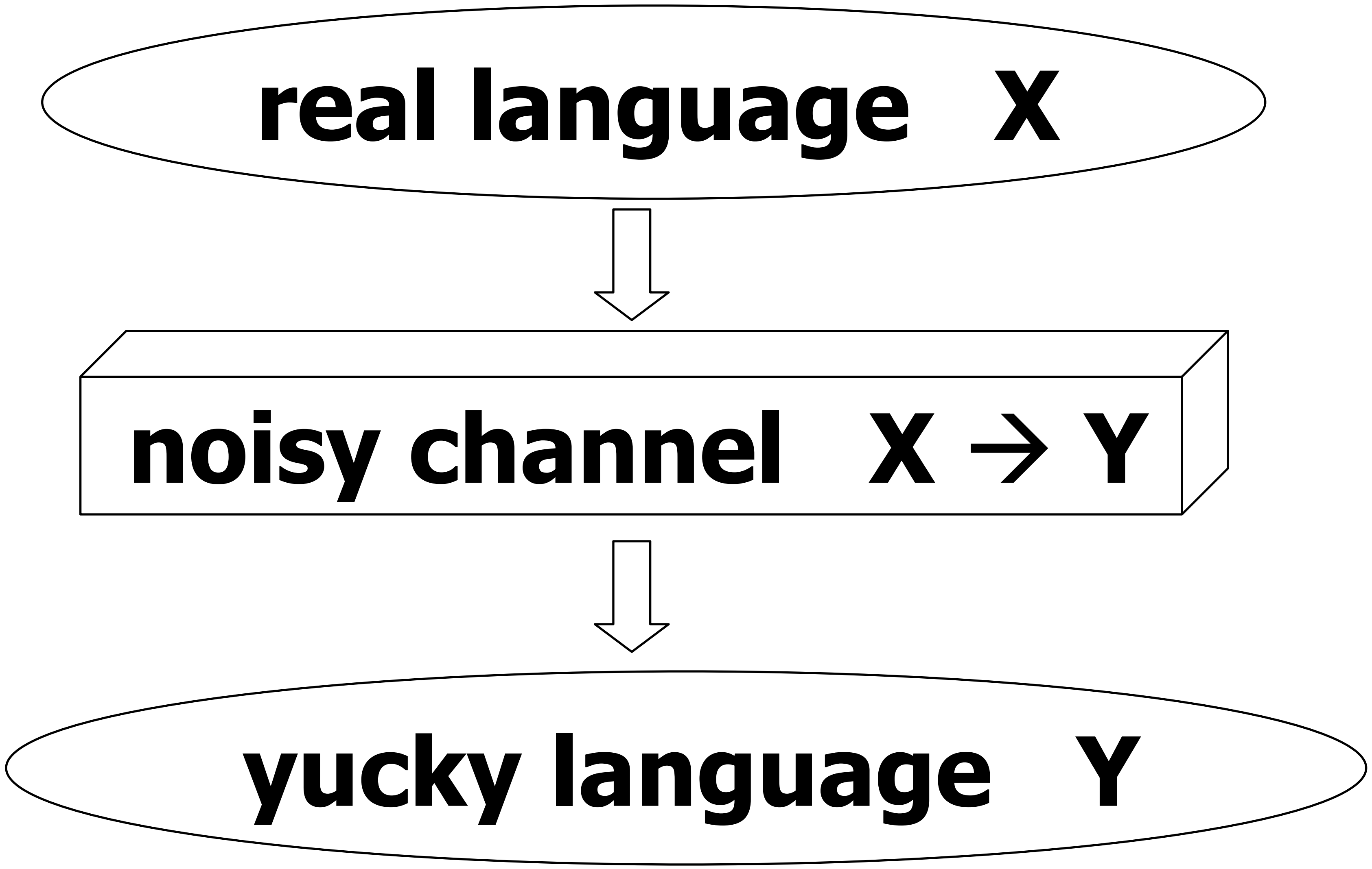
- What does this say?
  - And what other words are substrings?

- Given L = a "lexicon" FSA that matches all English words.
- How to apply to this problem?
- What if Lexicon is weighted?
- From unigrams to bigrams?
- Smooth L to include unseen words?

Spelling correction

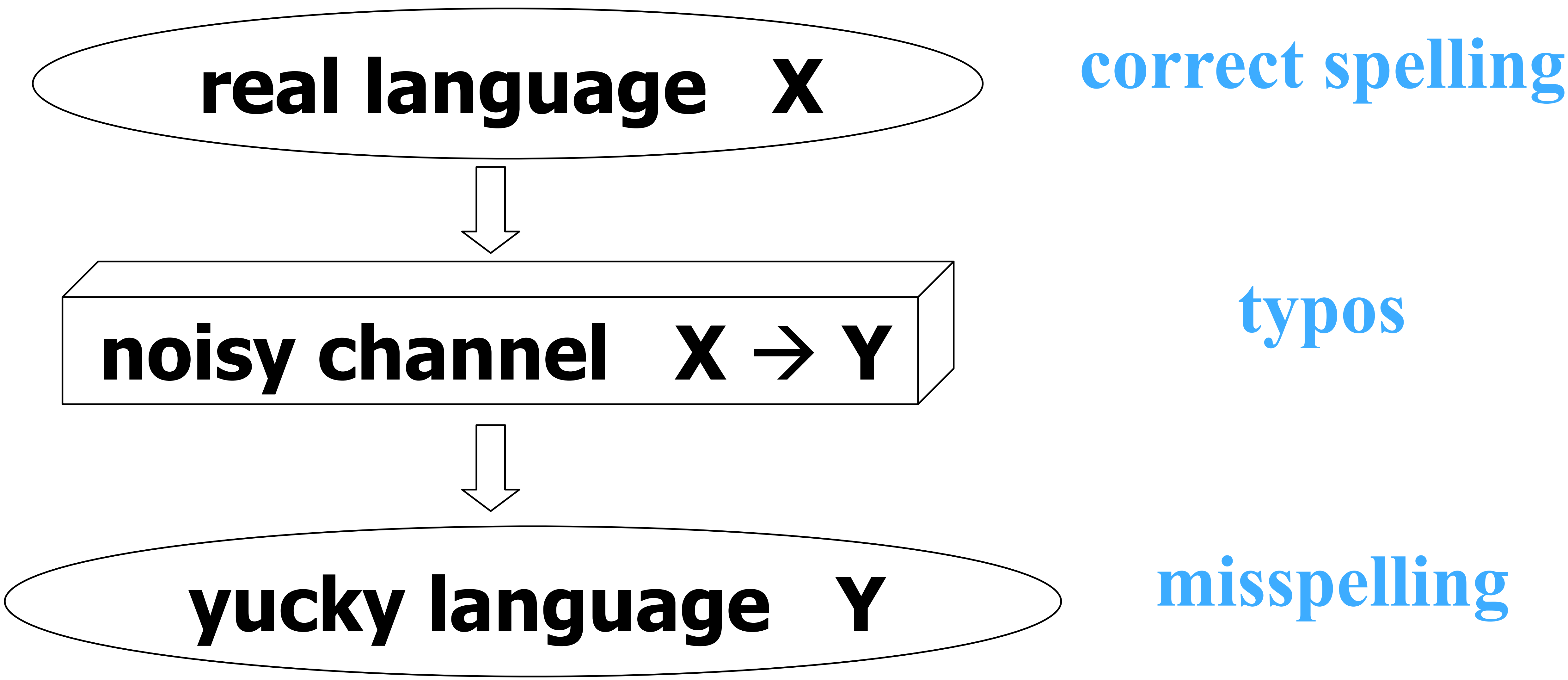
- Spelling correction also needs a lexicon L
- But there is distortion …
  - Let T be a transducer that models common typos and other spelling errors
    - ance (→) ence (deliverance, ...)
    - e → ε (deliverance, ...)
    - ε → e // Cons _ Cons (athlete, ...)
    - rr → r (embarrasş occurrence, …)
    - ge → dge (privilege, …)
    - etc.
  - Now what can you do with L .o. T ?
- Should T and L have probabilities?
- Want T to include "all possible" errors …

Noisy Channel Model



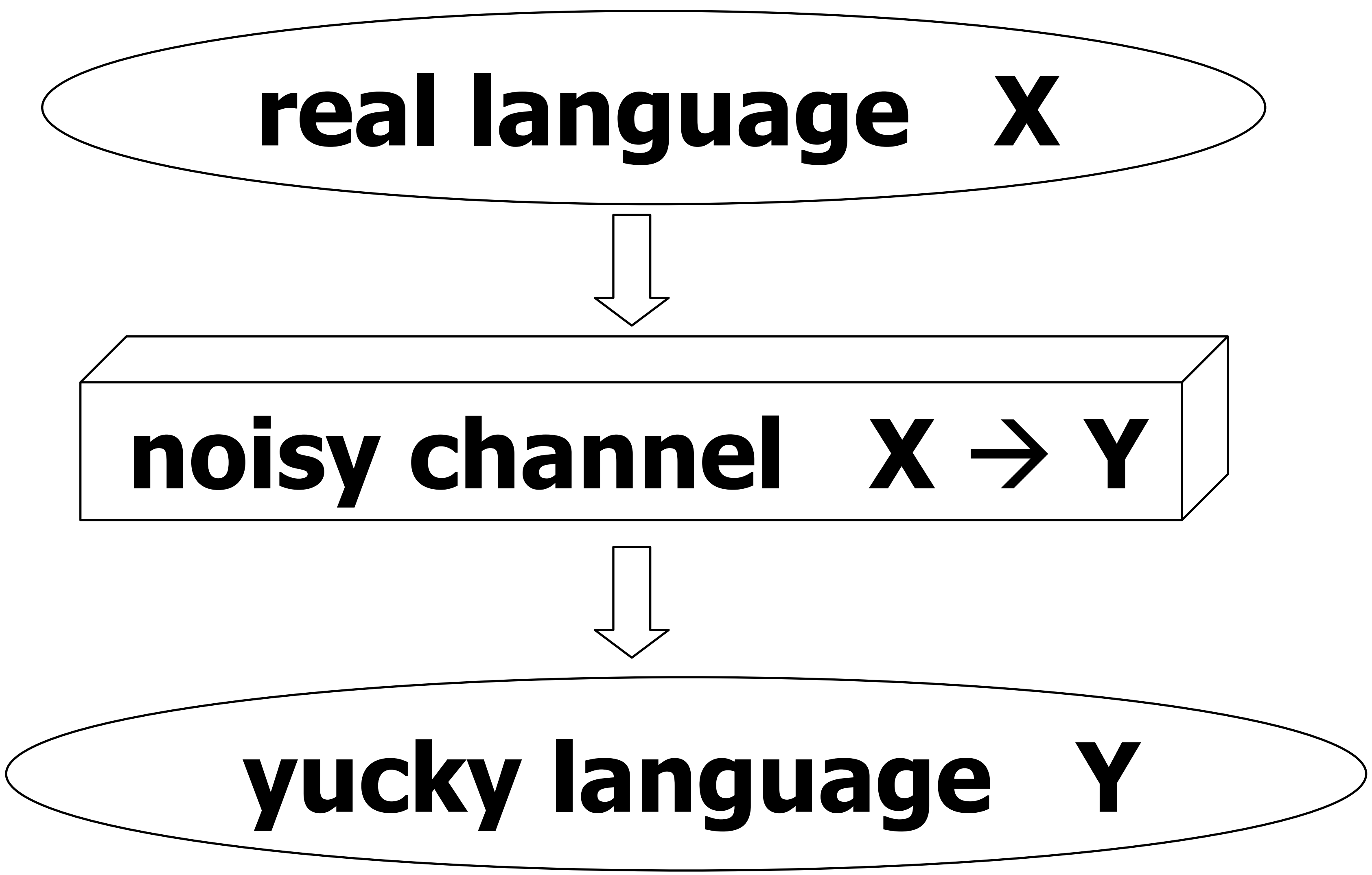
want to recover X from Y

Noisy Channel Model



want to recover X from Y

Noisy Channel Model



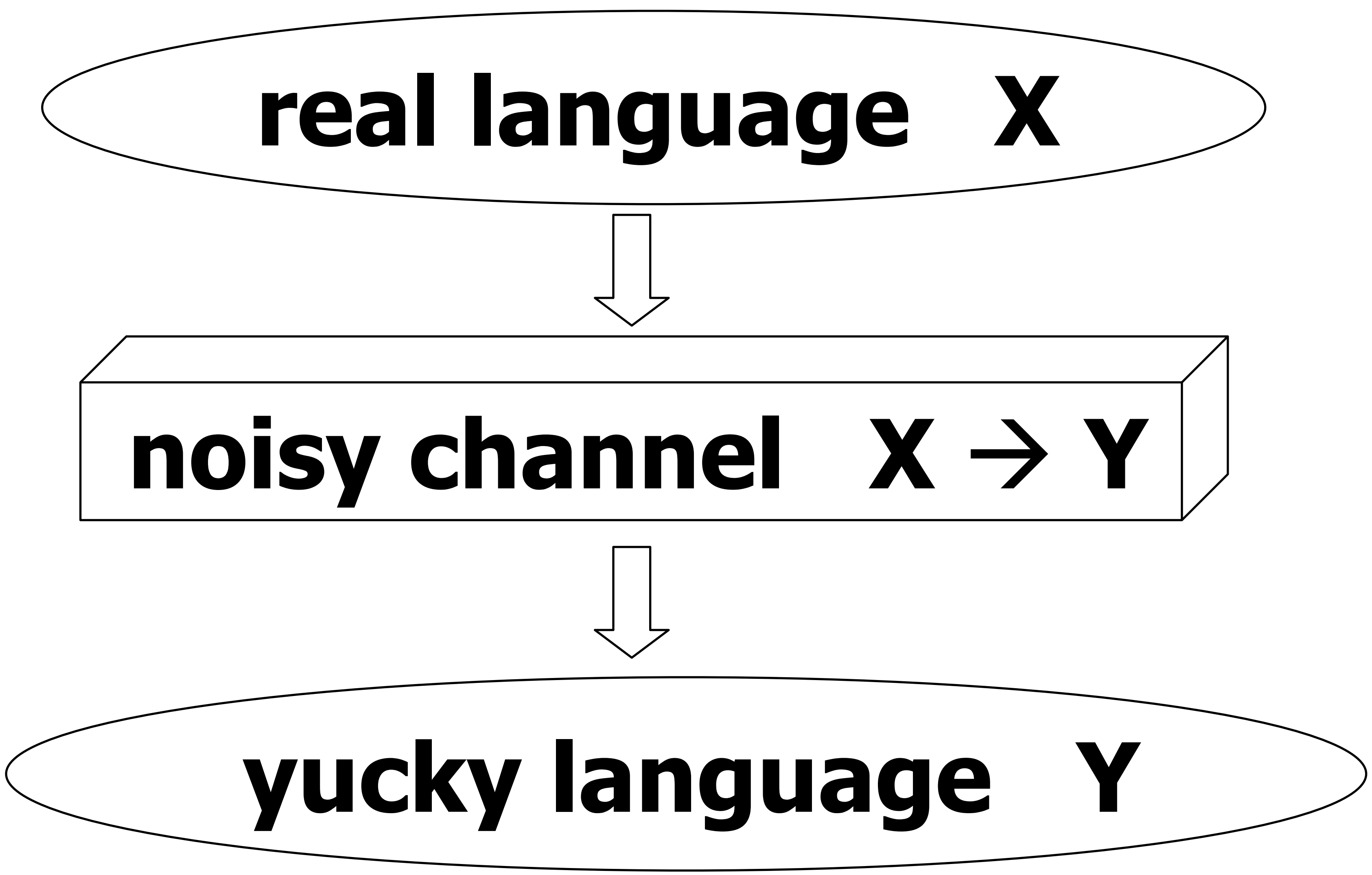
(lexicon space)*

delete spaces

text w/o spaces

want to recover X from Y

Noisy Channel Model



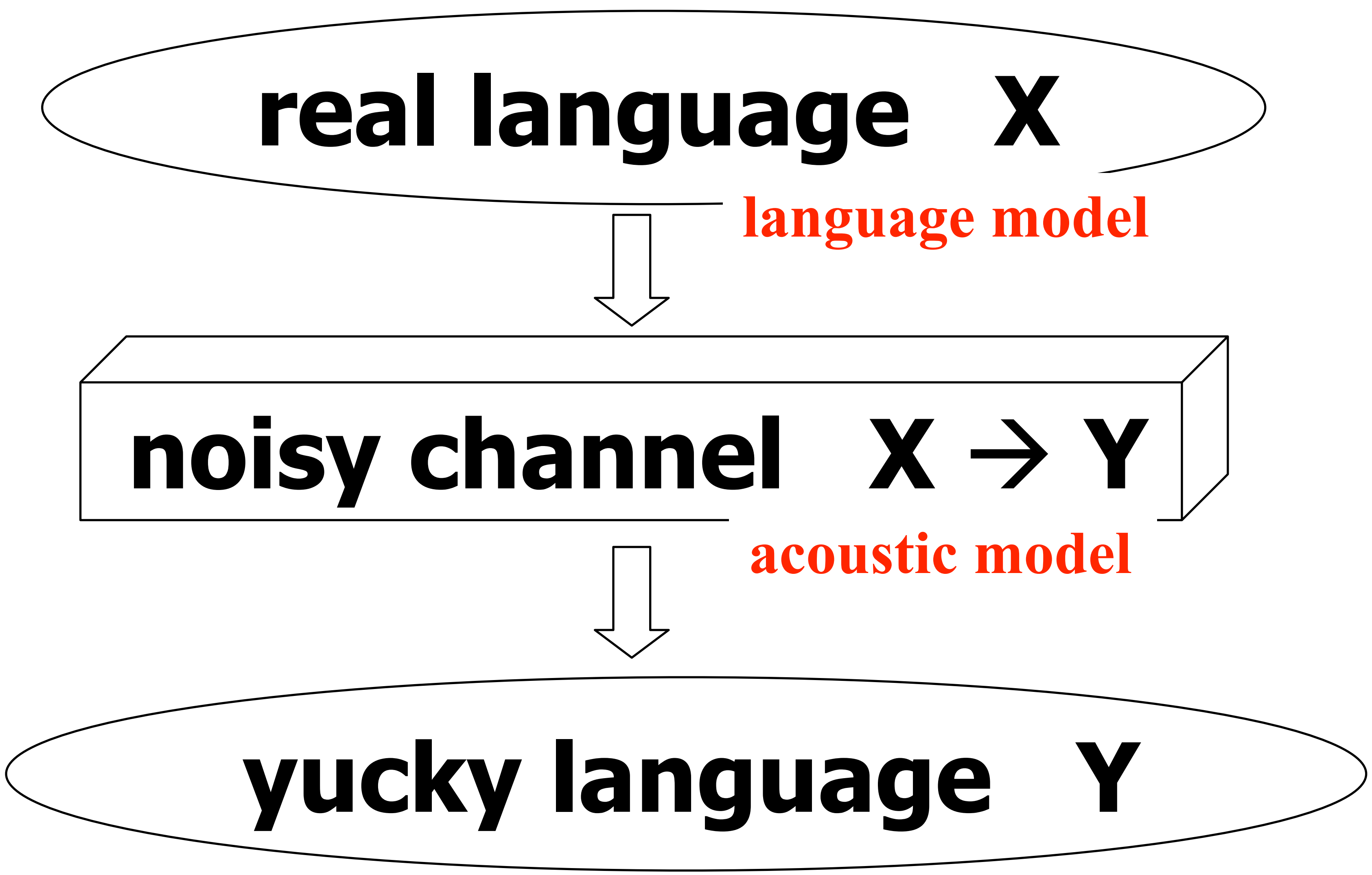
(lexicon space)*

pronunciation

speech

want to recover X from Y

Noisy Channel Model



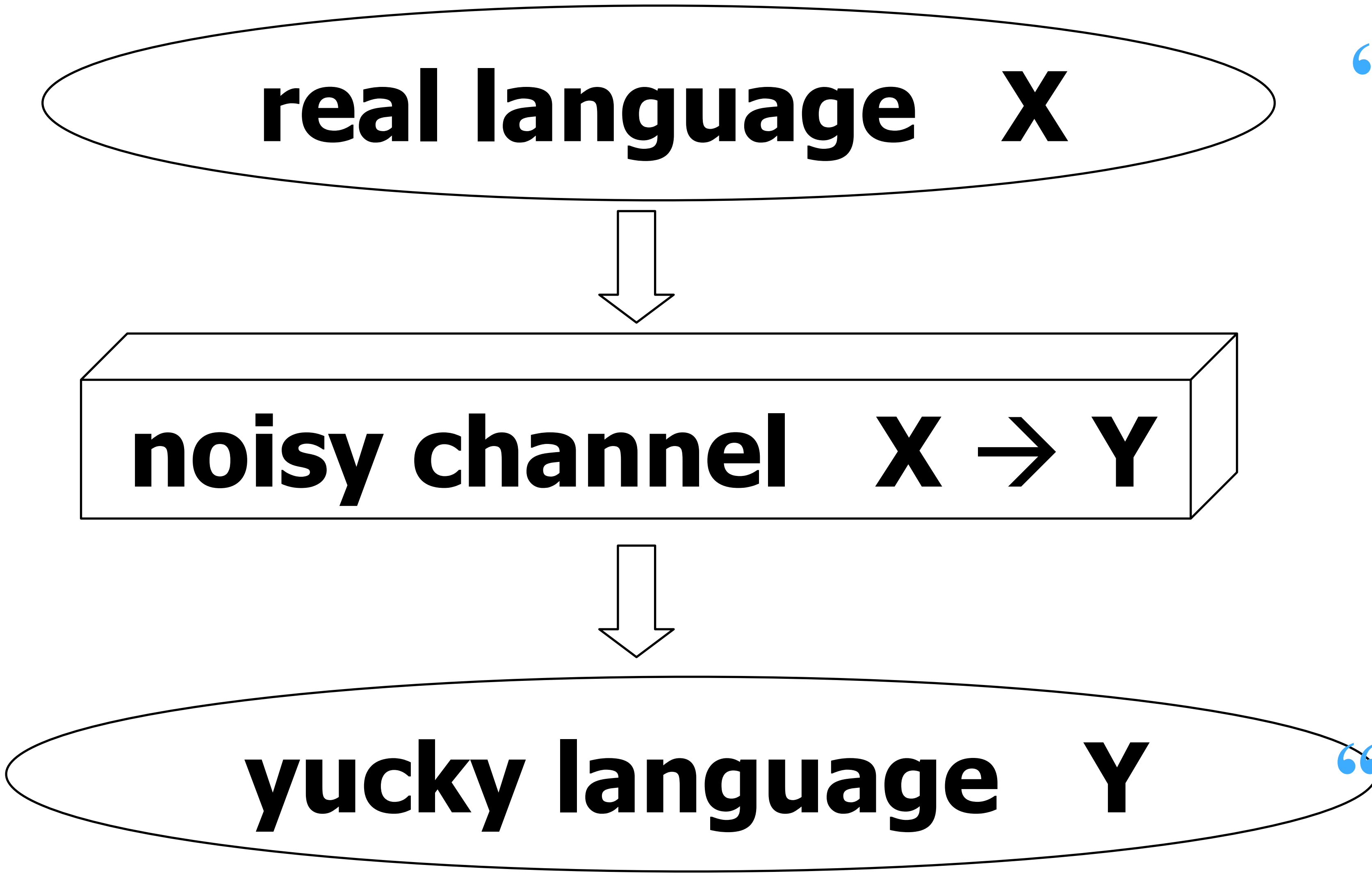
(lexicon space)*

pronunciation

speech

want to recover X from Y

Noisy Channel Model



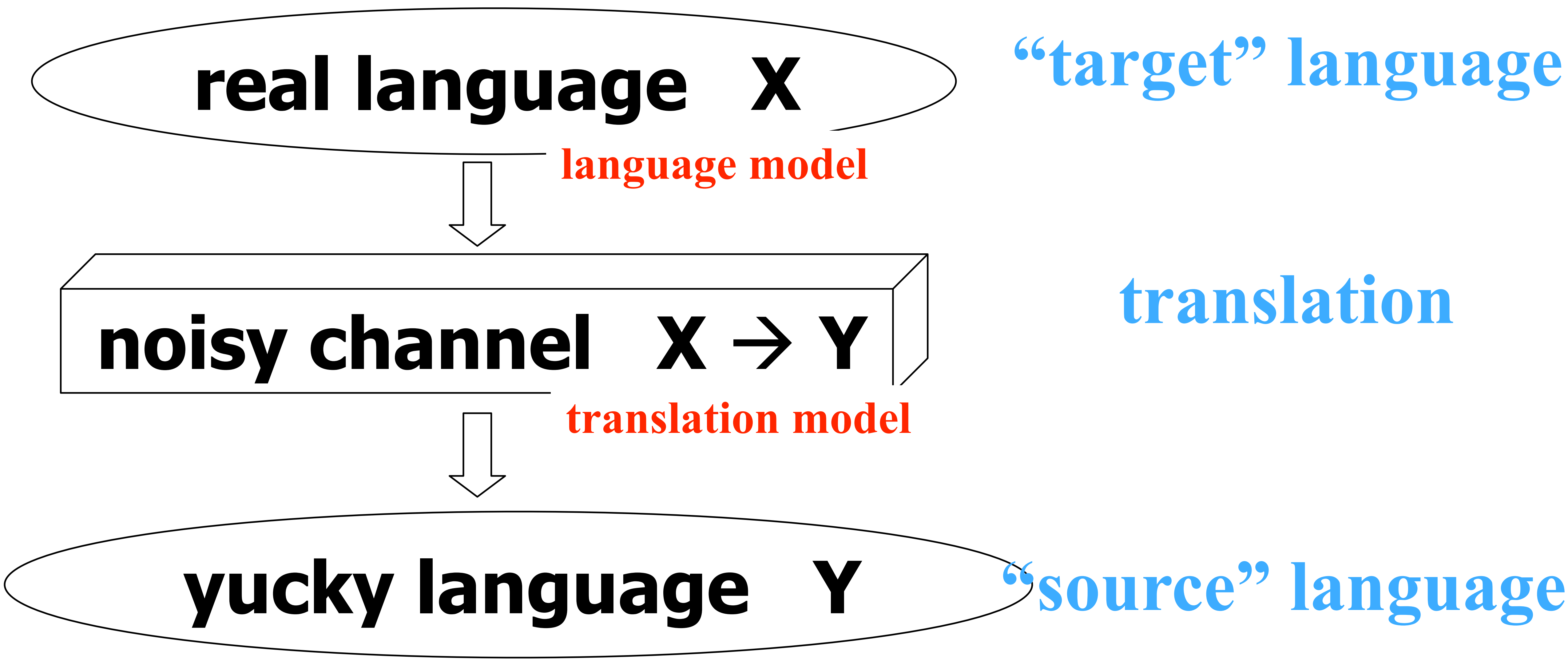
"target" language

translation

"source" language

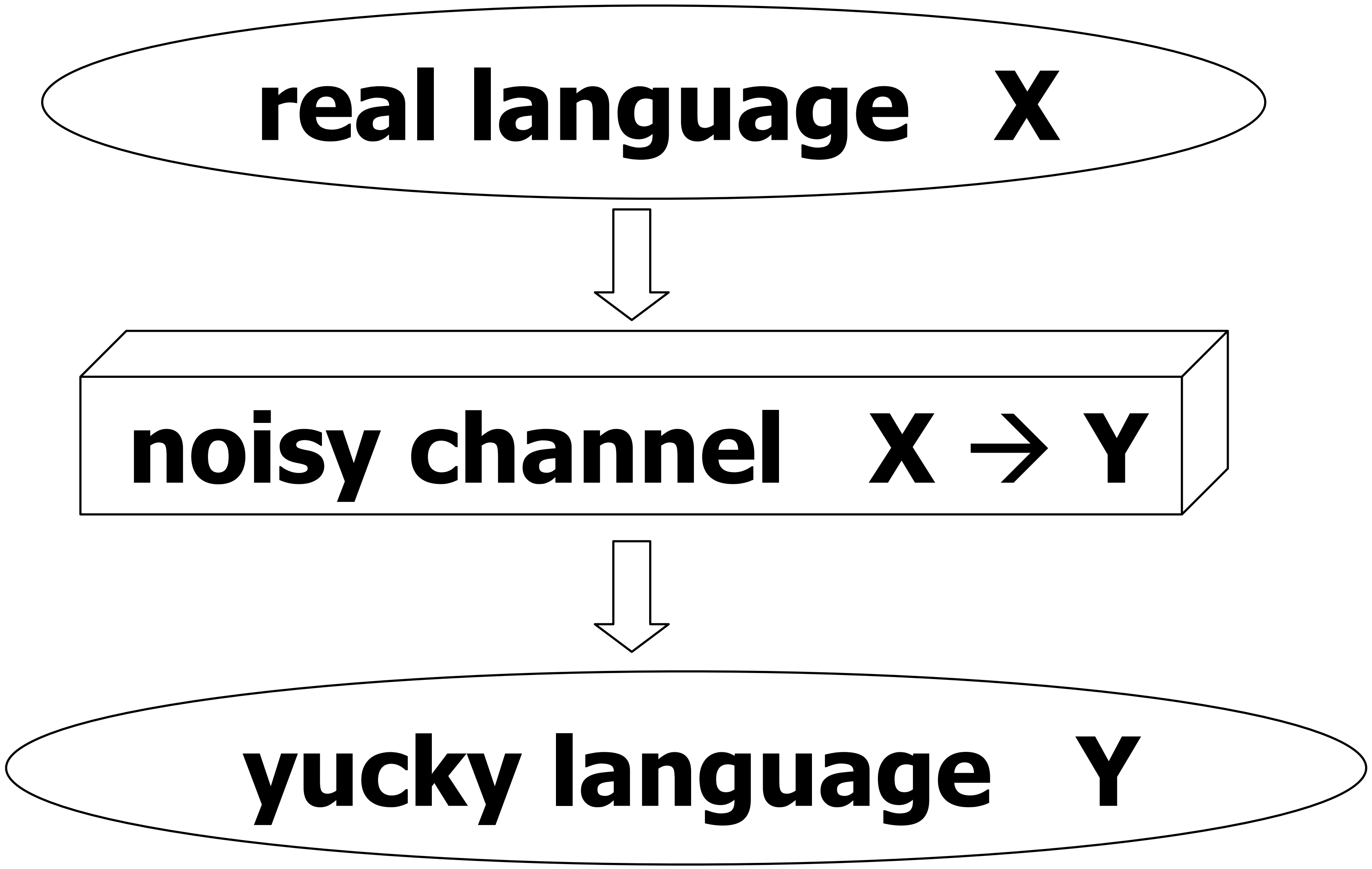
want to recover X from Y

Noisy Channel Model



want to recover X from Y

Noisy Channel Model



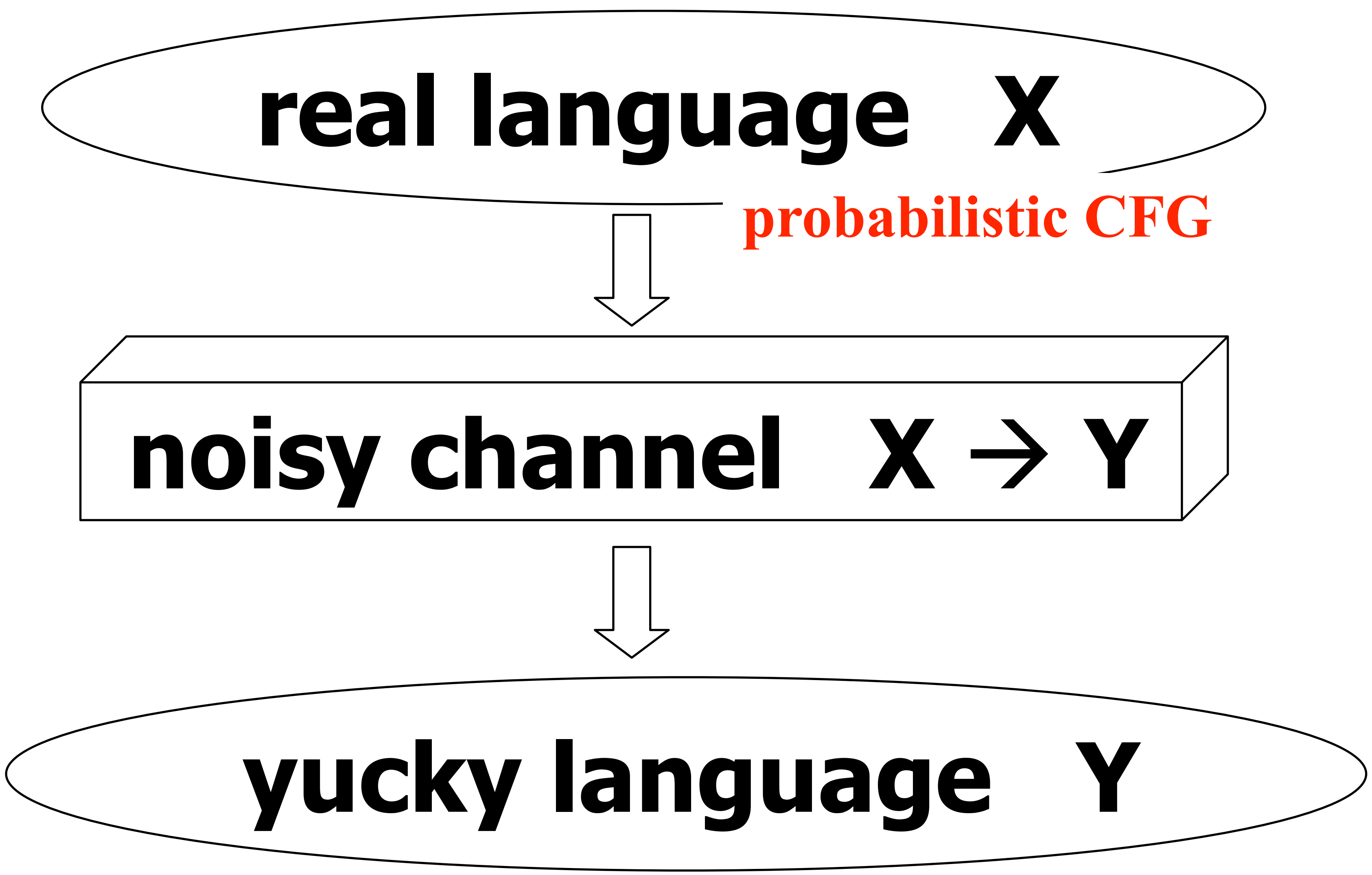
tree

delete everything but terminals

text

want to recover X from Y

Noisy Channel Model



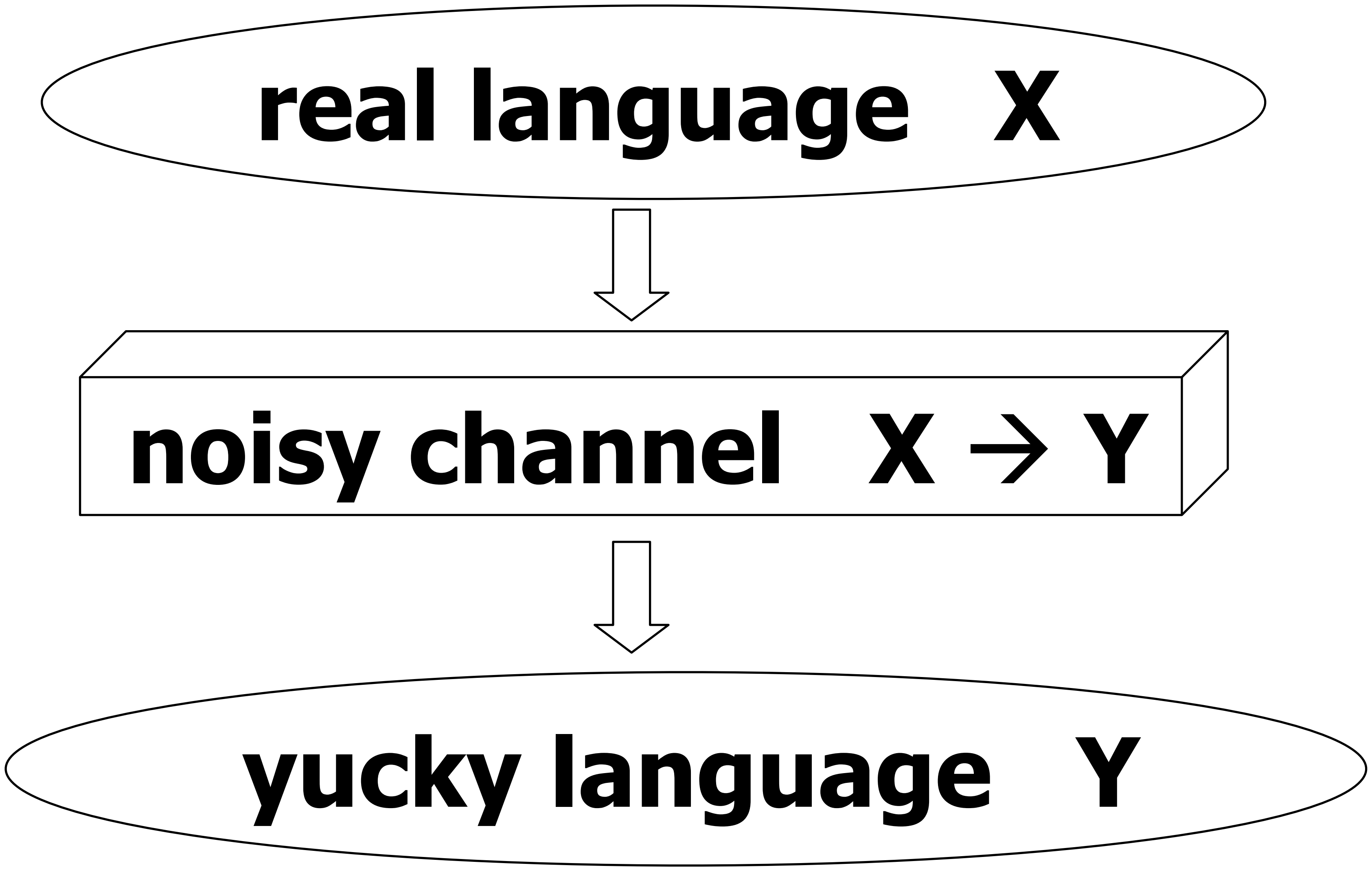
tree

delete everything
but terminals

text

want to recover X from Y

Noisy Channel Model



$$p(X)$$

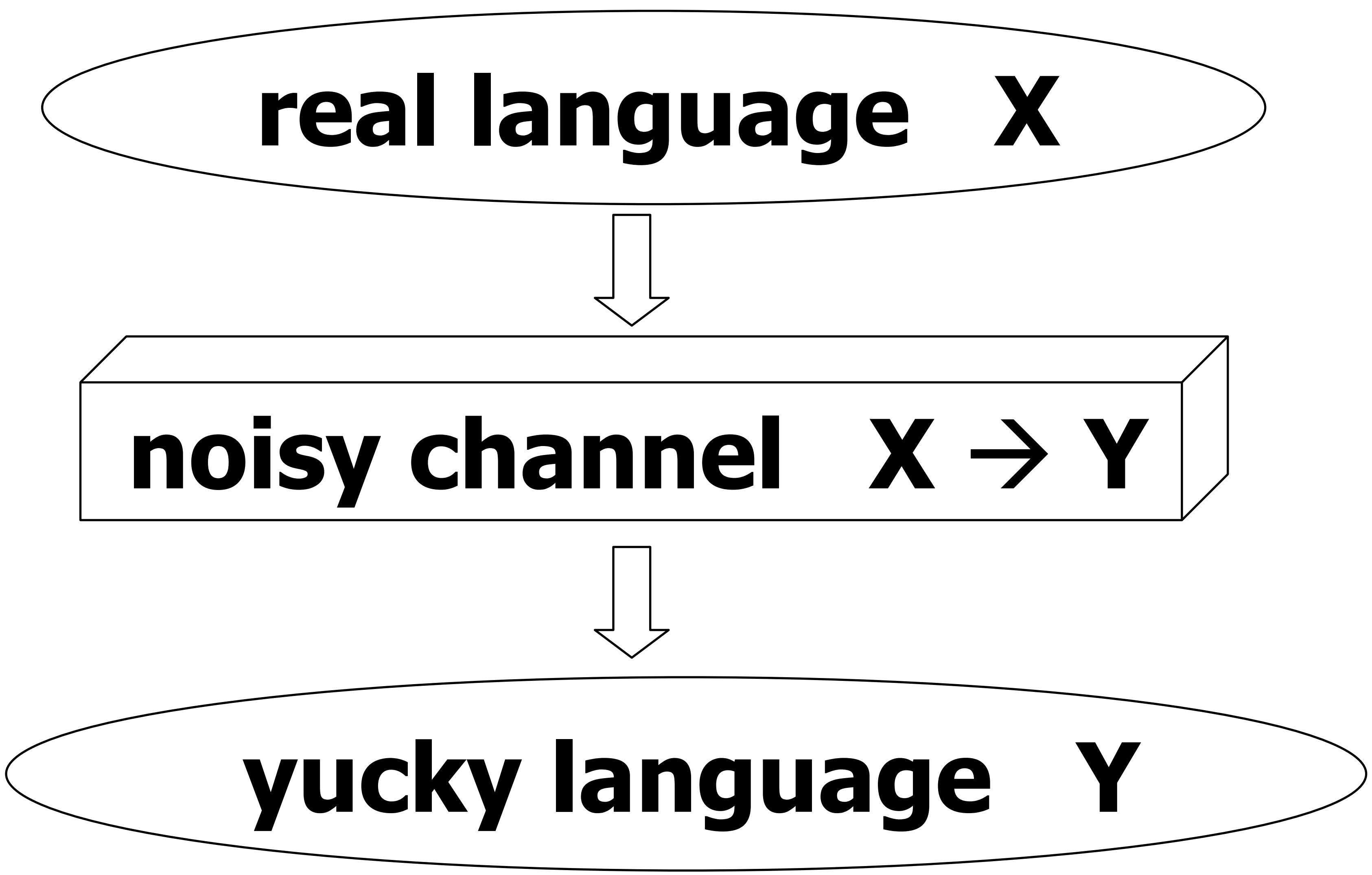
$$*$$

$$p(Y \mid X)$$

$$=$$

$$p(X,Y)$$

Noisy Channel Model



$p(X)$

$*$

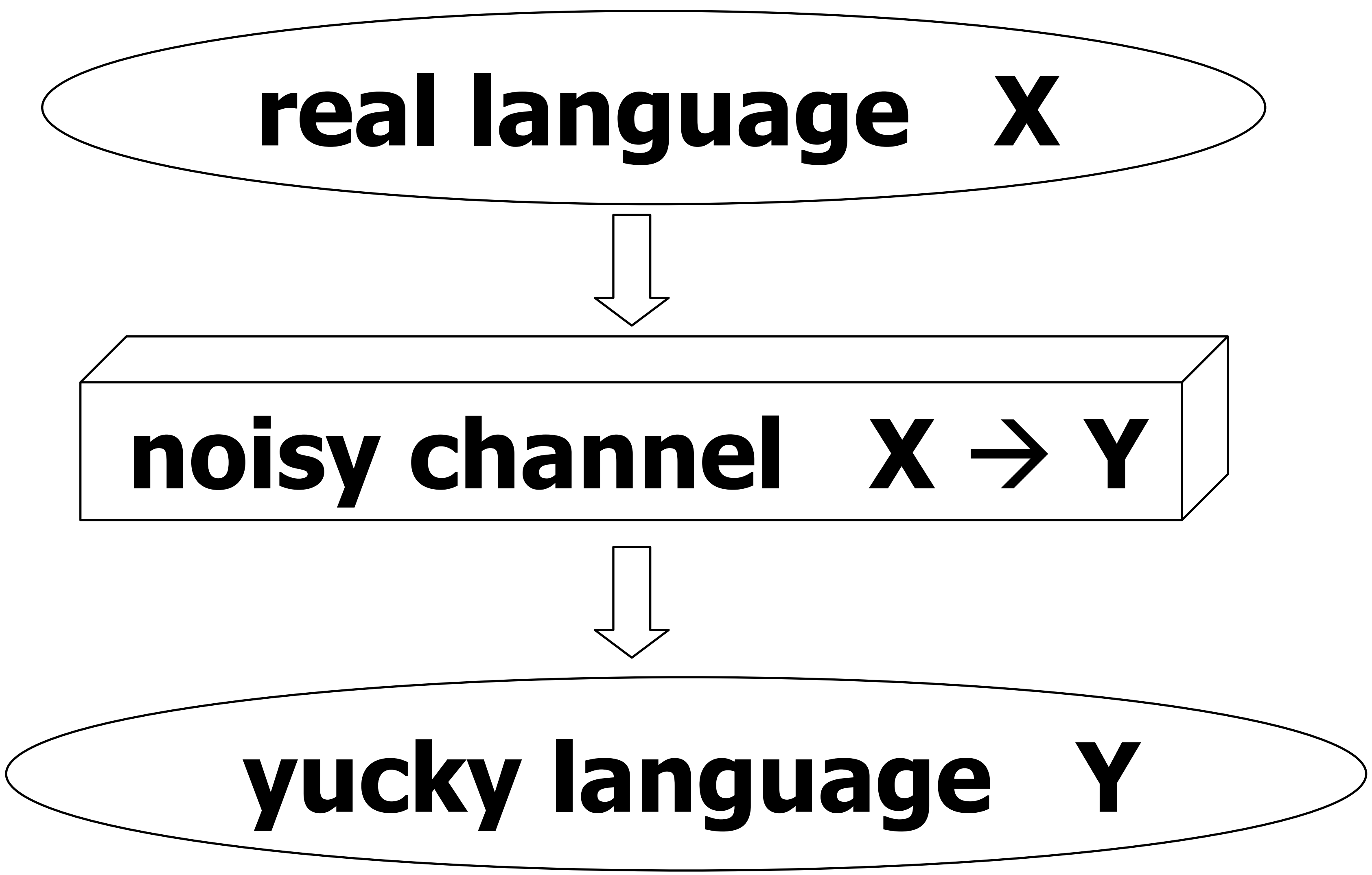
$p(Y \mid X)$

$=$

$p(X,Y)$

want to recover $x \in X$ from $y \in Y$

Noisy Channel Model



$$p(X)$$

$$*$$

$$p(Y \mid X)$$

$$=$$

$$p(X,Y)$$

want to recover $x \in X$ from $y \in Y$

choose x that maximizes $p(x \mid y)$ or equivalently $p(x,y)$

Noisy Channel Model

$$p(X)$$

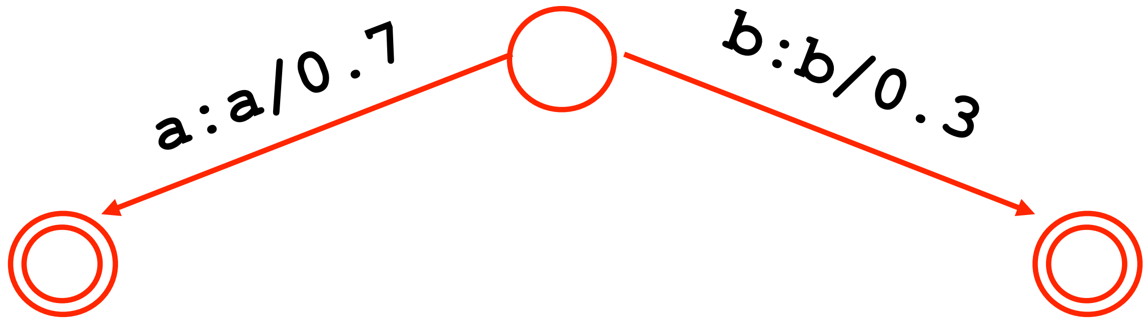
$$*$$

$$p(Y \mid X)$$

$$=$$

$$p(X,Y)$$

Noisy Channel Model



$$p(X)$$

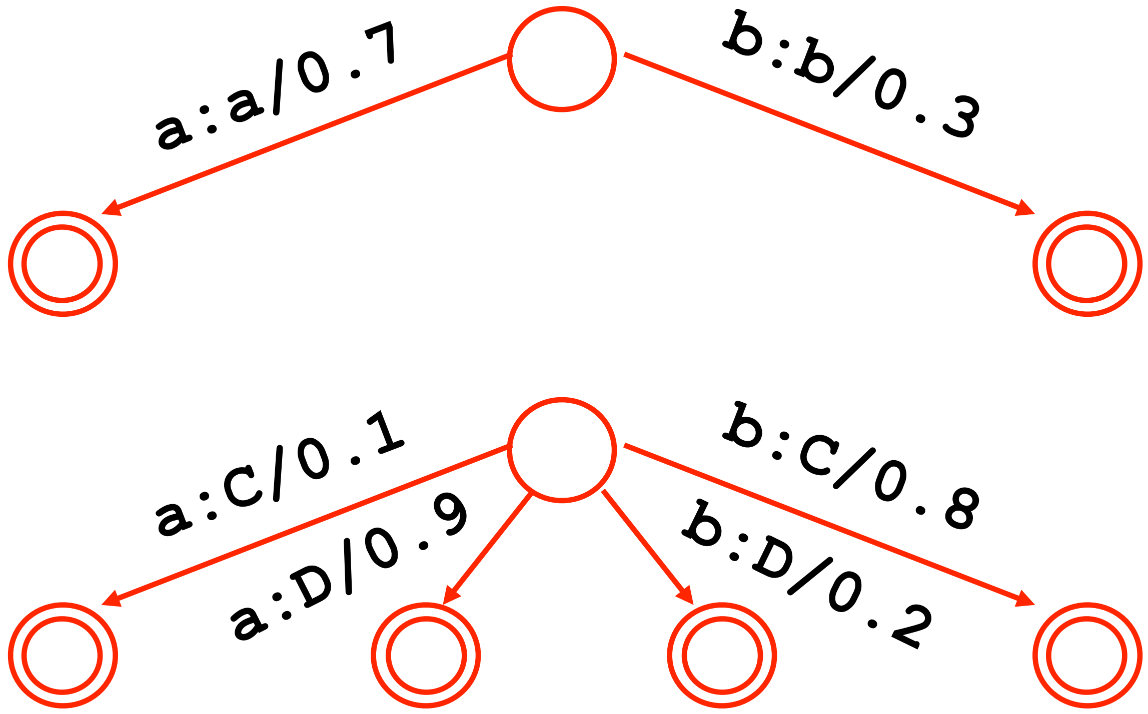
$$*$$

$$p(Y \mid X)$$

$$=$$

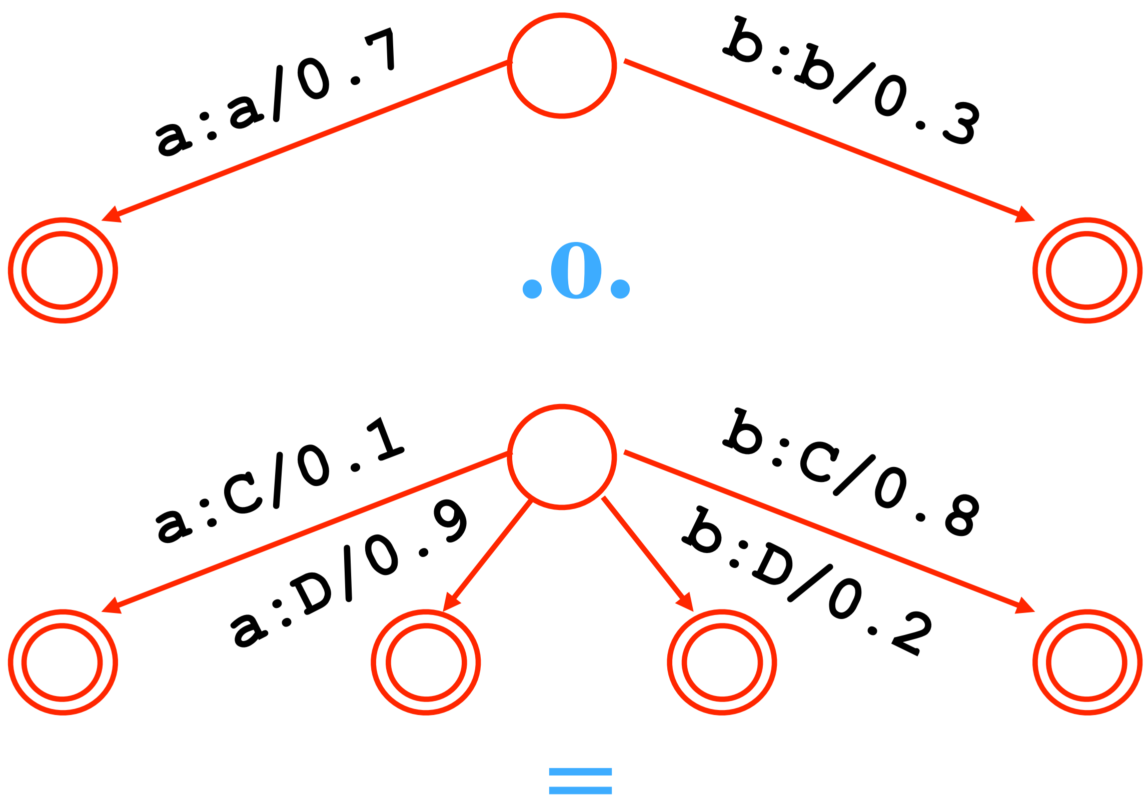
$$p(X,Y)$$

Noisy Channel Model



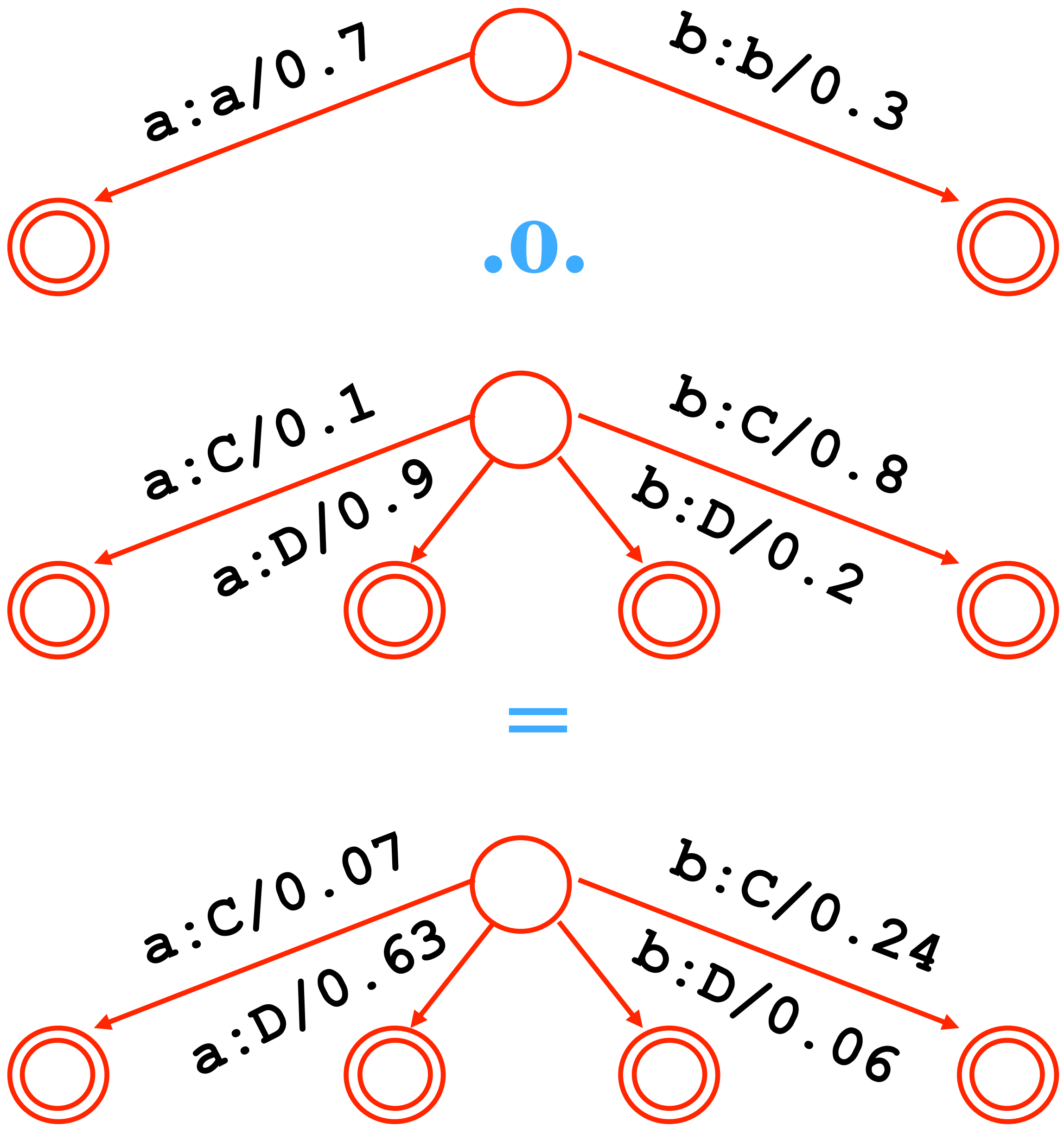
$$p(X) * p(Y \mid X) = p(X,Y)$$

Noisy Channel Model



$$p(X) * p(Y \mid X) = p(X,Y)$$

Noisy Channel Model



p(X)

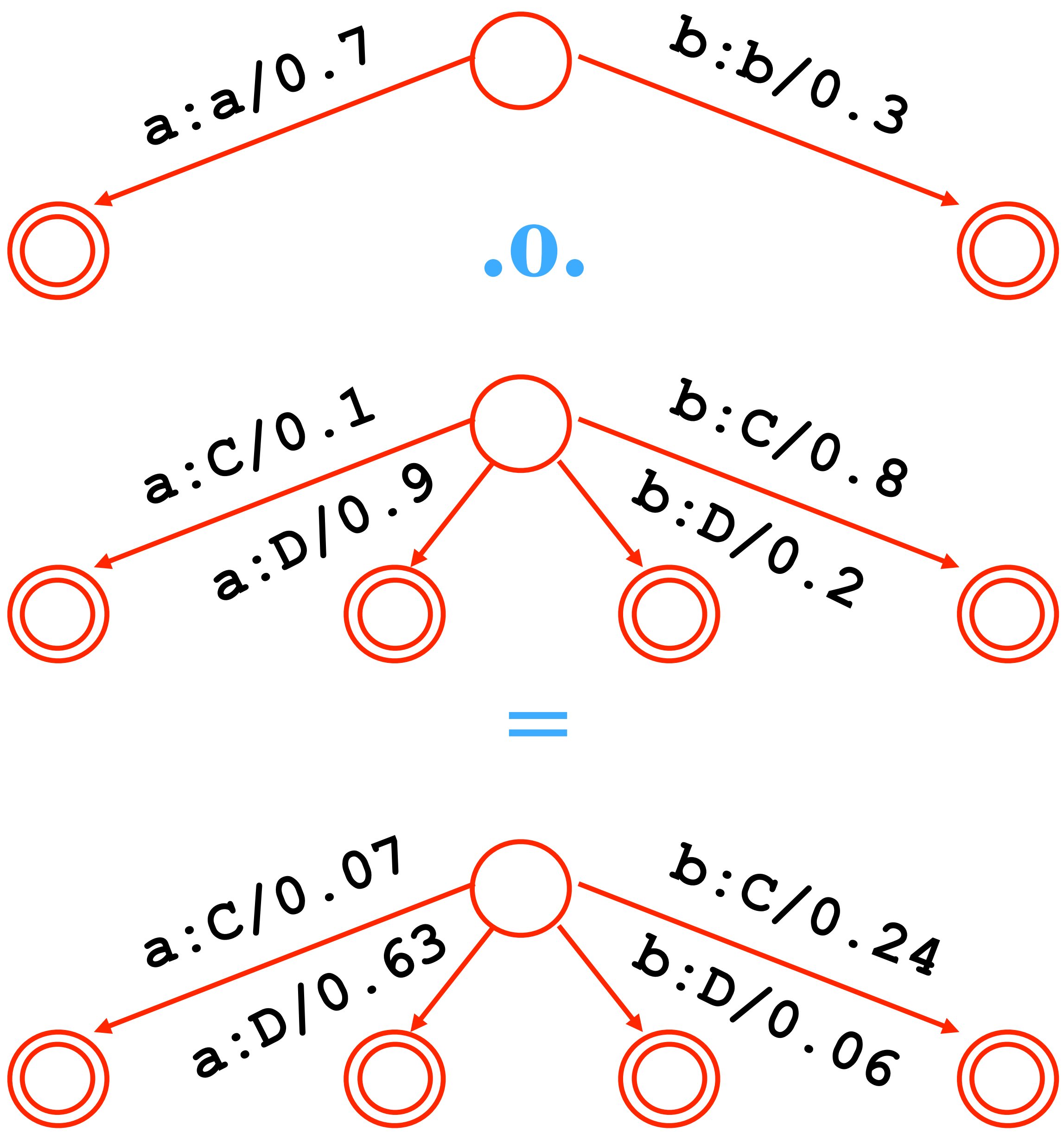
*

p(Y | X)

=

p(X,Y)

Noisy Channel Model



p(X)

*

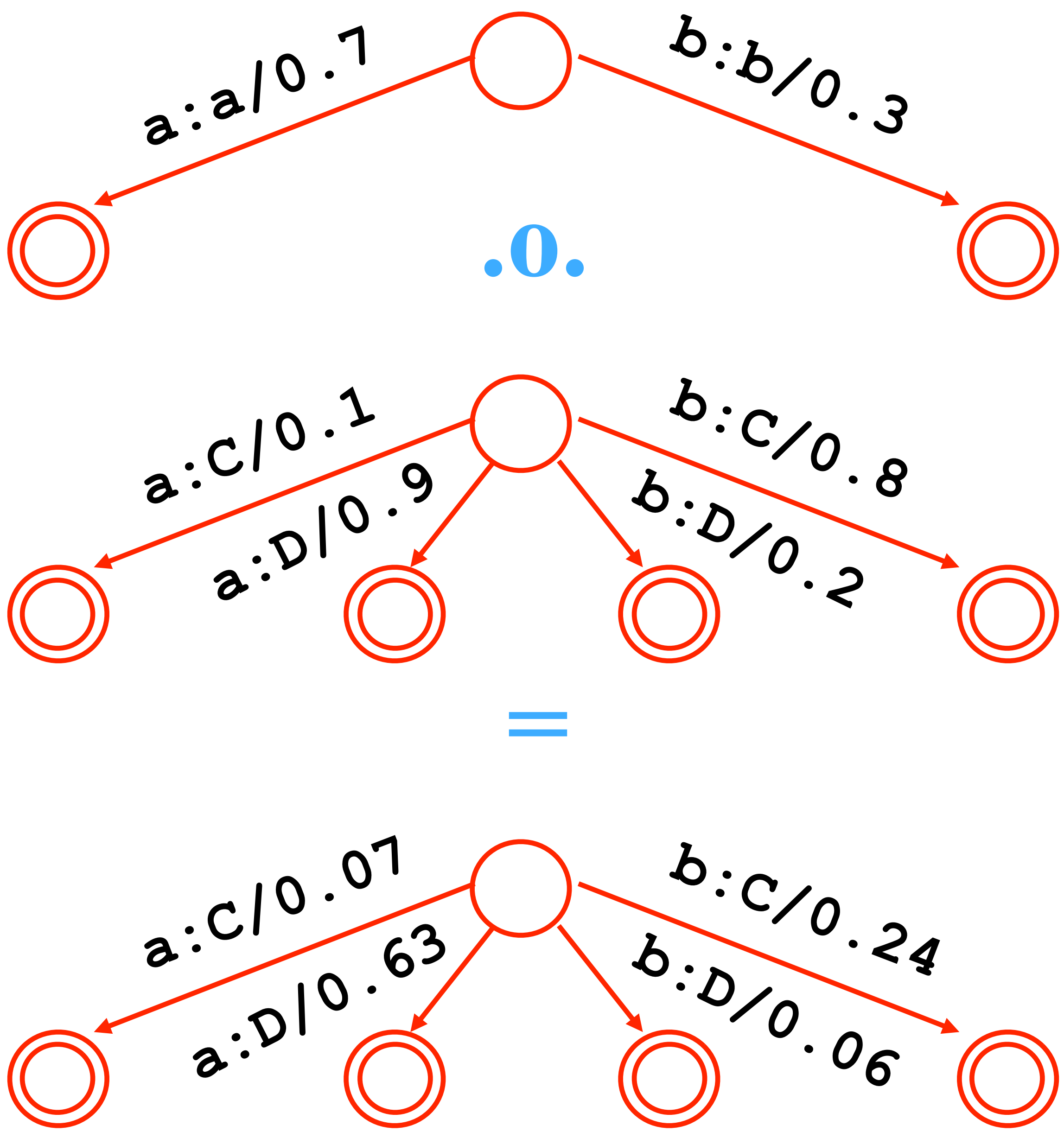
p(Y | X)

=

p(X,Y)

Note p(x,y) sums to 1.

Noisy Channel Model



$p(X)$

$*$

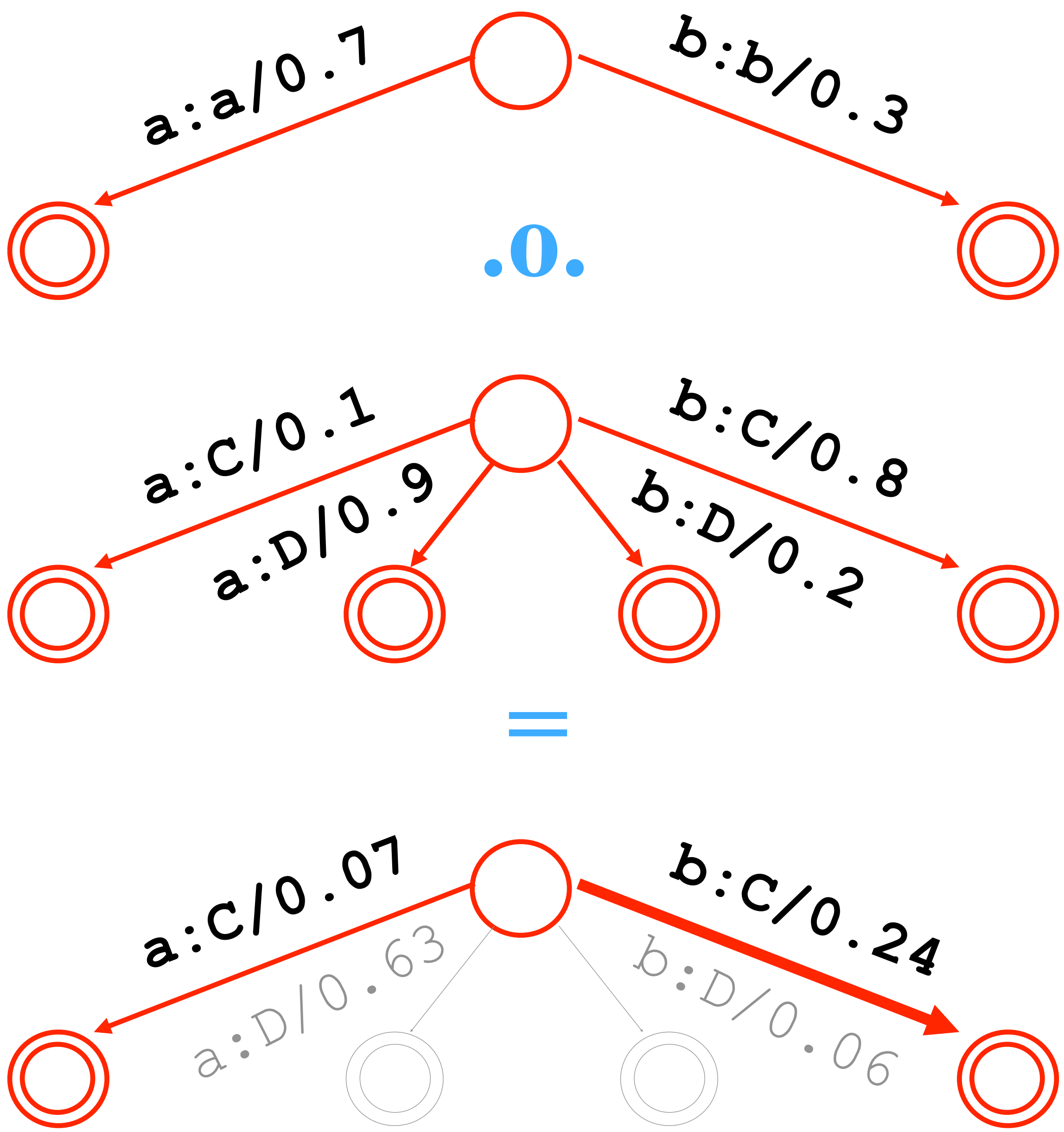
$p(Y \mid X)$

$=$

$p(X,Y)$

Note p(x,y) sums to 1.
Suppose y="C"; what is best "x"?

Noisy Channel Model



p(X)

*

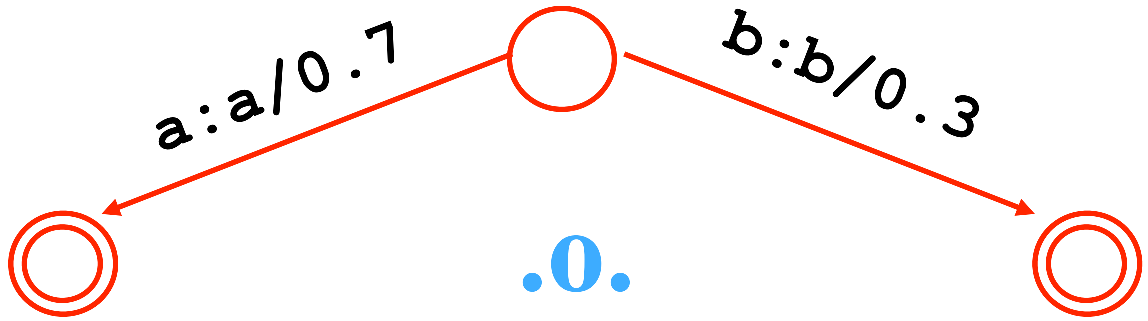
p(Y | X)

=

p(X,Y)

Suppose y="C"; what is best "x"?

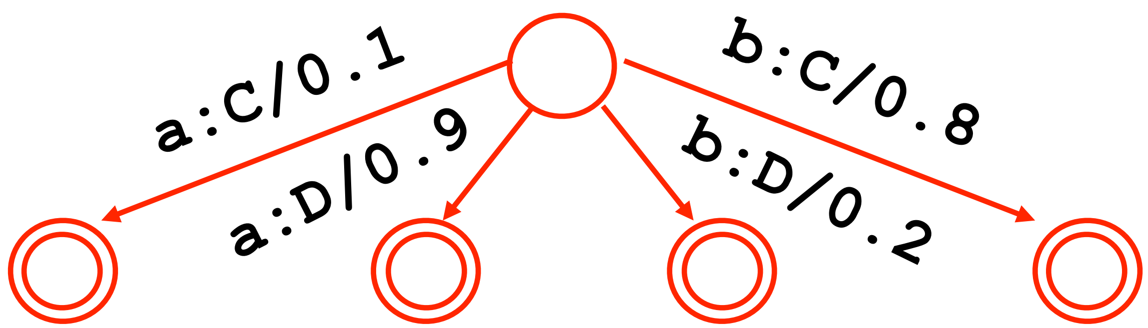
Noisy Channel Model



$p(X)$

.o.

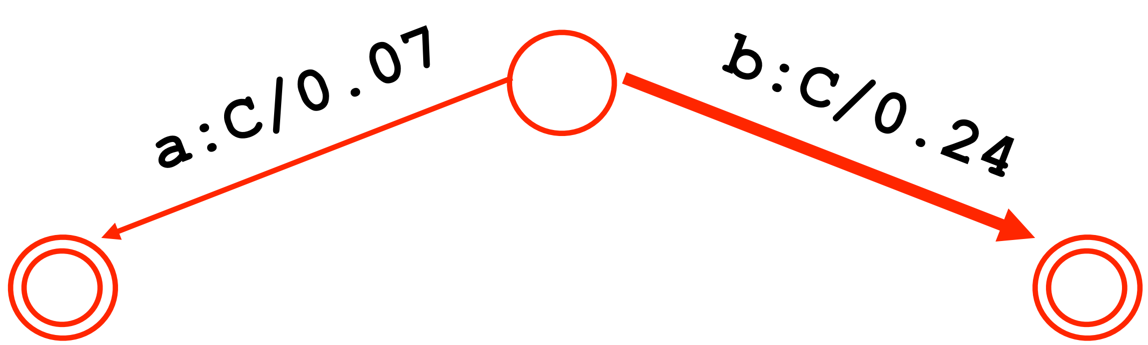
$*$



$p(Y \mid X)$

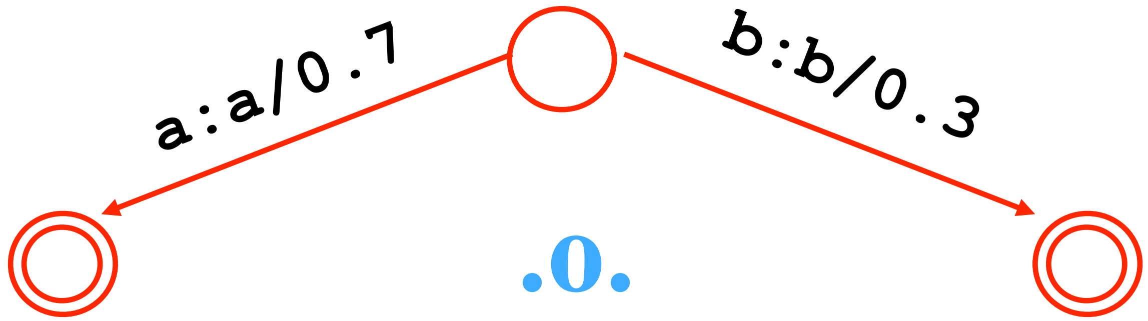
=

=

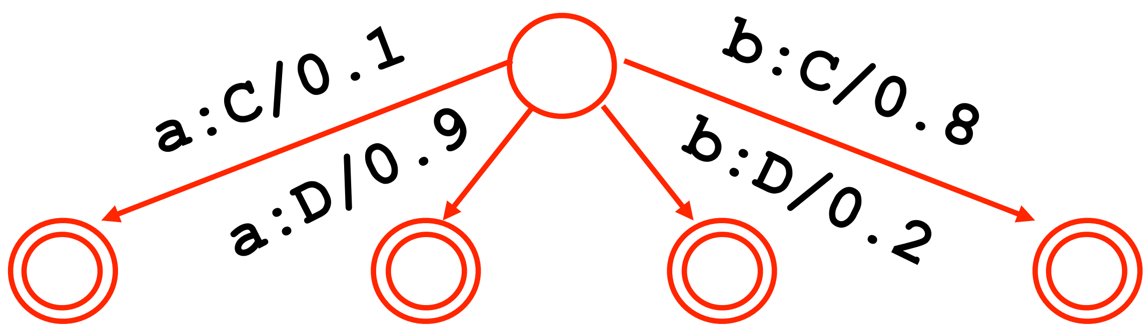


$p(X, y)$

Noisy Channel Model

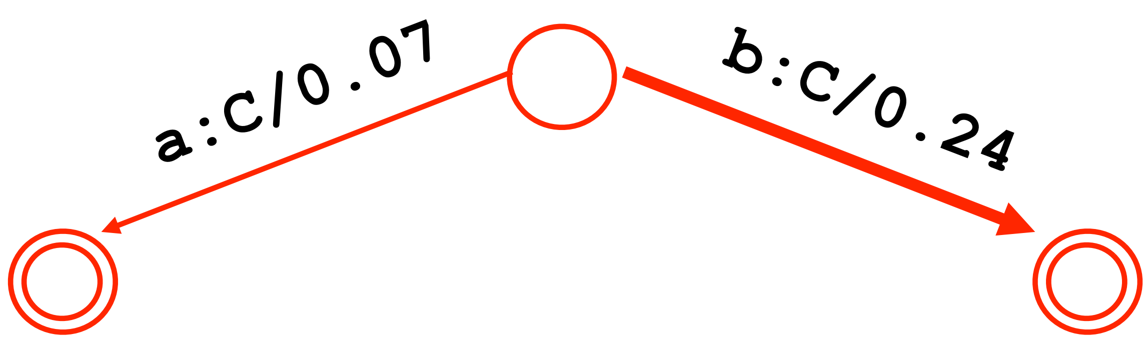


.o.



restrict just to paths compatible with output "C"

=



$p(X)$

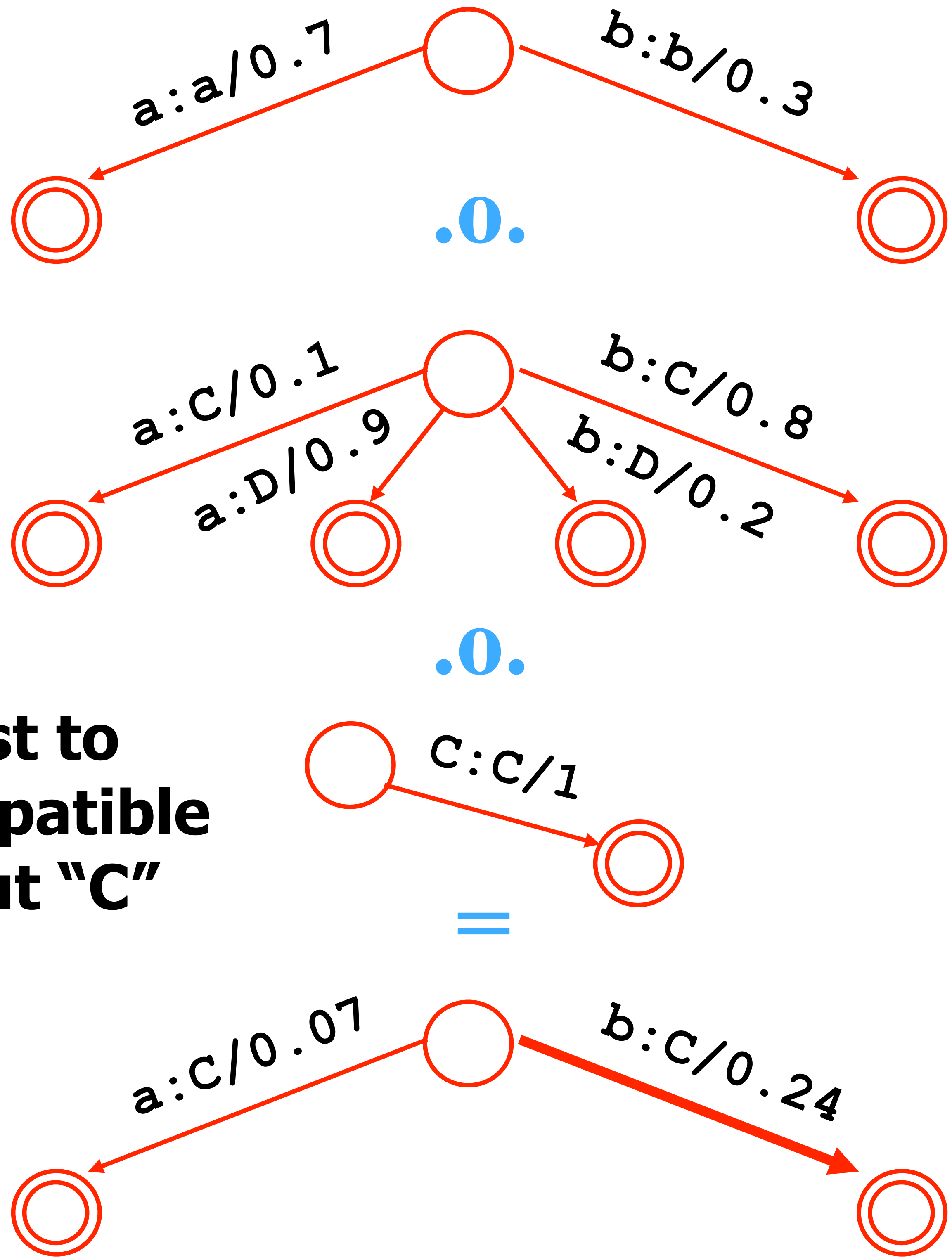
$*$

$p(Y \mid X)$

$=$

$p(X, y)$

Noisy Channel Model



restrict just to paths compatible with output "C"

p(X)

*

p(Y | X)

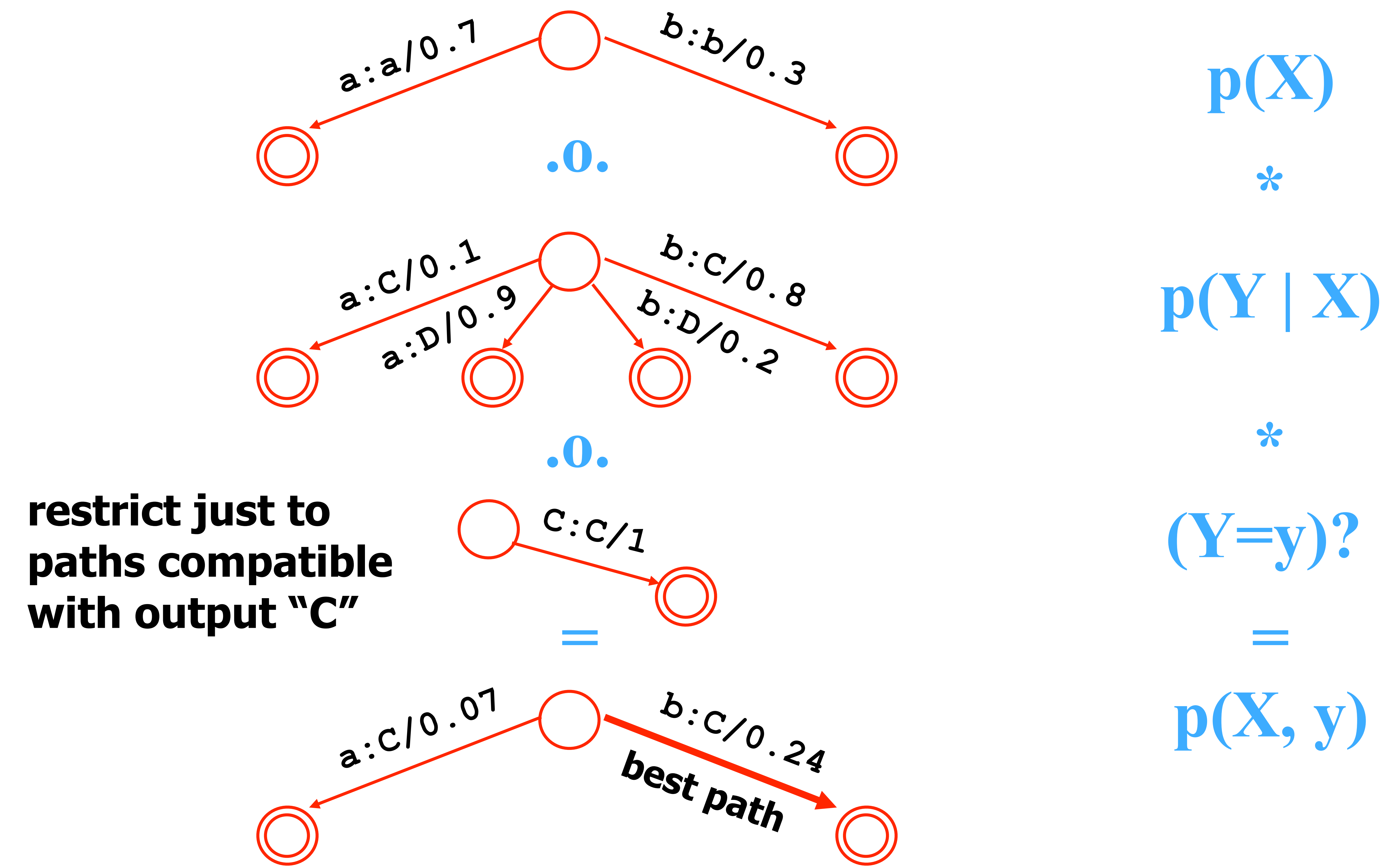
*

(Y=y)?

=

p(X, y)

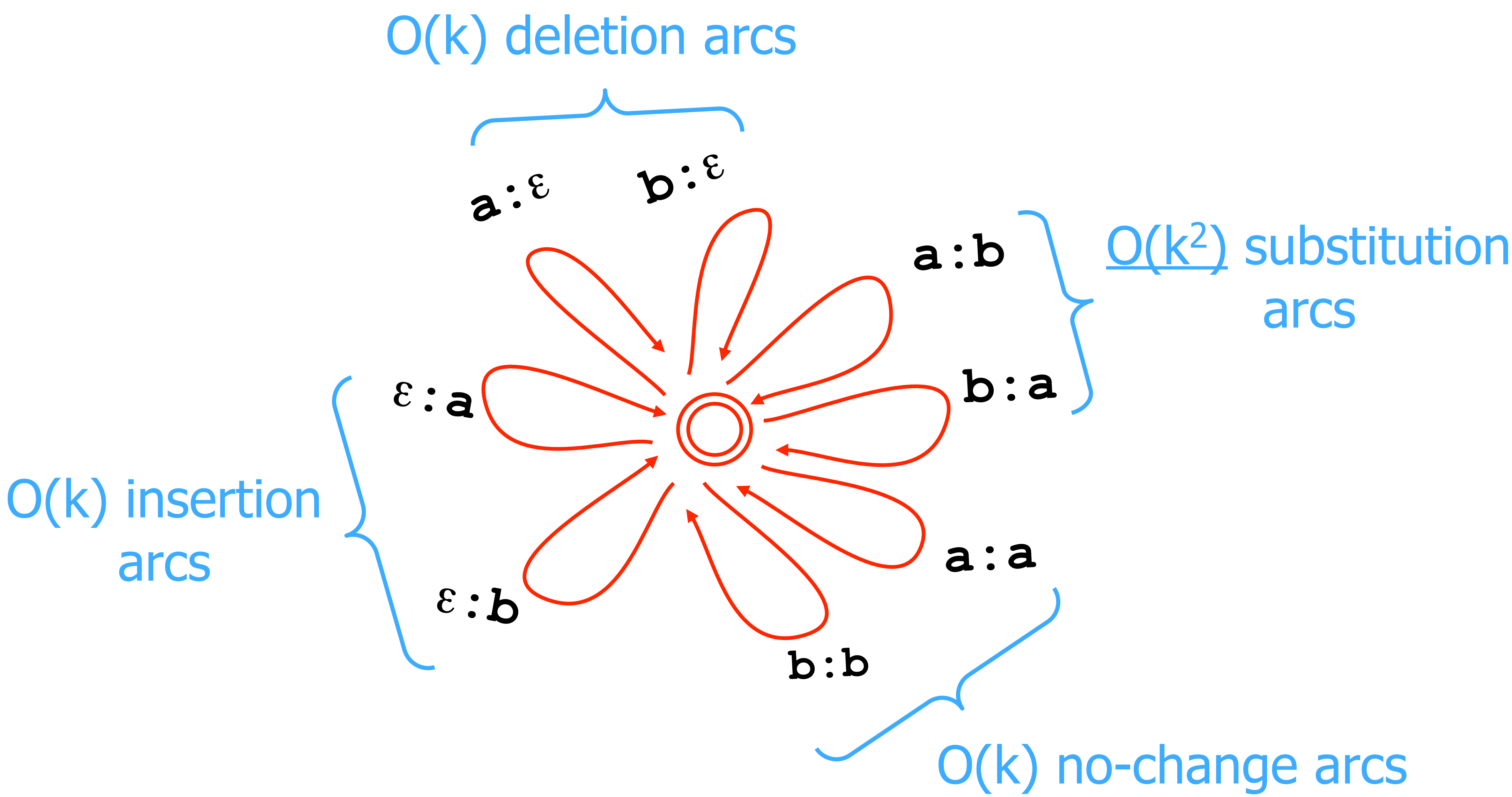
Noisy Channel Model



Morpheme Segmentation

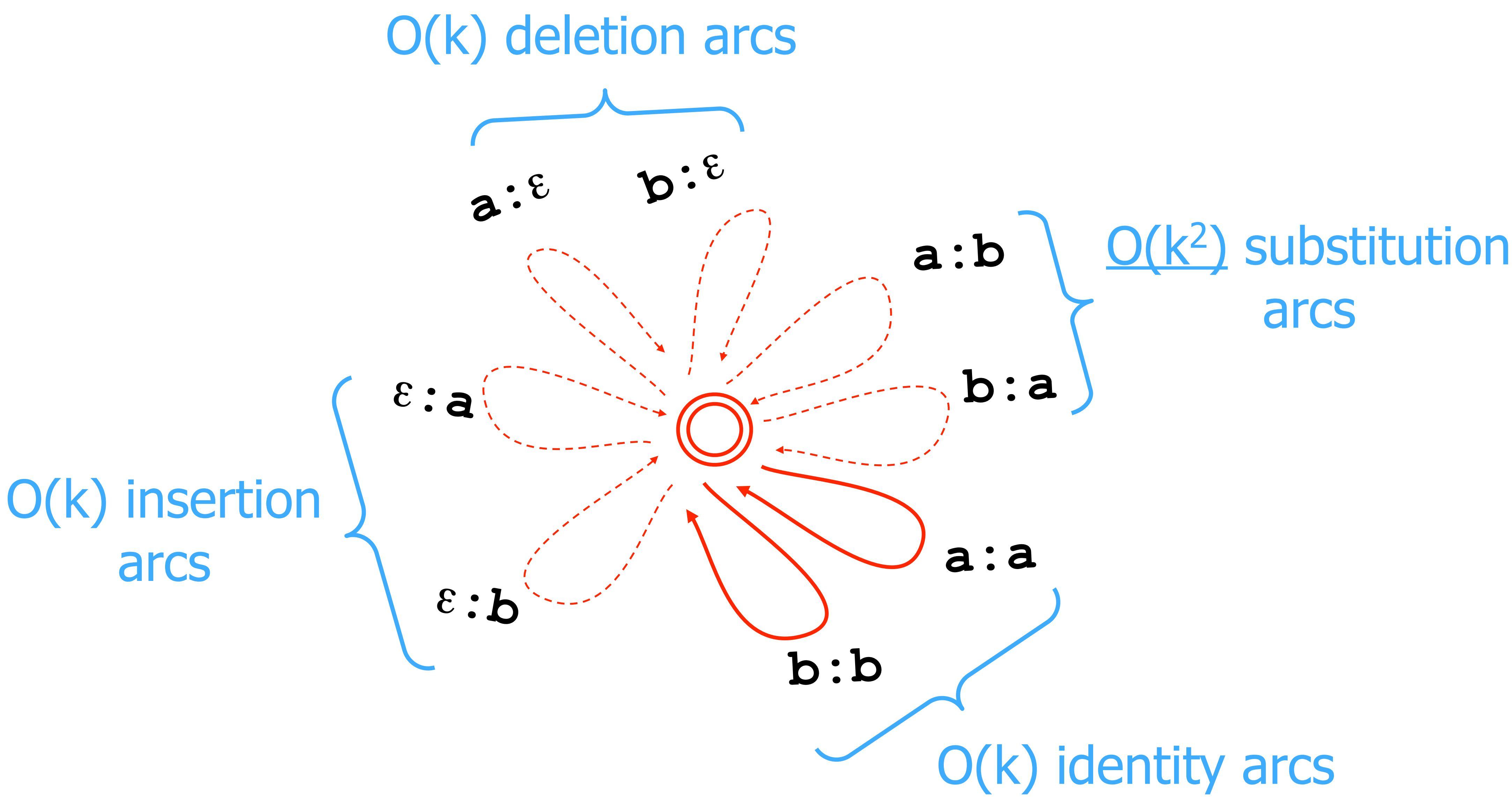
- Let Lexicon be a machine that matches all Turkish words
  - Same problem as word segmentation (in, e.g., Chinese)
  - Just at a lower level: morpheme segmentation
  - Turkish word: uygarlaştıramadıklarımızdanmışsınızcasına
    = uygar+laş+tır+ma+dık+ları+mız+dan+mış+sınız+ca+sı+na
    (behaving) as if you are among those whom we could not cause to become civilized
  - Some constraints on morpheme sequence: bigram probs
  - Generative model – concatenate then fix up joints
    - stop + -ing = stopping, fly + -s = flies, vowel harmony
    - Use a cascade of transducers to handle all the fixups
  - But this is just morphology!
  - Can use probabilities here too (but people often don't)

Edit Distance Transducer



Stochastic

$\wedge$ Edit Distance Transducer

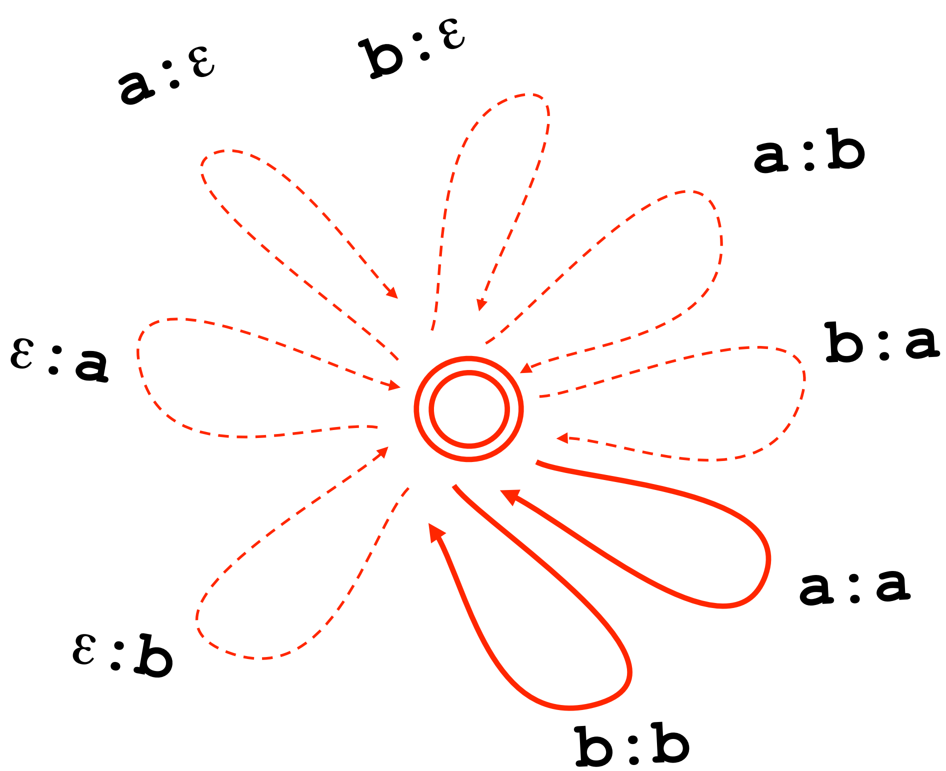


Likely edits = high-probability arcs

Stochastic
Edit Distance Transducer

clara

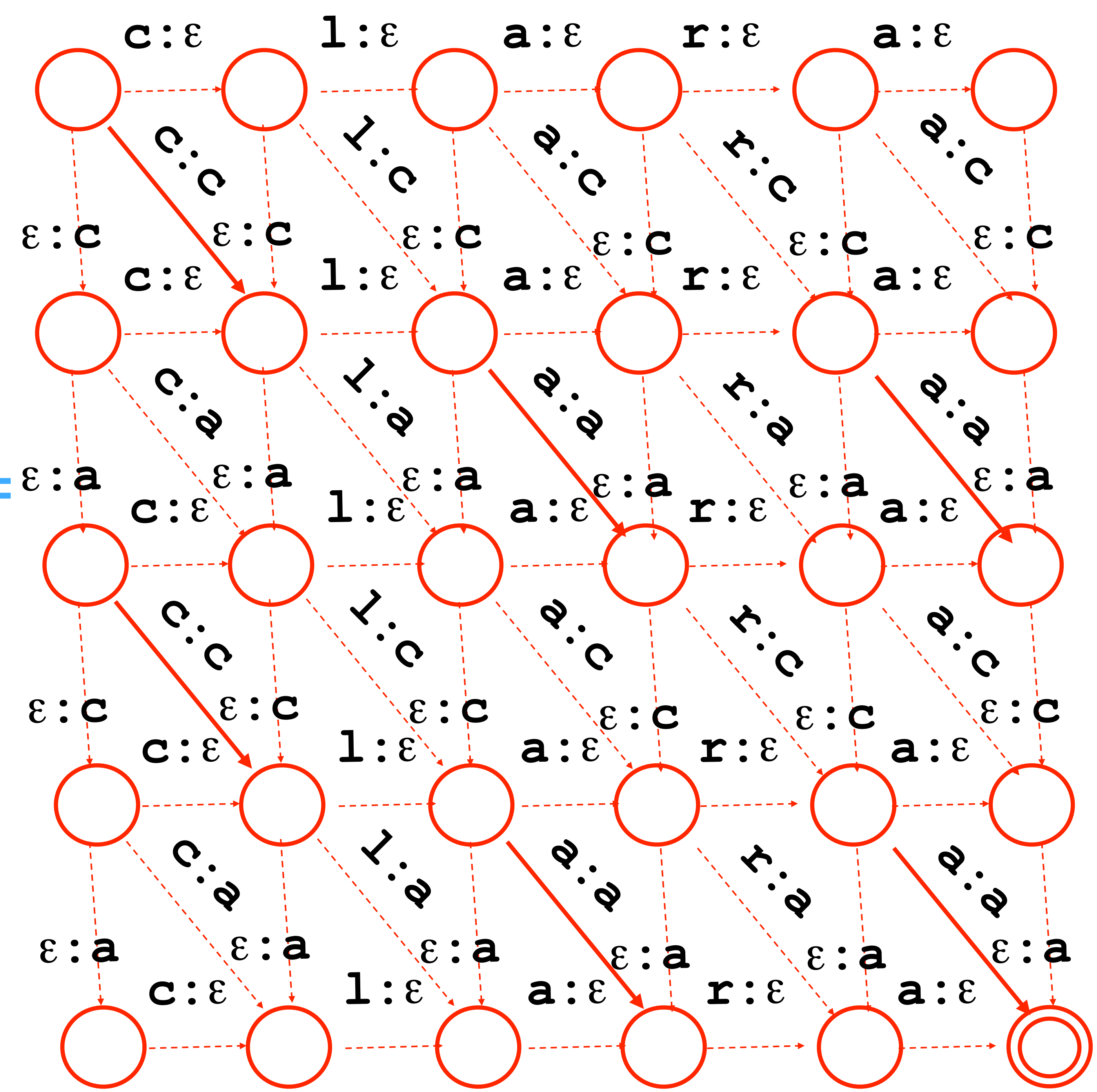
.o.



.o.

caca

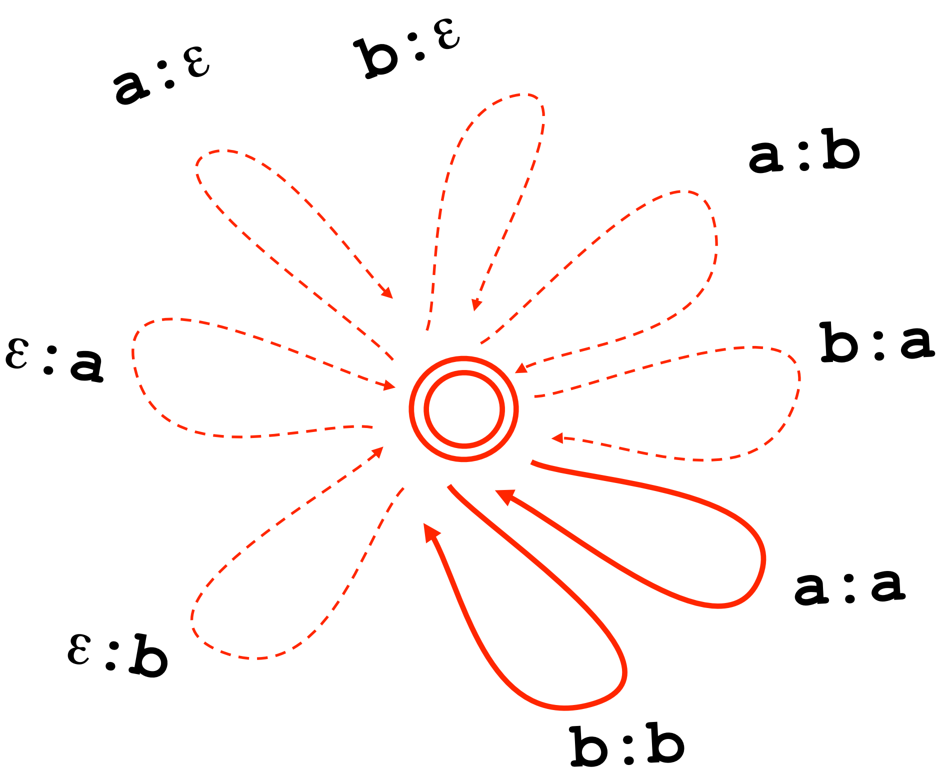
=



Stochastic Edit Distance Transducer

clara

.o.

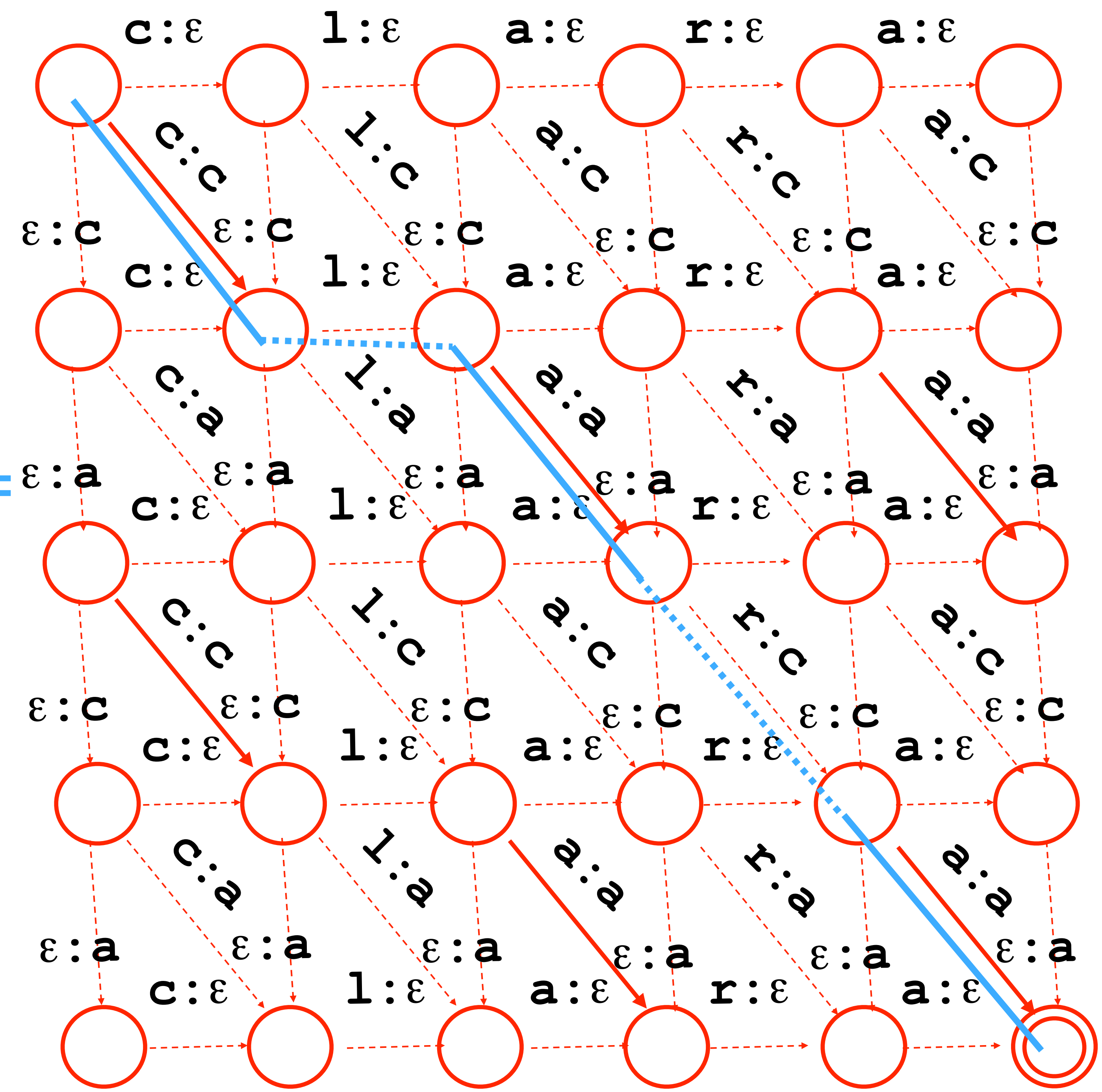


.o.

caca

=

Best path (by Dijkstra's algorithm)



Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model	p(word seq)
.o.	
pronunciation model	p(phone seq | word seq)
.o.	
acoustic model	p(acoustics | phone seq)

Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model p(word seq)

.o.

pronunciation model p(phone seq | word seq)

.o.

acoustic model p(acoustics | phone seq)

.o.

observed acoustics

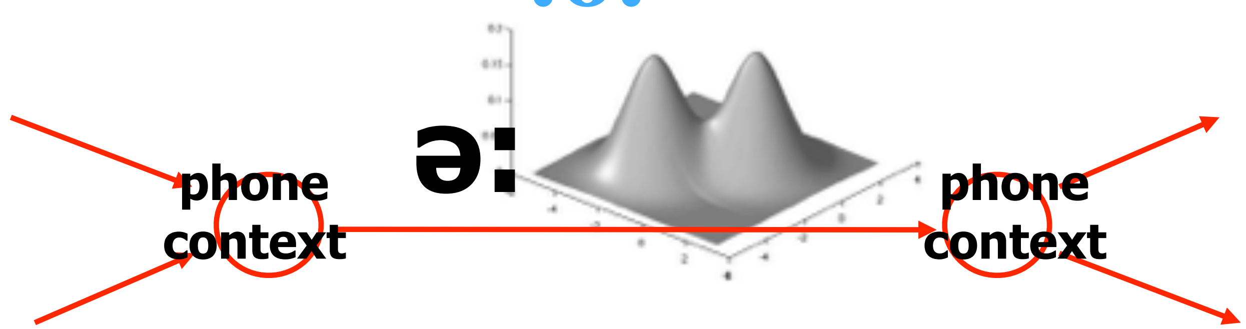
Speech Recognition by FST Composition
(Pereira & Riley 1996)

trigram language model p(word seq)

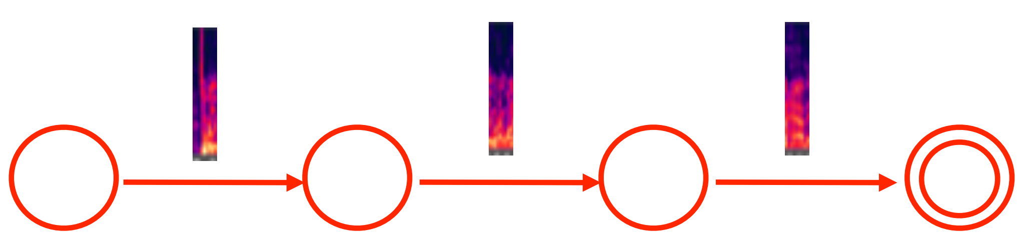
.o.

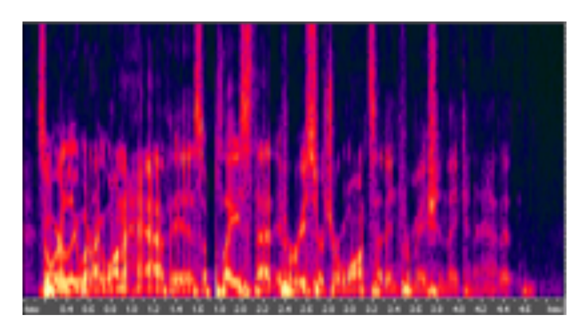
p(phone seq | word seq)

.o.



p(acoustics | phone seq)





Speech Recognition by FST Composition
(Pereira & Riley 1996)

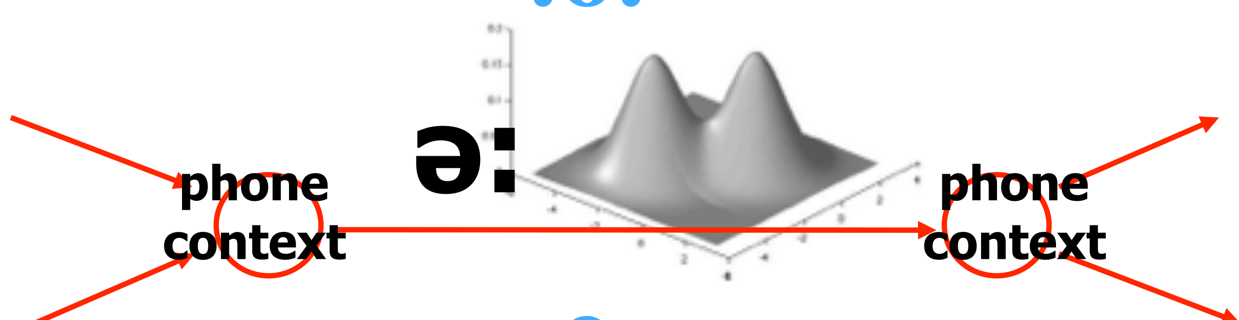
trigram language model

p(word seq)

.o.

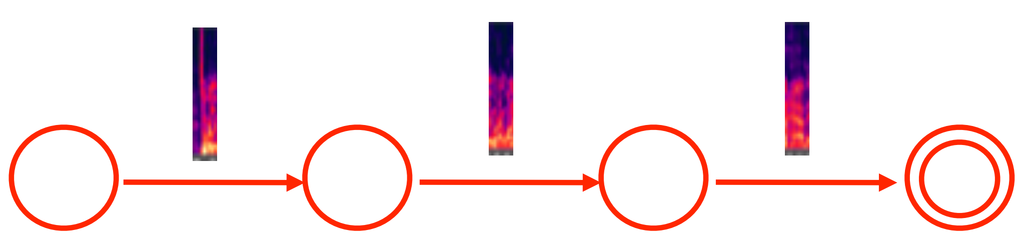
p(phone seq | word seq)

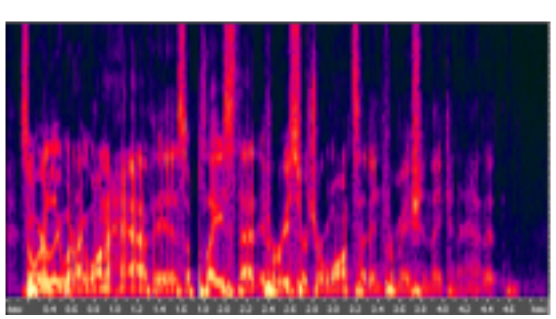
.o.



p(acoustics | phone seq)

.o.





Transliteration (Knight & Graehl, 1998)

Angela Johnson
アンジラ・ジョンソン
(a n jira jyo n so n)

New York Times
ニューヨーク・タイムズ
(nyu u yo o ku ta i mu zu)

ice cream
アイスクリーム
(a i su ku ri i mu)

Omaha Beach
オマハビーチ
(omahabiitchi)

pro soccer
プロサッカー
(purosakkaa)

Tonya Harding
トーニャ・ハーディング
(toonya haadingu)

ramp
ランプ
(ranpu)

lamp
ランプ
(ranpu)

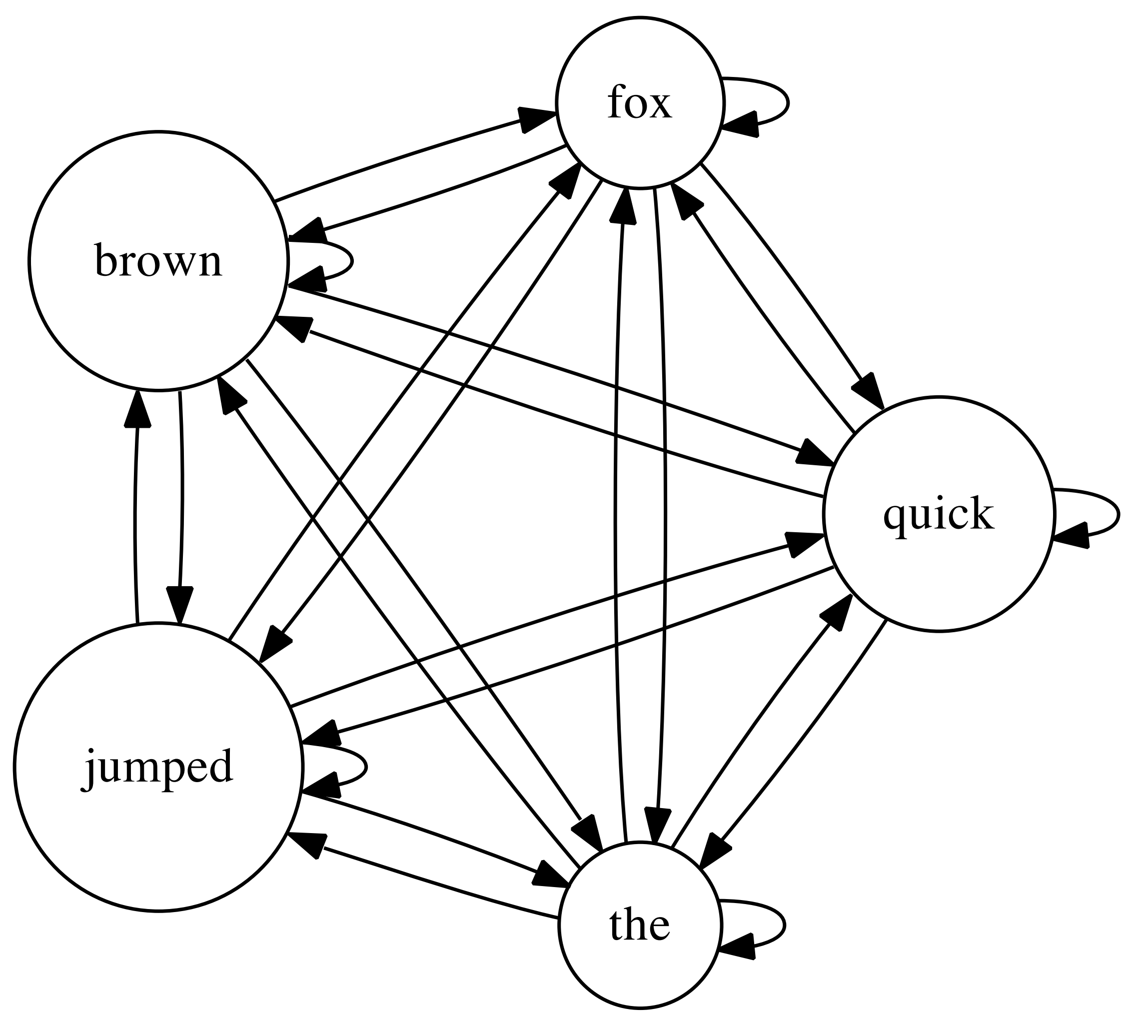
casual fashion
カジュアルヒァッション
(kajyuaruhasshyon)

team leader
チームリーダー
(chiimuriidaa)

1. P(w) — generates written English word sequences.
2. P($e|w$) — pronounces English word sequences.
3. P($j|e$) — converts English sounds into Japanese sounds.
4. P($k|j$) — converts Japanese sounds to katakana writing.
5. P($o|k$) — introduces misspellings caused by optical character recognition (OCR).

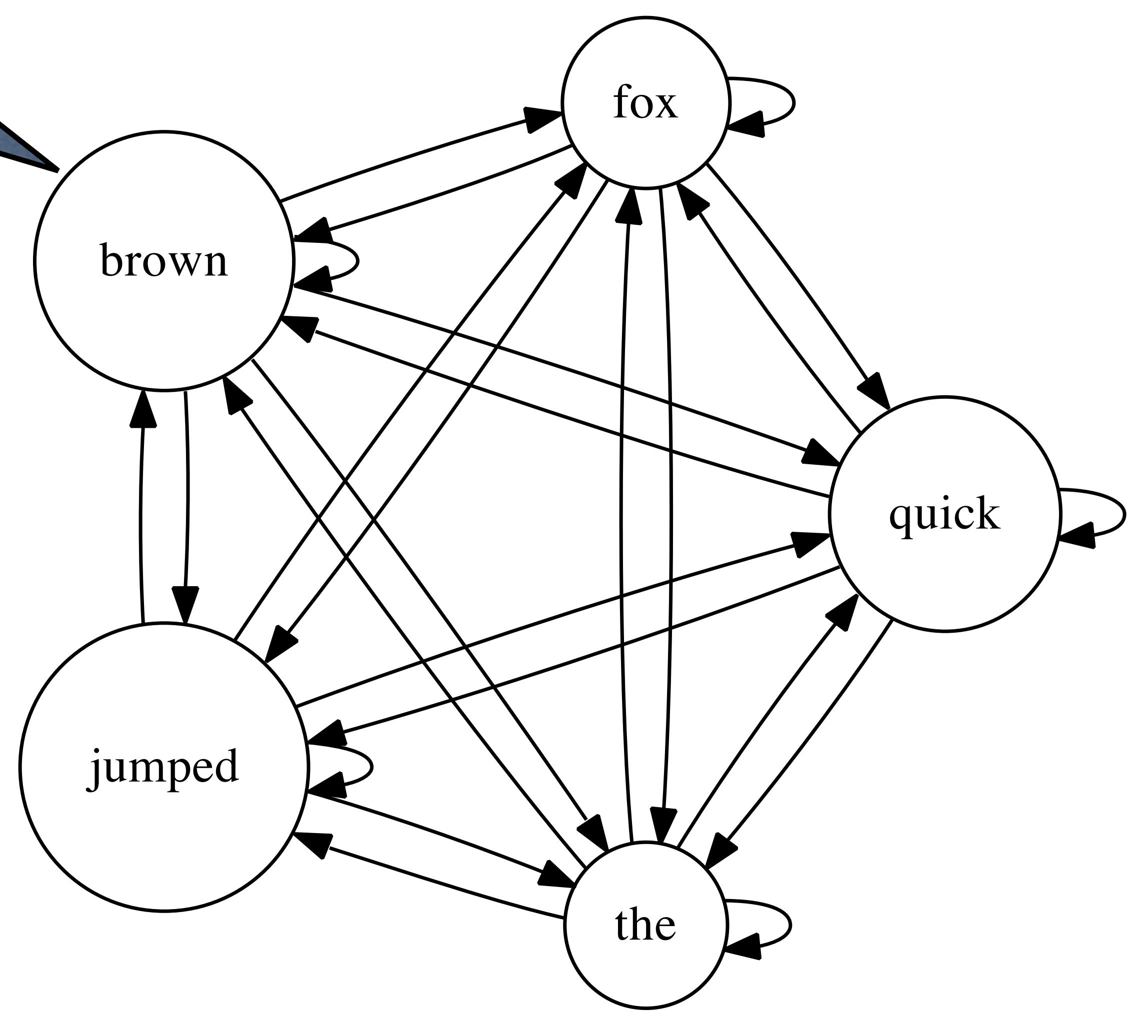
Part-of-Speech Tagging

Bigram LM as FSM



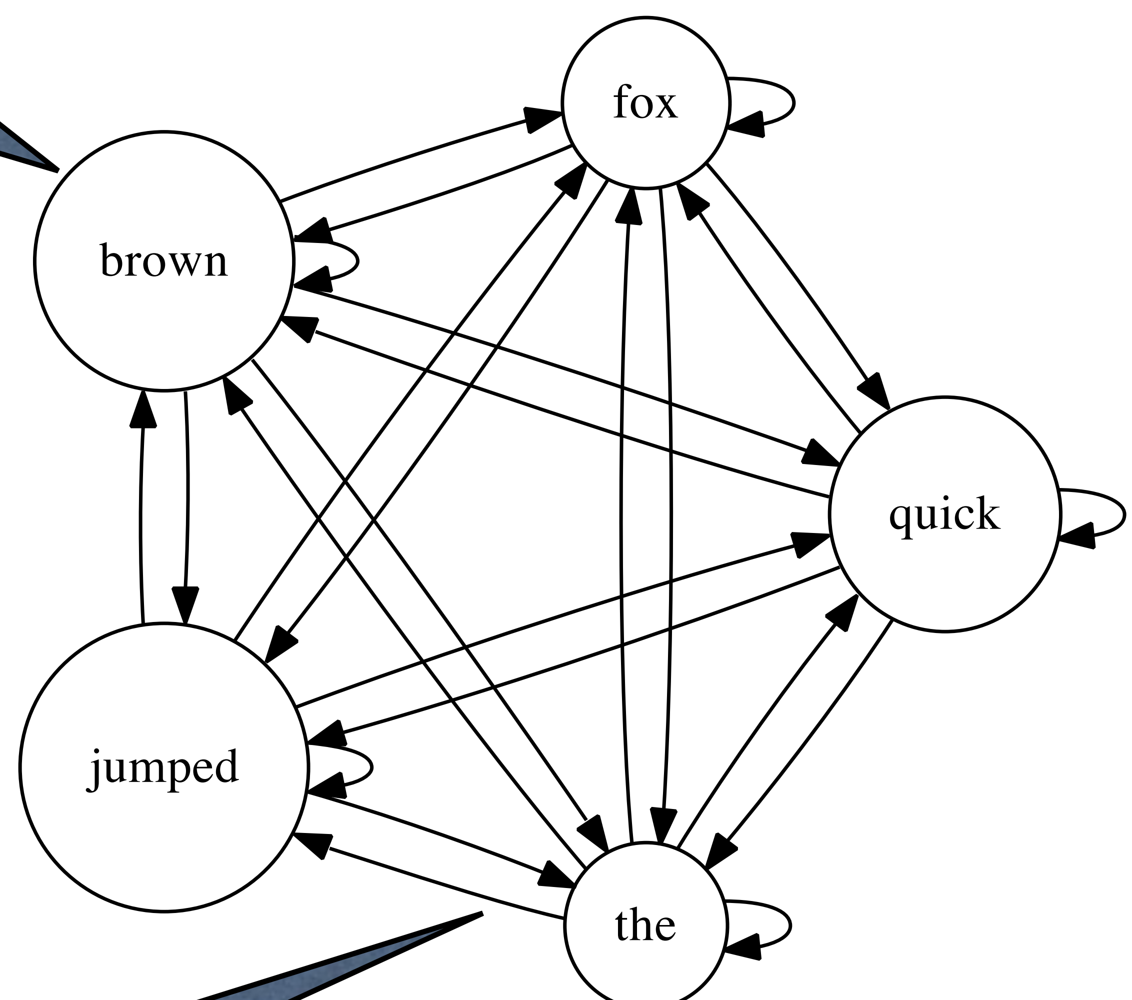
Bigram LM as FSM

V states



Bigram LM as FSM

V states

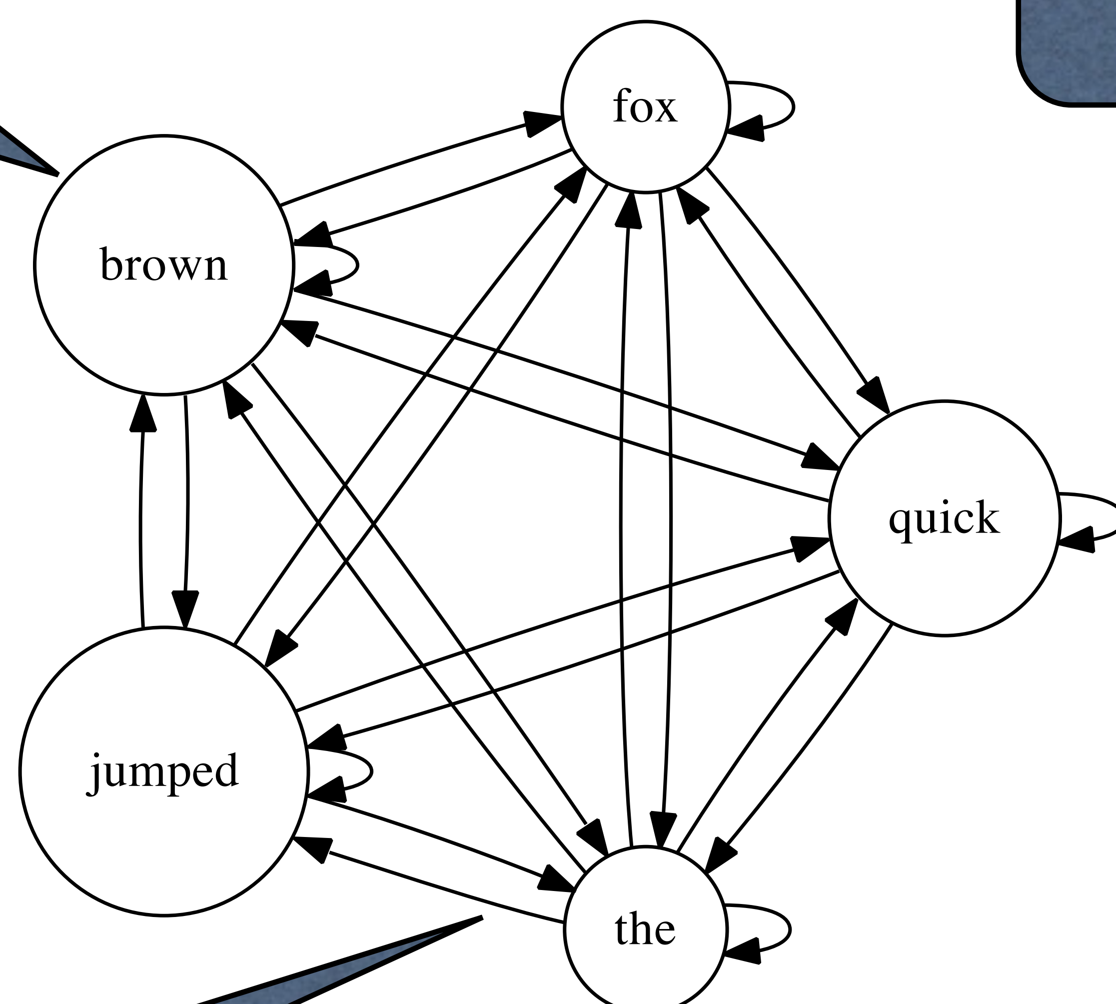


O(V^2) arcs
(& parameters)

Bigram LM as FSM

V states

What about a trigram model?

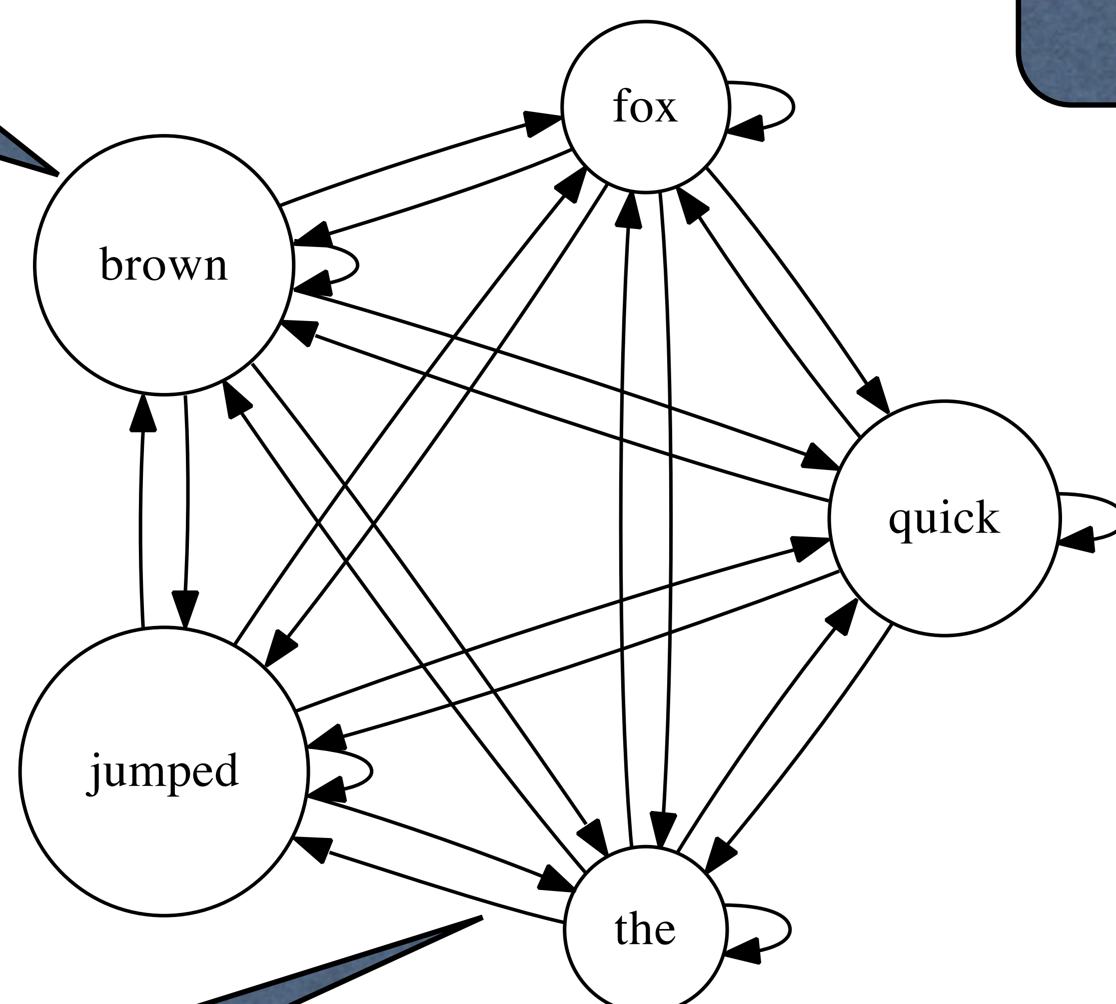


$O(V^2)$ arcs
(& parameters)

Bigram LM as FSM

V states

What about a trigram model?



O(V^2) arcs
(& parameters)

What about backoff?

Grammatical Categories

- "Parts of speech" (partes orationis)
  - Some Cool Kids call them "word classes"
- Folk definitions
  - Nouns: people, places, concepts, things, ...
  - Verbs: expressive of action
  - Adjectives: properties of nouns
- In linguistics, defined by role in syntax

$$\text{The} \left\{ \begin{array}{l} \text{sad} \\ \text{intelligent} \\ \text{green} \\ \text{fat} \\ \ldots \end{array} \right\} \text{one is in the corner.}$$

"Substitution test"

The Tagging Task

The Tagging Task

Input: `the lead paint is unsafe`

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- Uses:

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- Uses:
  - text-to-speech (how do we pronounce “lead”?)

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- Uses:
  - text-to-speech (how do we pronounce "lead"?)
  - can write regexps like `(Det) Adj* N+` over the output

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- Uses:
  - text-to-speech (how do we pronounce "lead"?)
  - can write regexps like `(Det) Adj* N+` over the output
  - preprocessing to speed up parser (but a little dangerous)

The Tagging Task

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- Uses:
  - text-to-speech (how do we pronounce “lead”?)
  - can write regexps like `(Det) Adj* N+` over the output
  - preprocessing to speed up parser (but a little dangerous)
  - if you know the tag, you can back off to it in other tasks

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task
- Been done to death by different methods

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)
  - Can be done well with methods that look at local context

Why Do We Care?

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- The first statistical NLP task
- Been done to death by different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task (in English)
  - Can be done well with methods that look at local context
  - Though should "really" do it by parsing!

Tagged Data Sets

- Brown Corpus
  - Designed to be a representative sample from 1961
    - news, poetry, “belles lettres”, short stories
  - 87 different tags
- Penn Treebank
  - 45 different tags
  - Currently most widely used for English
- Now a paradigm in lots of other languages
  - Chinese Treebank has over 200 tags

Penn Treebank POS Tags

PART-OF-SPEECH	TAG	EXAMPLES
Adjective	JJ	happy, bad
Adjective, comparative	JJR	happier, worse
Adjective, cardinal number	CD	3, fifteen
Adverb	RB	often, particularly
Conjunction, coordination	CC	and, or
Conjunction, subordinating	IN	although, when
Determiner	DT	this, each, other, the, a, some
Determiner, postdeterminer	JJ	many, same
Noun	NN	aircraft, data
Noun, plural	NNS	women, books
Noun, proper, singular	NNP	London, Michael
Noun, proper, plural	NNPS	Australians, Methodists
Pronoun, personal	PRP	you, we, she, it
Pronoun, question	WP	who, whoever
Verb, base present form	VBP	take, live

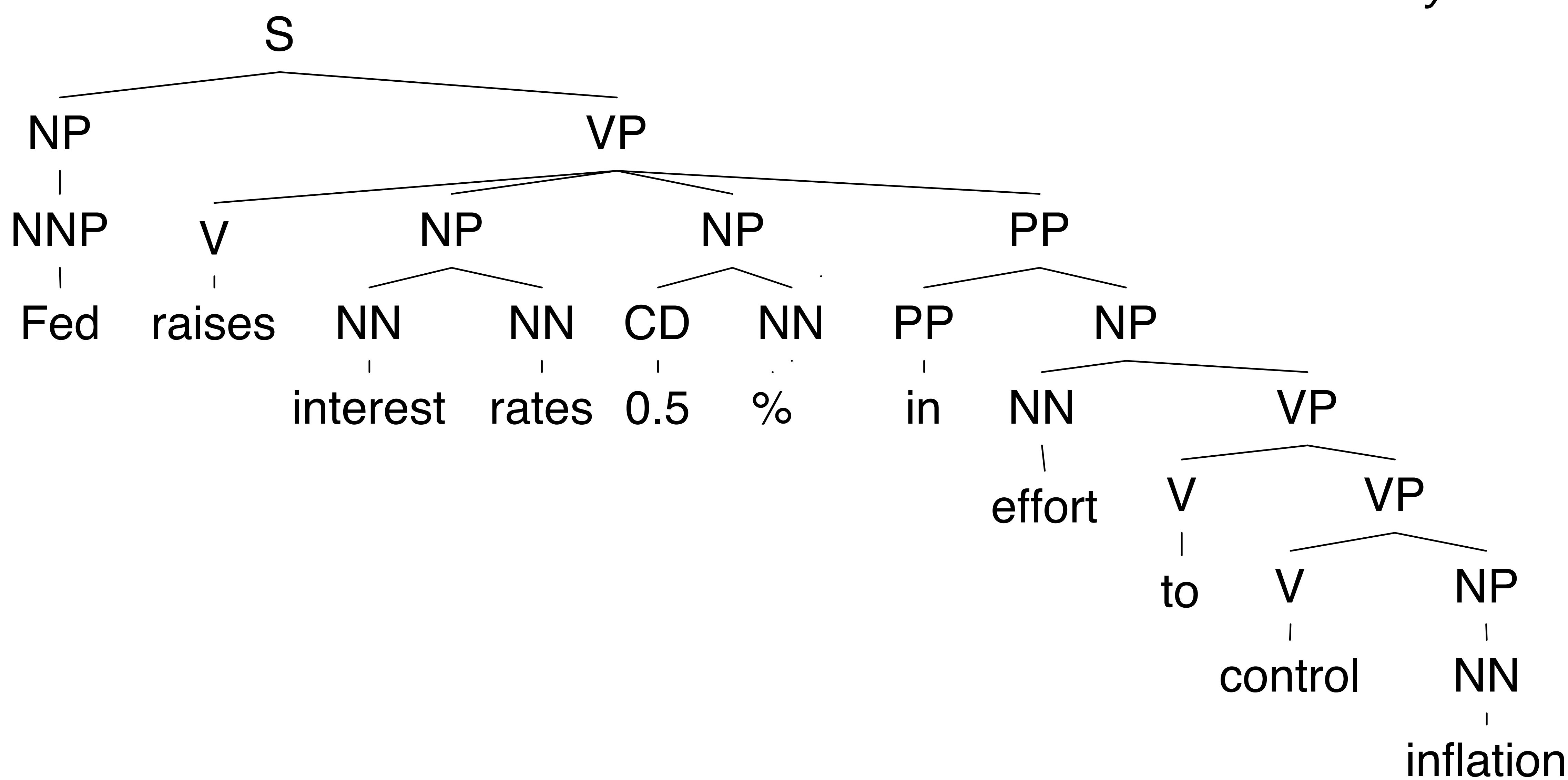
Word Class Classes

- Importantly for predicting POS tags, there are two broad classes
- “Closed class” words
  - Belong to classes that don’t accept new members
  - Determiners: the, a, an, this, ...
  - Prepositions: in, on, of, ...
- “Open class” words
  - Nouns, verbs, adjectives, adverbs, ...
- “Closed” is relative: These words are born and die over longer time scales (e.g, “regarding”)

Ambiguity in Language

```
Fed raises interest rates 0.5%
in effort to control inflation
```

NY Times headline 17 May 2000



Part-of-speech Ambiguity

		VB				
	VBZ	VBZ	VBZ			
NNP	NNS	NNS	NNS	CD	NN	
`Fed`	`raises`	`interest`	`rates`	`0.5`	`%`	`in effort to control inflation`

Degree of Supervision

Degree of Supervision

- Supervised: Training corpus is tagged by humans

Degree of Supervision

- Supervised: Training corpus is tagged by humans
- Unsupervised: Training corpus isn't tagged

Degree of Supervision

- Supervised: Training corpus is tagged by humans
- Unsupervised: Training corpus isn't tagged
- Partly supervised: Training corpus isn't tagged, but you have a dictionary giving possible tags for each word

Degree of Supervision

- Supervised: Training corpus is tagged by humans
- Unsupervised: Training corpus isn't tagged
- Partly supervised: Training corpus isn't tagged, but you have a dictionary giving possible tags for each word

Degree of Supervision

- Supervised: Training corpus is tagged by humans
- Unsupervised: Training corpus isn't tagged
- Partly supervised: Training corpus isn't tagged, but you have a dictionary giving possible tags for each word

- We'll start with the supervised case and move to decreasing levels of supervision.

Current Performance

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

Current Performance

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- How many tags are correct?

Current Performance

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- How many tags are correct?
  - About 97% currently

Current Performance

Input: `the lead paint is unsafe`

Output: `the/Det lead/N paint/N is/V unsafe/Adj`

- How many tags are correct?
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
    - Tag every word with its most frequent tag
    - Tag unknown words as nouns

What Should We Look At?

Bill directed a cortege of autos through the dunes

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

What Should We Look At?

correct tags

Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

Each unknown tag is **constrained** by its word

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

Each unknown tag is **constrained** by its word and by the tags to its immediate left and right.

What Should We Look At?

correct tags

PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

Each unknown tag is **constrained** by its word and by the tags to its immediate left and right. But those tags are unknown too …

What Should We Look At?

correct tags

PN Verb Det Noun Prep Noun Prep Det Noun

Bill directed a cortege of autos through the dunes

PN Adj Det Noun Prep Noun Prep Det Noun

Verb Verb Noun Verb

Adj

Prep

...?

some possible tags for each word (maybe more)

Each unknown tag is **constrained** by its word and by the tags to its immediate left and right. But those tags are unknown too ...

What Should We Look At?

correct tags

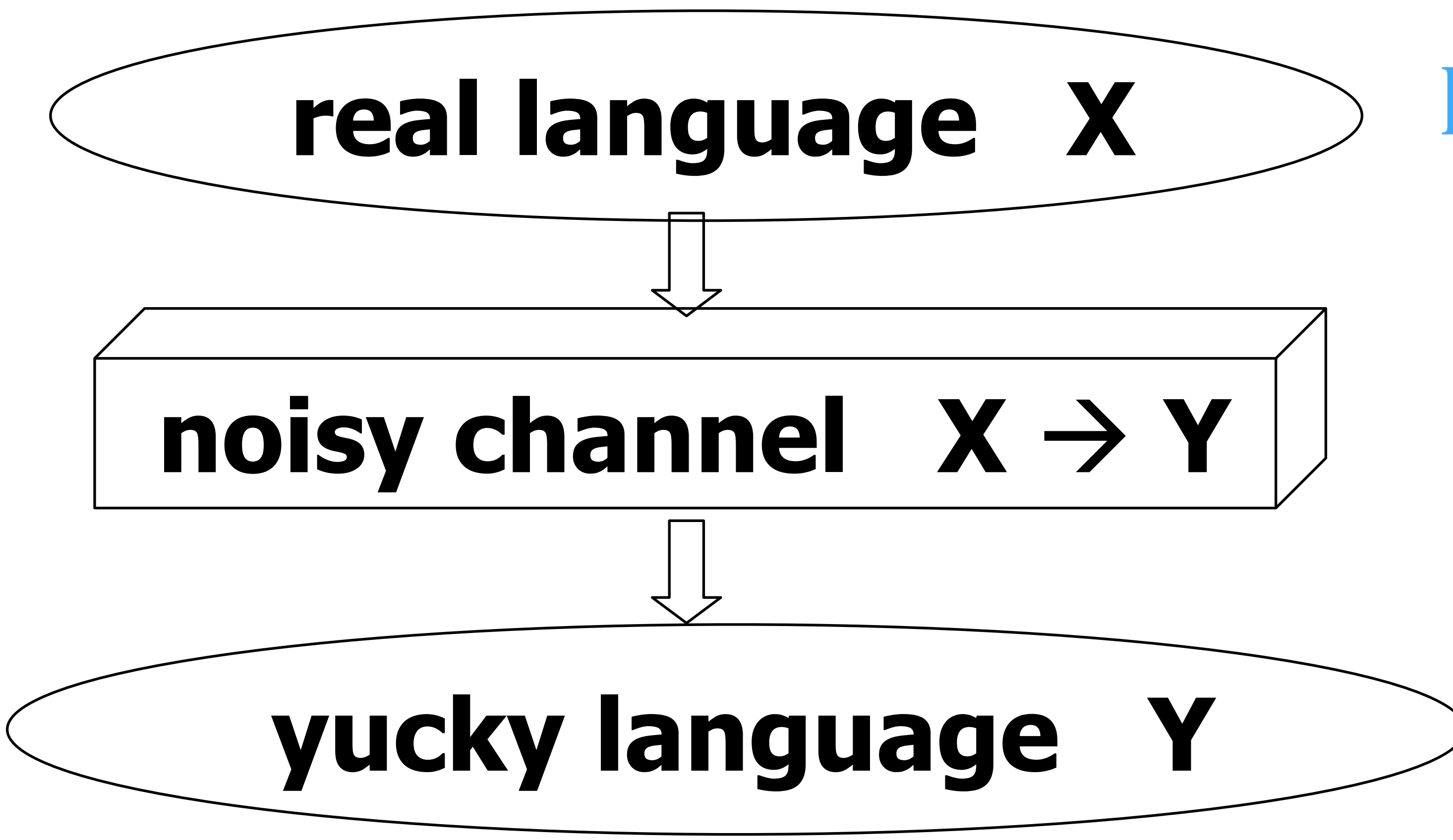
PN	Verb	Det	Noun	Prep	Noun	Prep	Det	Noun
Bill	directed	a	cortege	of	autos	through	the	dunes
PN	Adj	Det	Noun	Prep	Noun	Prep	Det	Noun
Verb	Verb	Noun	Verb					
			Adj					
			Prep					
			...?					

some possible tags for each word (maybe more)

Each unknown tag is **constrained** by its word and by the tags to its immediate left and right. But those tags are unknown too …

Finite-State Approaches

- Noisy Channel Model (statistical)



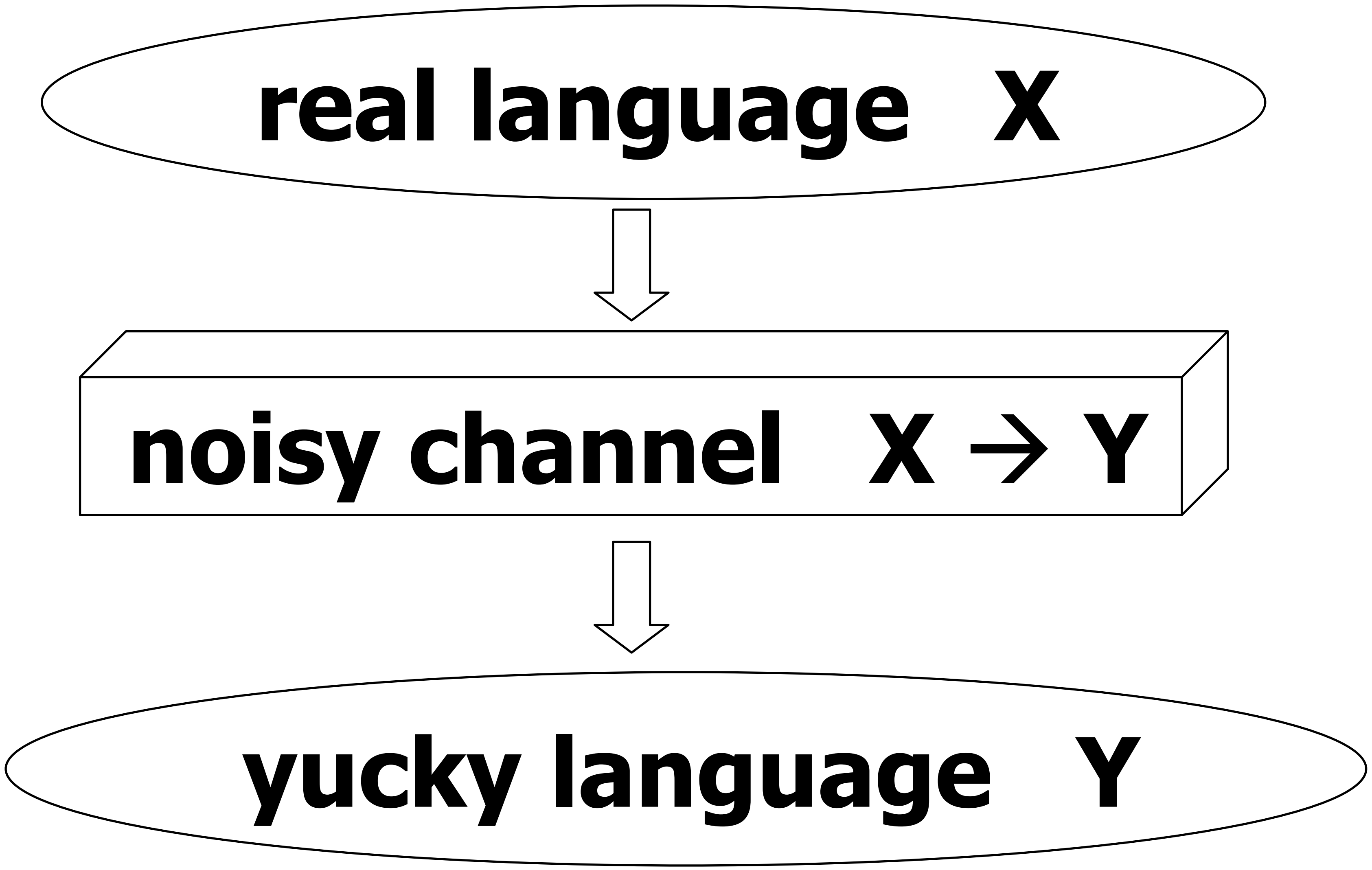
part-of-speech tags
(n-gram model)

replace tags
with words

text

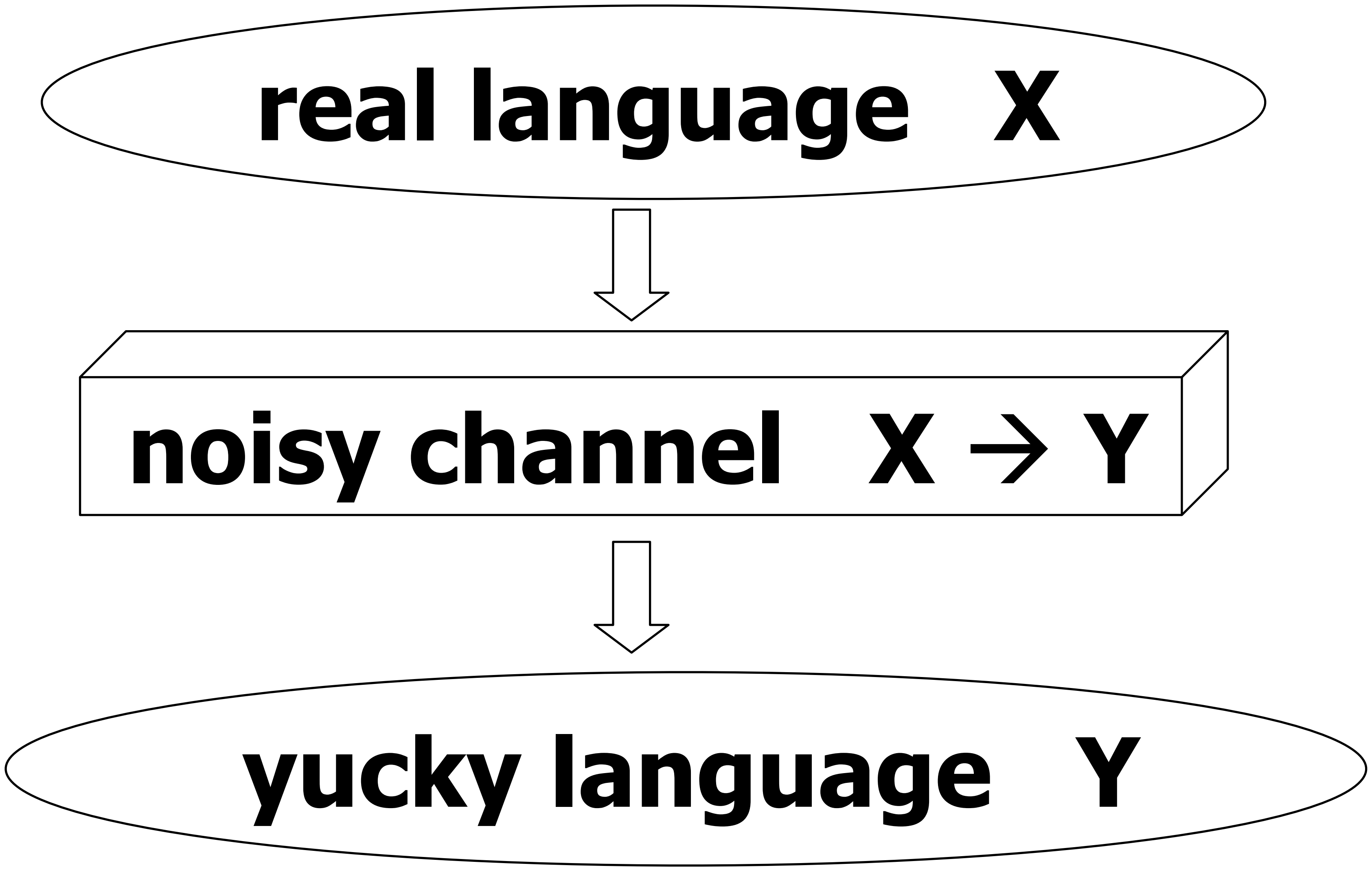
want to recover X from Y

Review: Noisy Channel



$$p(X) * p(Y \mid X) = p(X,Y)$$

Review: Noisy Channel



$$p(X)$$

$$*$$

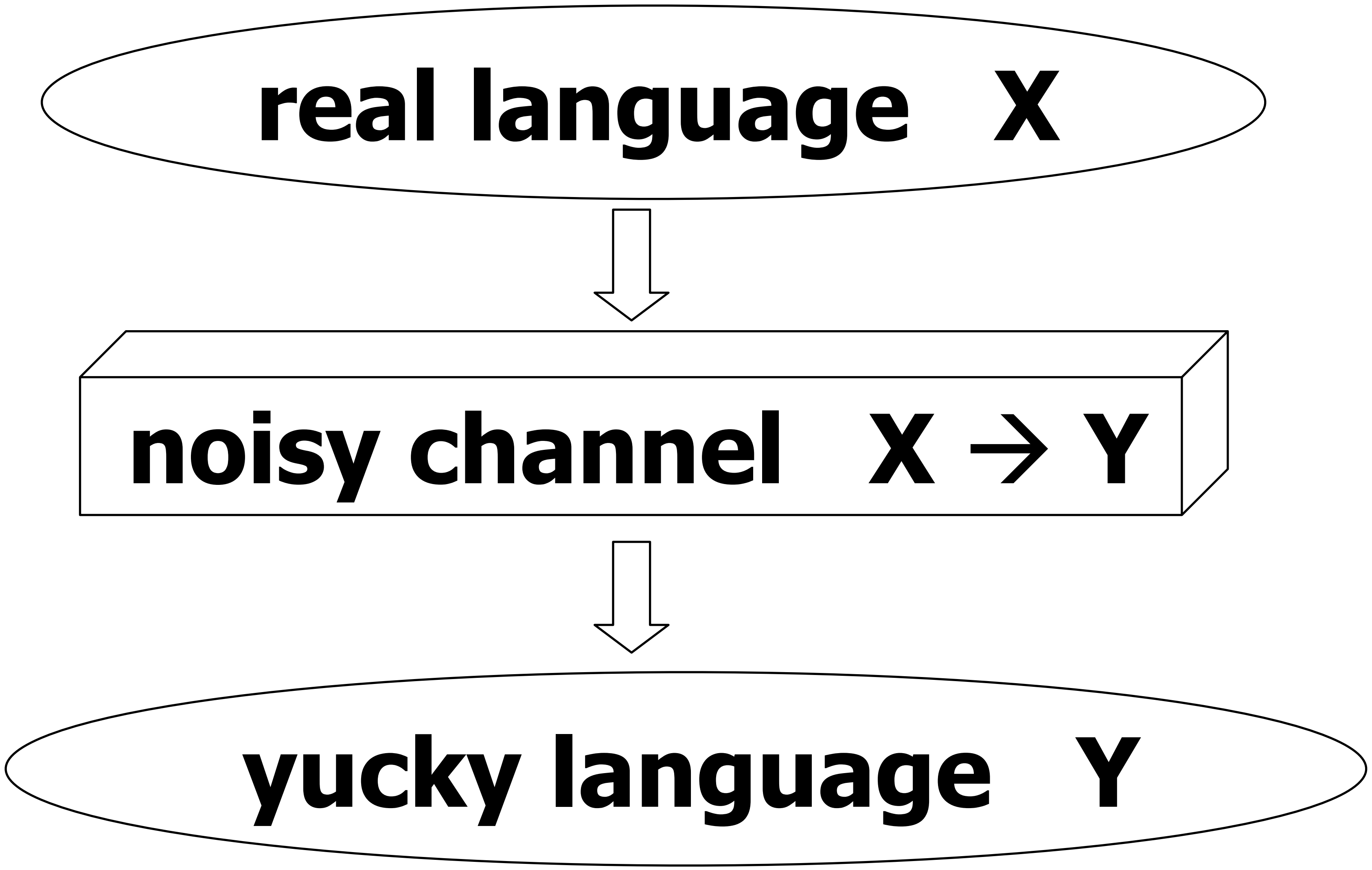
$$p(Y \mid X)$$

$$=$$

$$p(X,Y)$$

want to recover $x \in X$ from $y \in Y$

Review: Noisy Channel



$$p(X) * p(Y \mid X) = p(X,Y)$$

want to recover $x \in X$ from $y \in Y$

choose x that maximizes $p(x \mid y)$ or equivalently $p(x,y)$

Noisy Channel for Tagging

acceptor: p(tag sequence) $p(X)$

"Markov Model"

.o. $*$

transducer: tags → words $p(Y \mid X)$

"Unigram Replacement"

.o. $*$

acceptor: the observed words $(Y = y)?$

"straight line"

= $=$

transducer: scores candidate tag seqs on their joint probability with obs words; pick best path $p(X, y)$

Markov Model (bigrams)

Verb

Det

Start

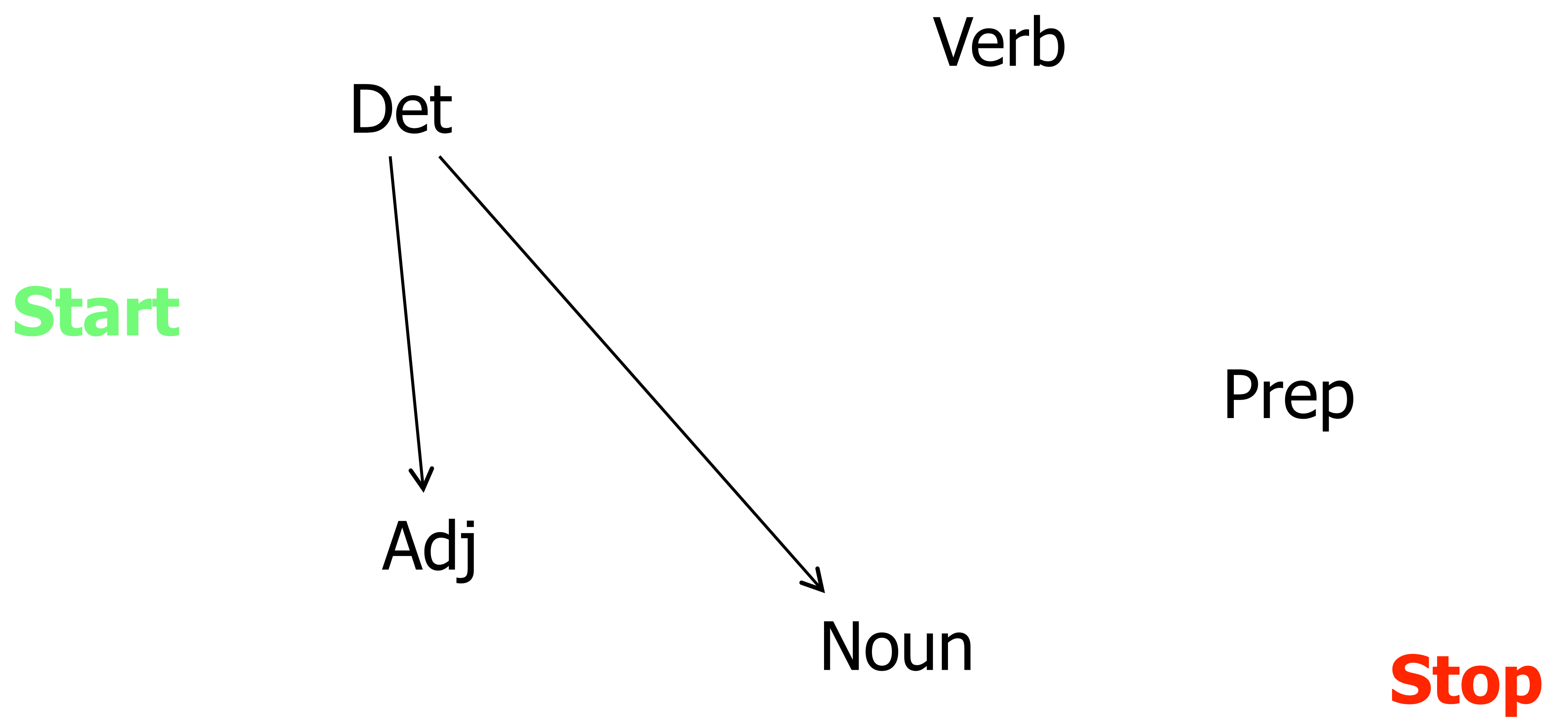
Prep

Adj

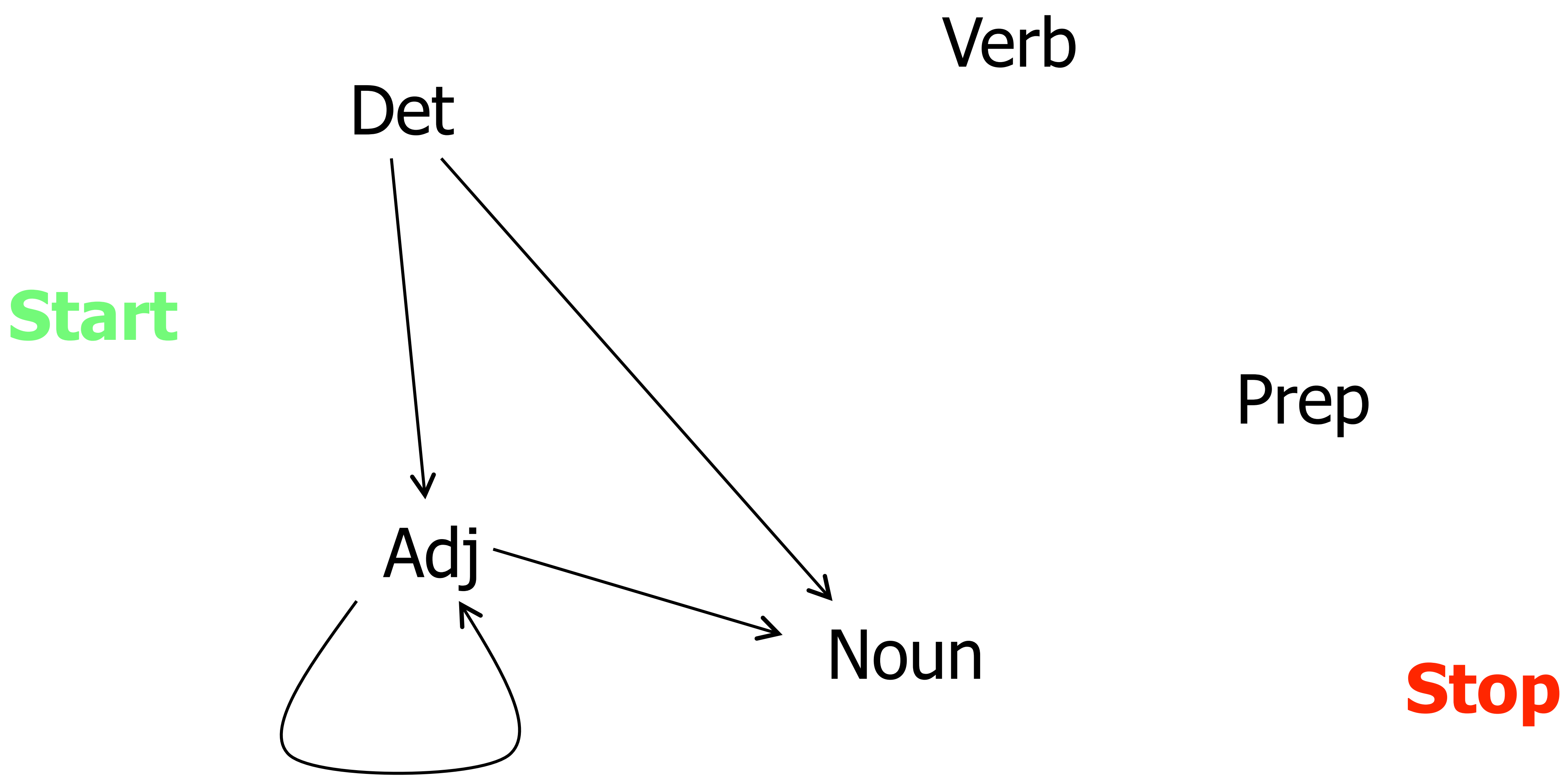
Noun

Stop

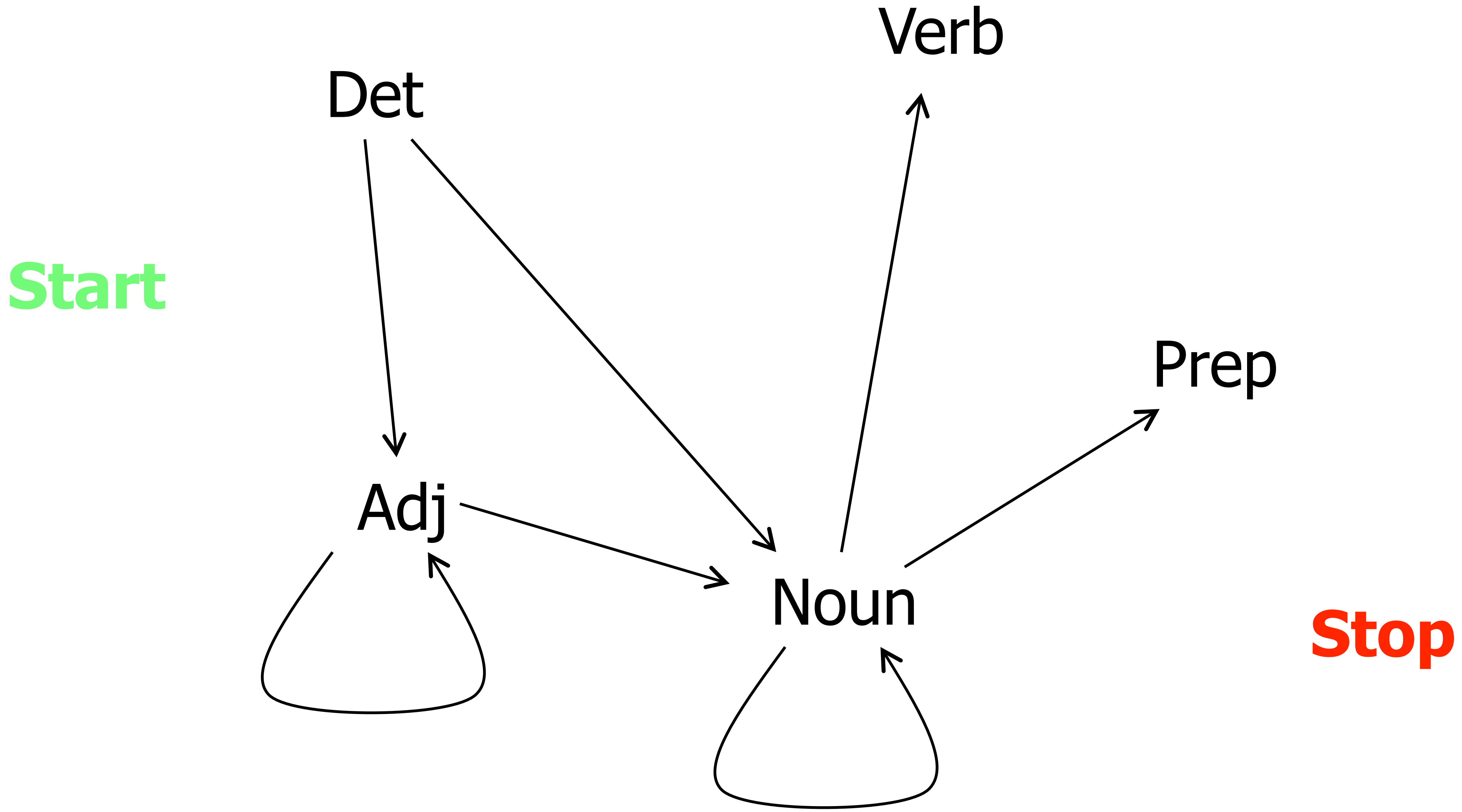
Markov Model (bigrams)



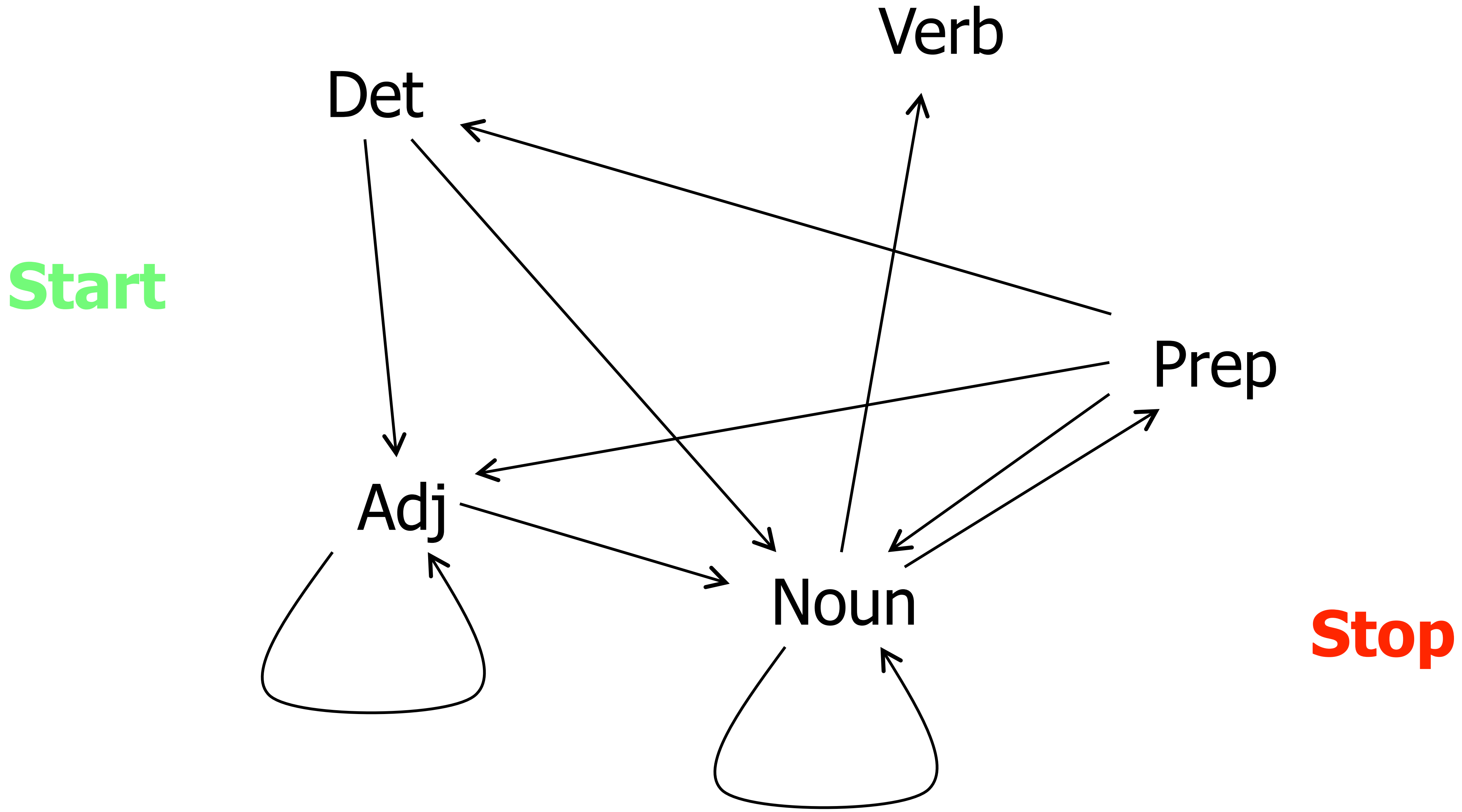
Markov Model (bigrams)



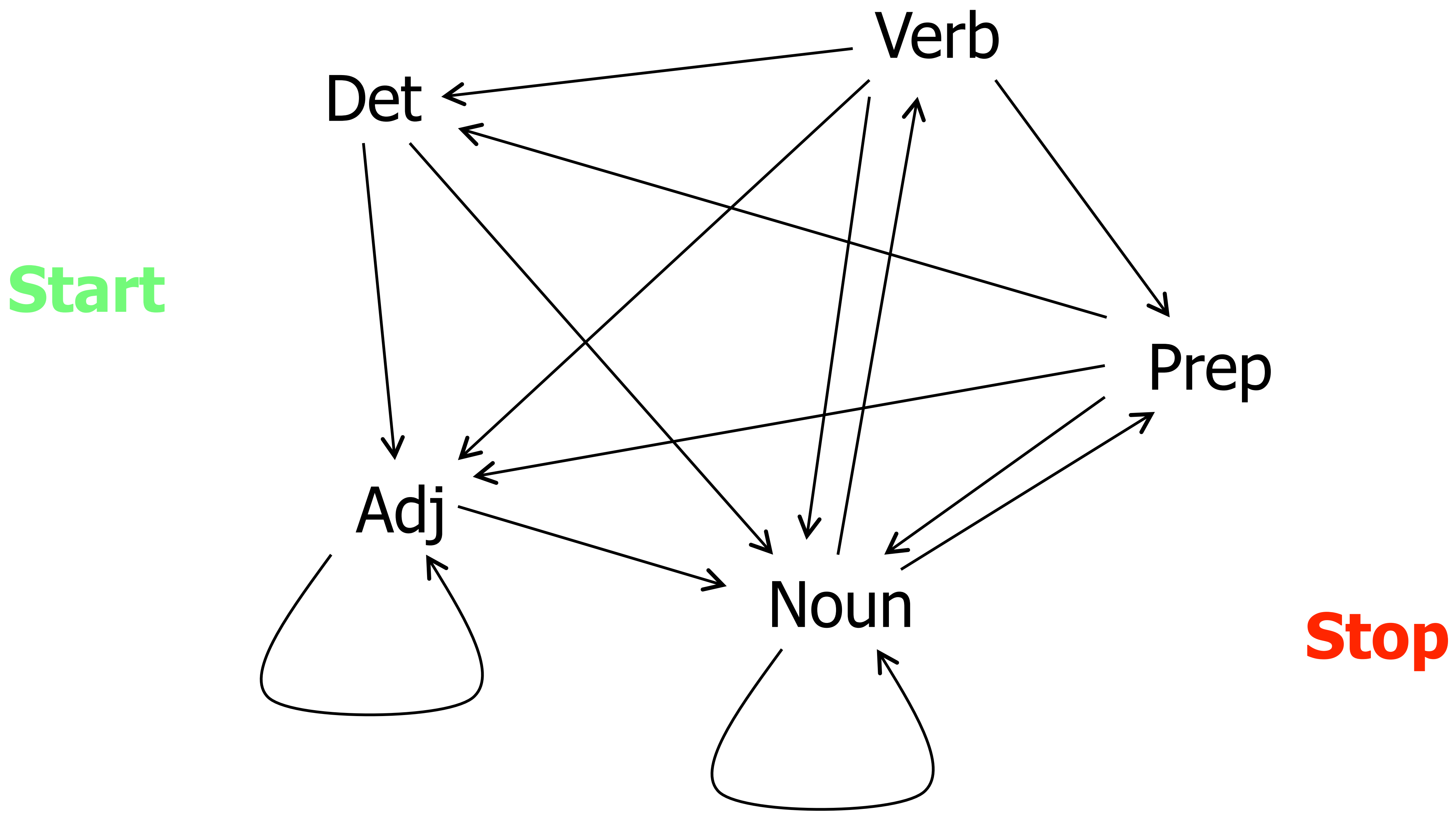
Markov Model (bigrams)



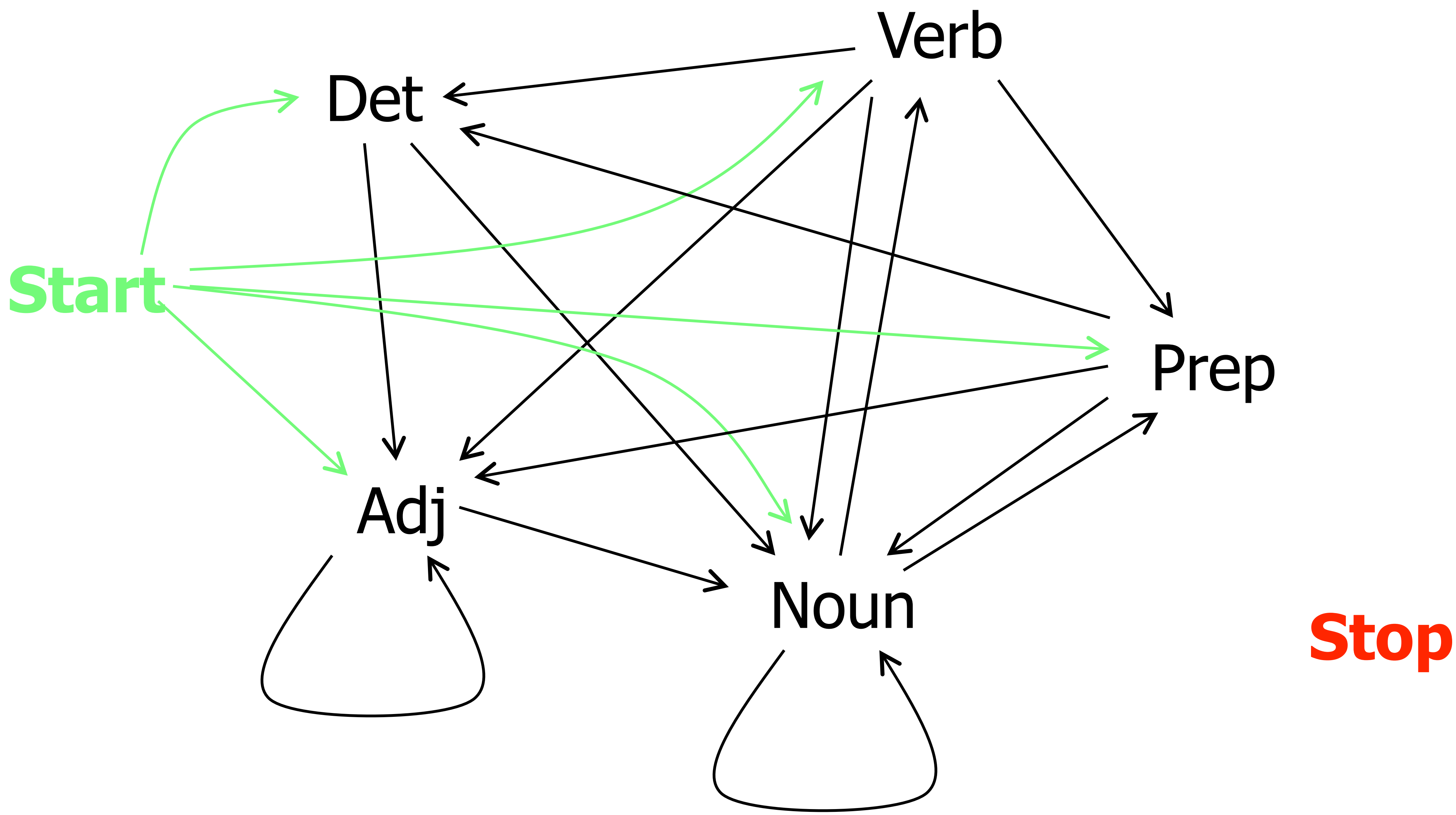
Markov Model (bigrams)



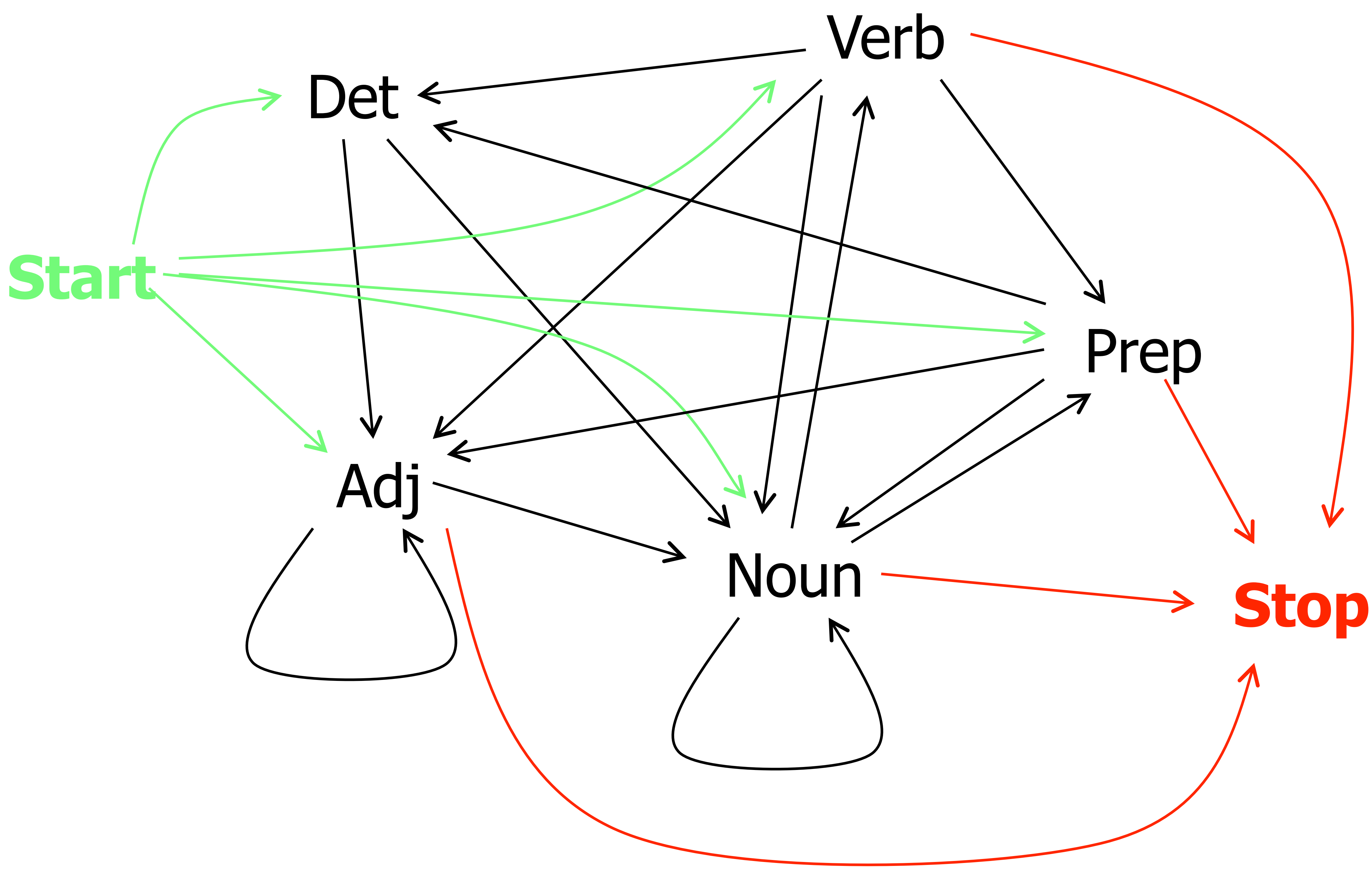
Markov Model (bigrams)



Markov Model (bigrams)



Markov Model (bigrams)



Markov Model

Verb

Det

Start

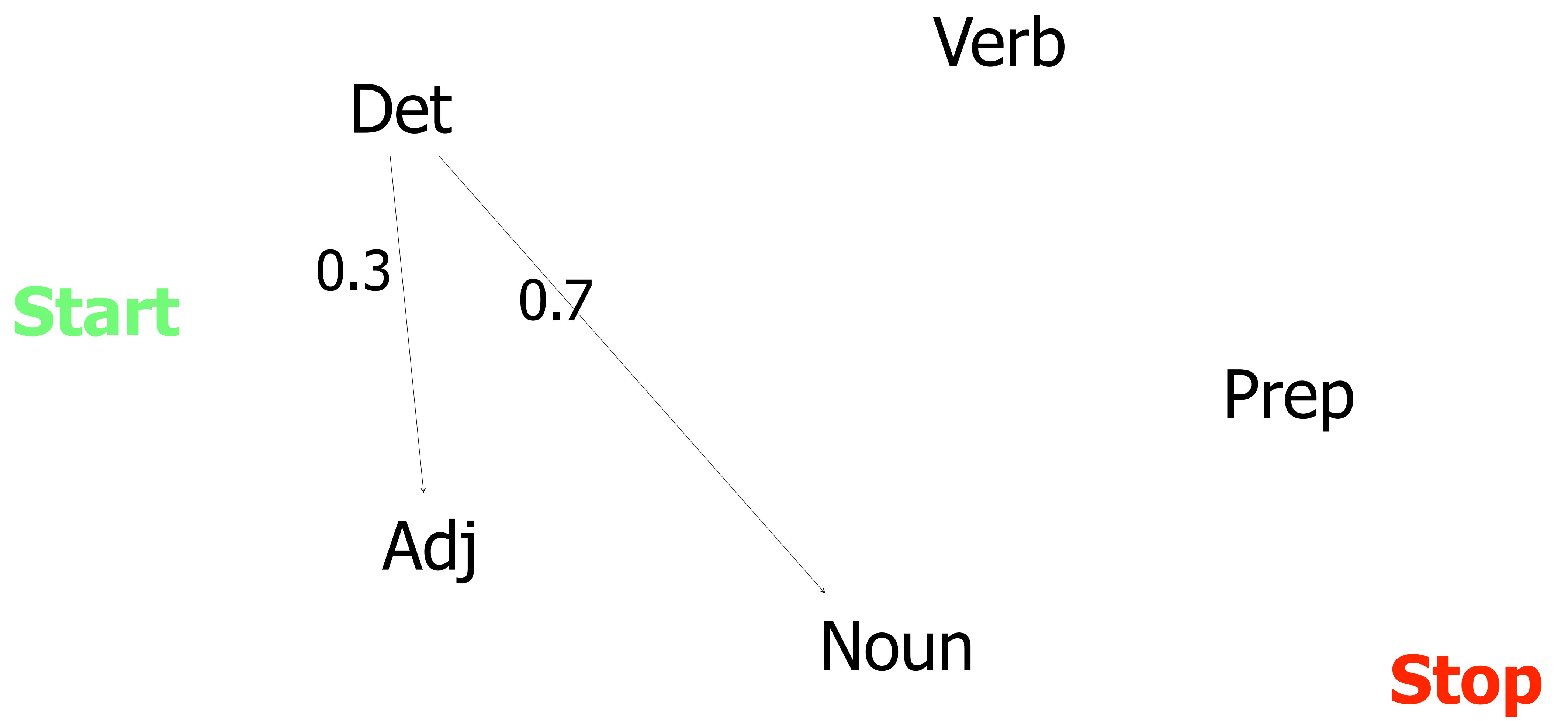
Prep

Adj

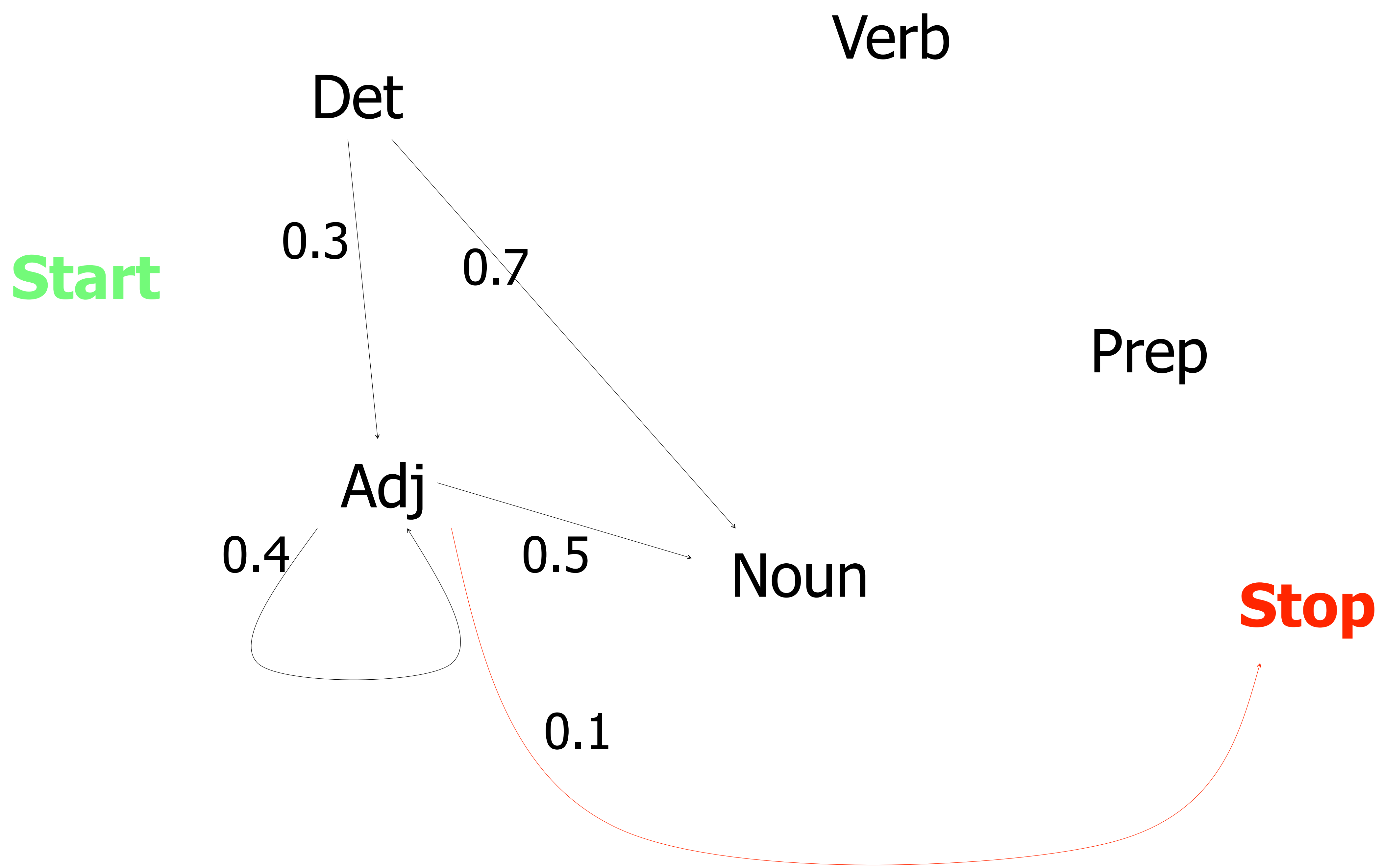
Noun

Stop

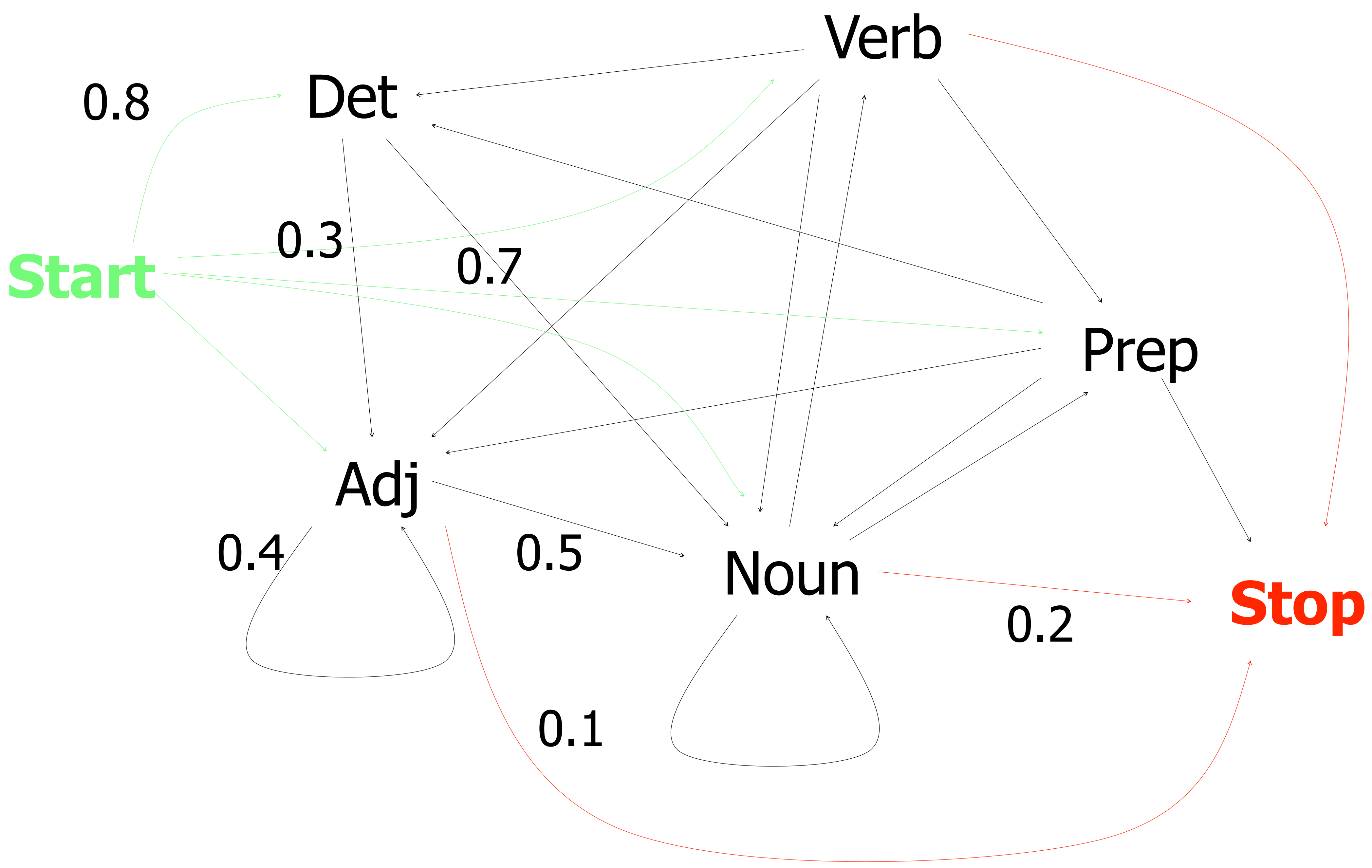
Markov Model



Markov Model

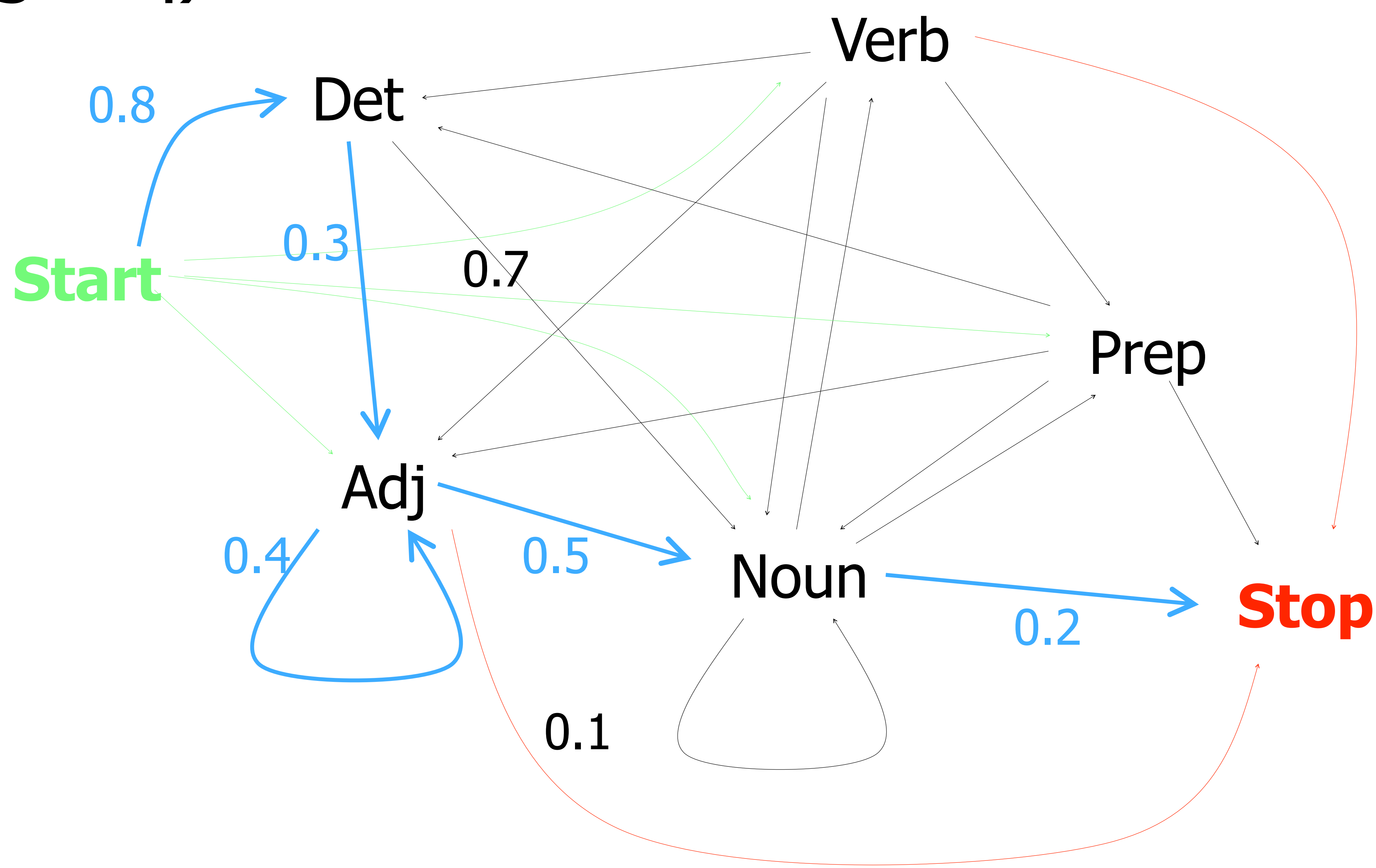


Markov Model



Markov Model

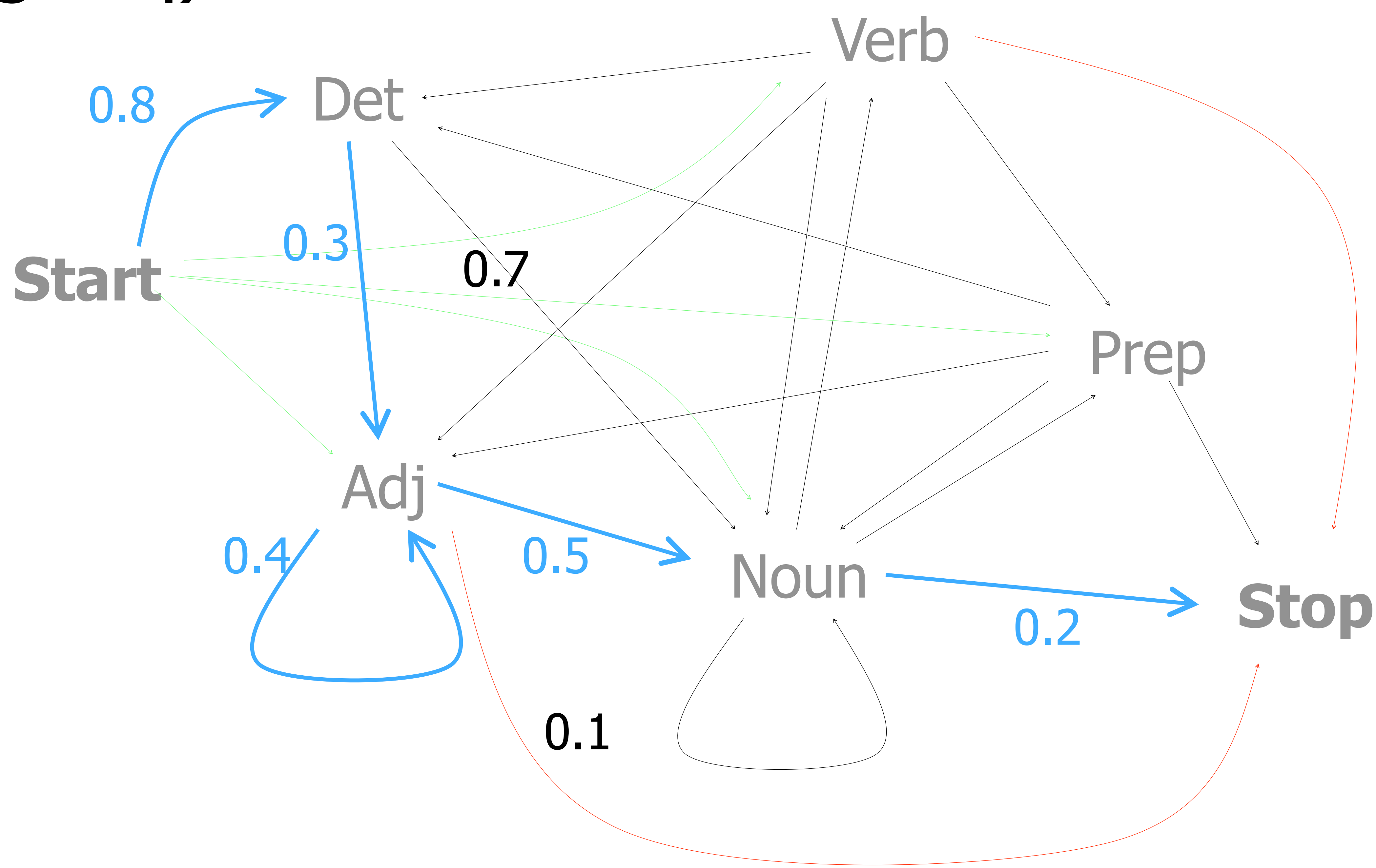
p(tag seq)



Start Det Adj Adj Noun Stop = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Markov Model as an FSA

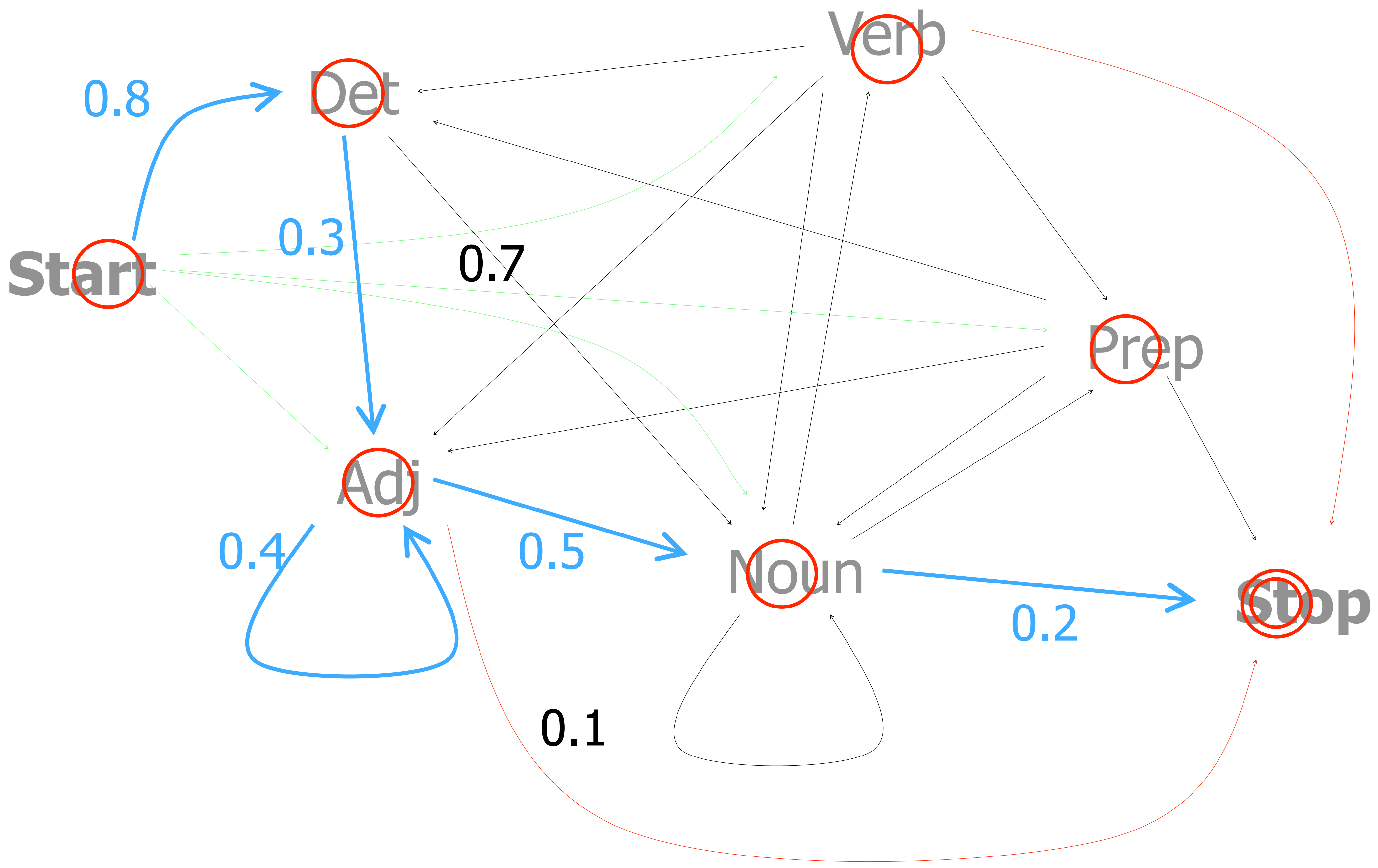
p(tag seq)



Start Det Adj Adj Noun Stop = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Markov Model as an FSA

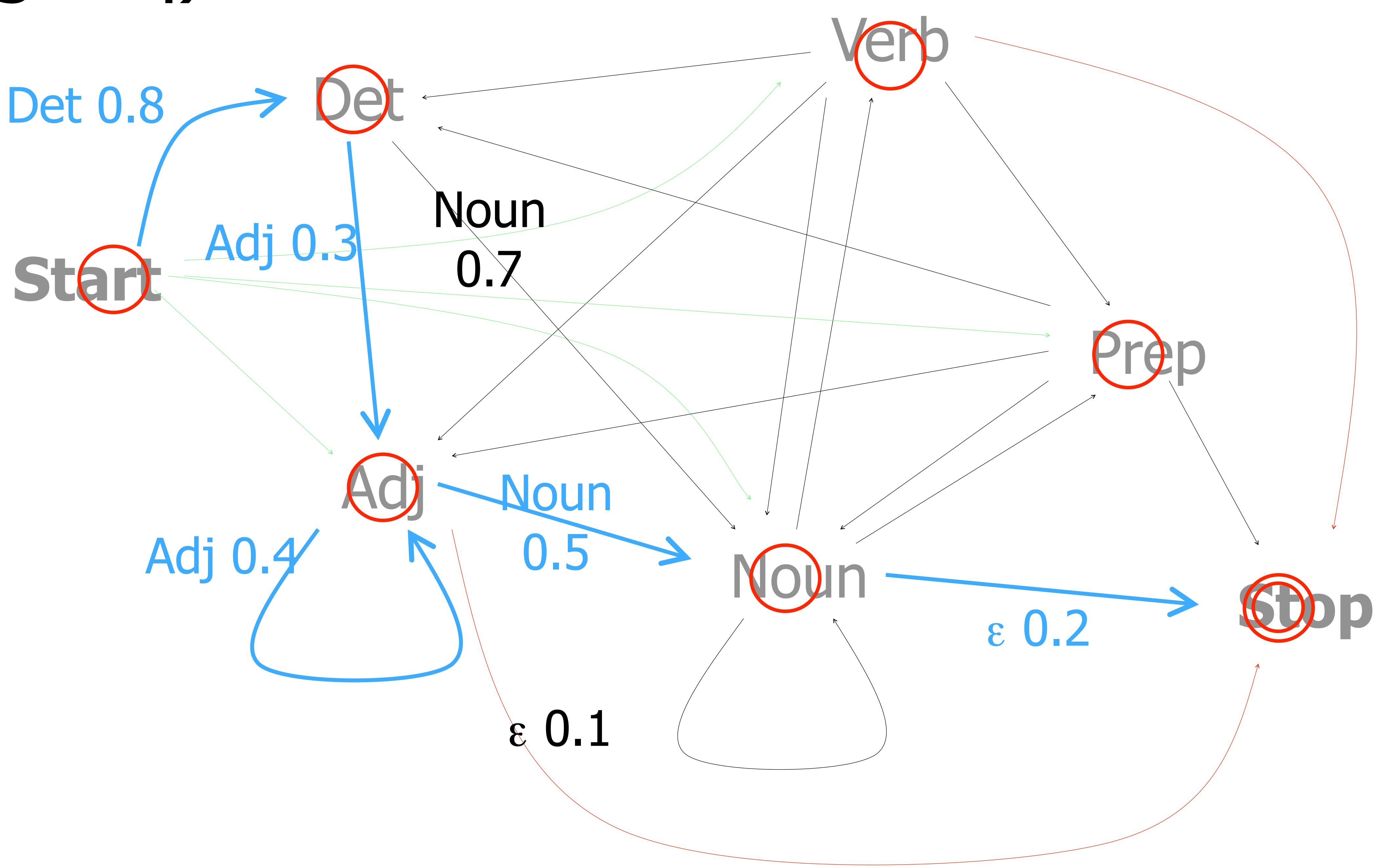
p(tag seq)



Start Det Adj Adj Noun **Stop** = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Markov Model as an FSA

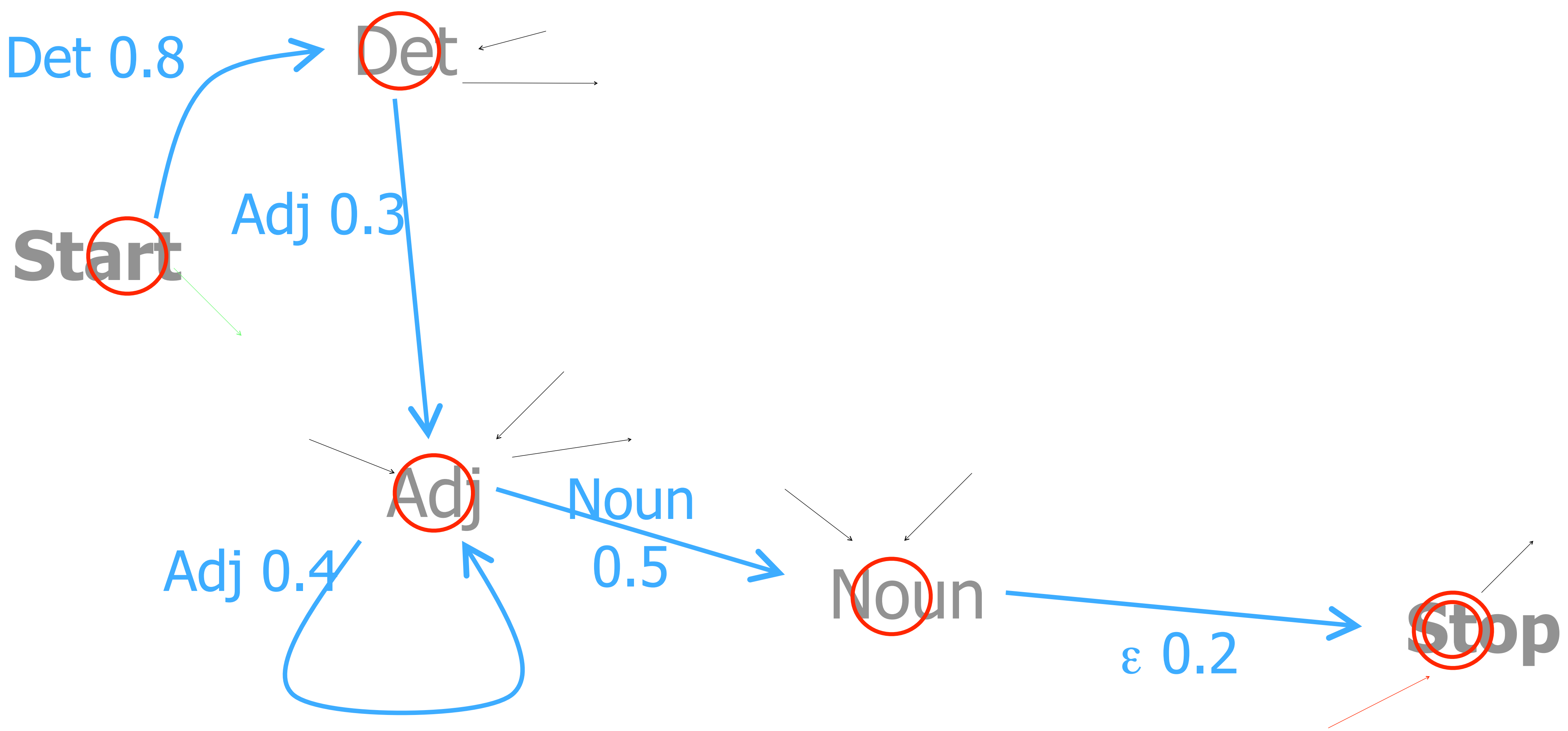
p(tag seq)



Start Det Adj Adj Noun **Stop** = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Markov Model (tag bigrams)

p(tag seq)



Start Det Adj Adj Noun **Stop** = 0.8 * 0.3 * 0.4 * 0.5 * 0.2

Noisy Channel for Tagging

automaton: p(tag sequence) $p(X)$

"Markov Model"

.o. $*$

transducer: tags → words $p(Y \mid X)$

"Unigram Replacement"

.o. $*$

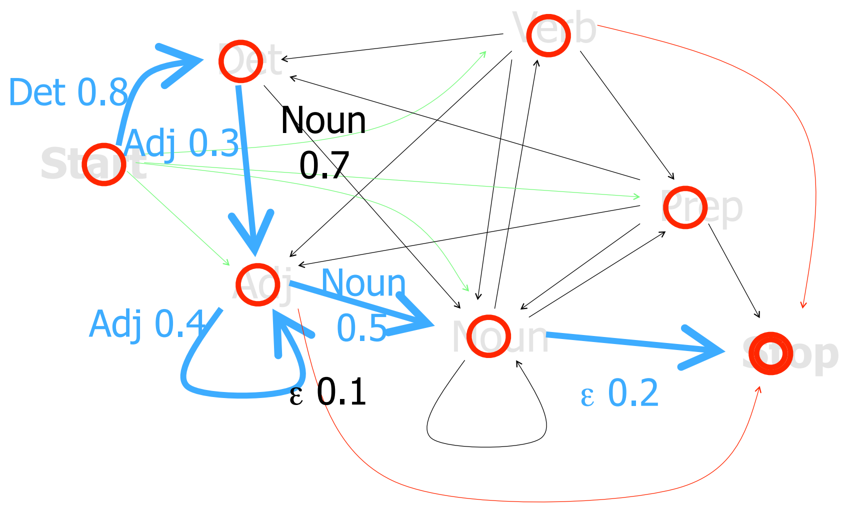
automaton: the observed words $p(y \mid Y)$

"straight line"

= =

transducer: scores candidate tag seqs on their joint probability with obs words; pick best path $p(X, y)$

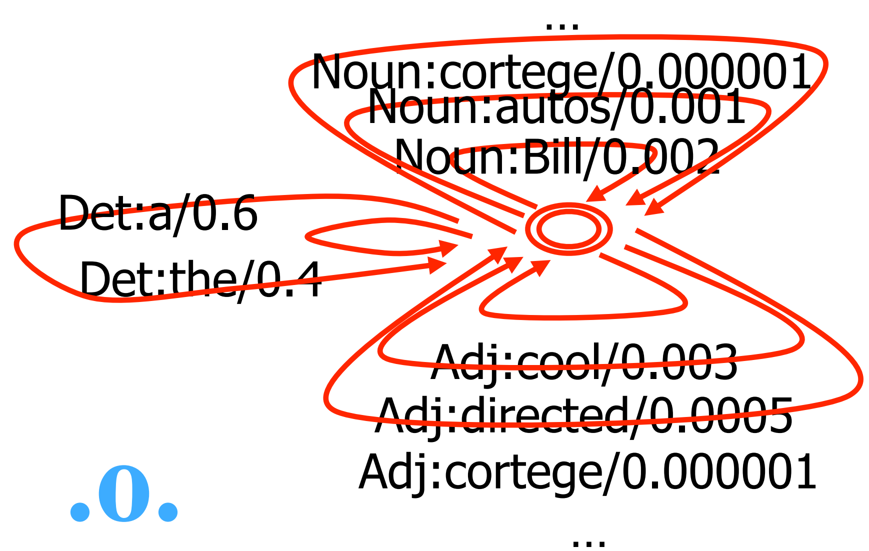
Noisy Channel for Tagging



p(X)

.o.

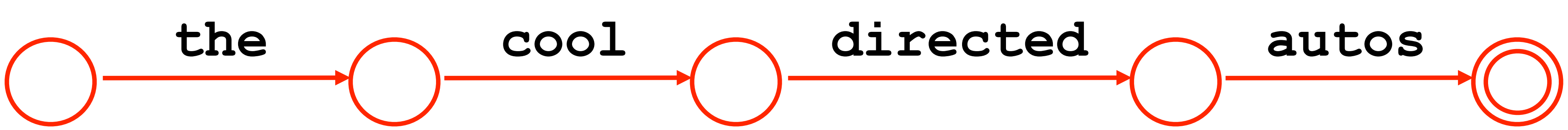
*



p(Y | X)

.o.

*



p(y | Y)

=

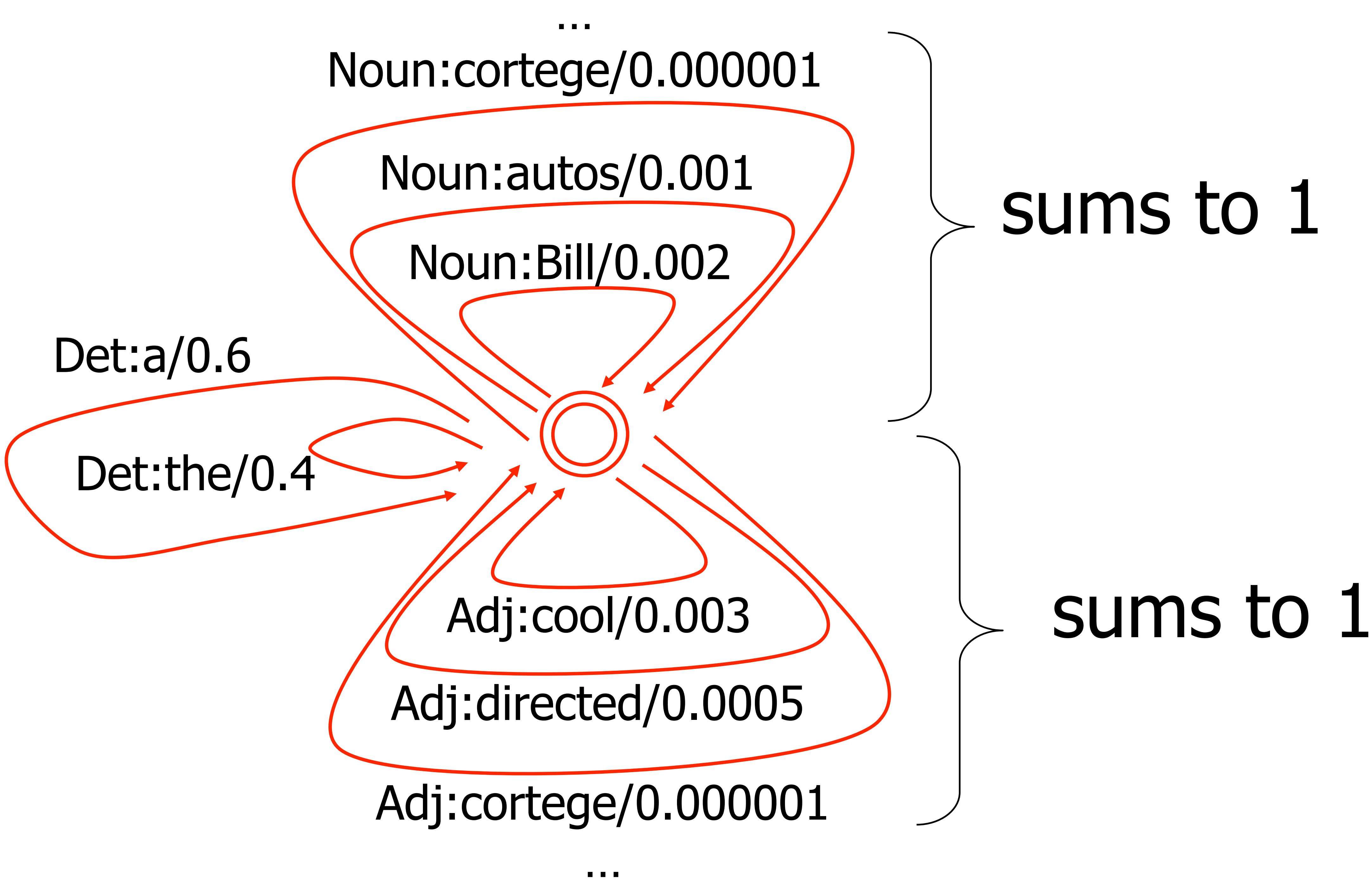
=

transducer: scores candidate tag seqs on their joint probability with obs words; we should pick best path

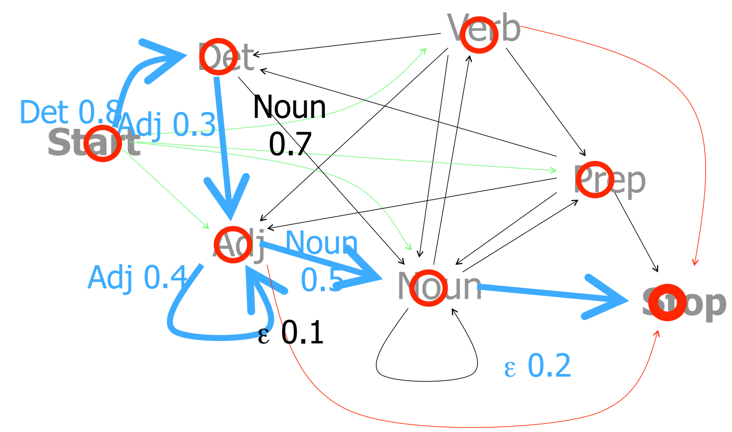
p(X, y)

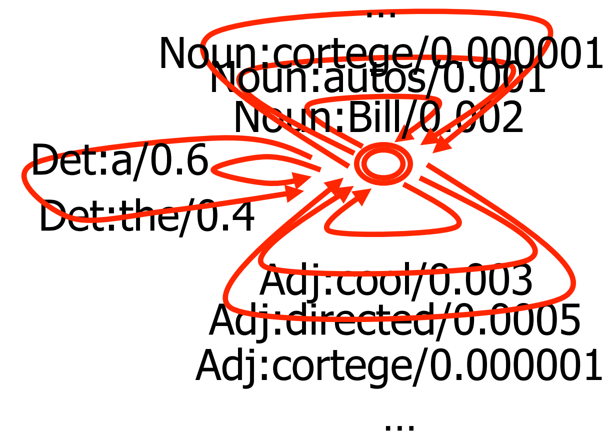
Unigram Replacement Model

p(word seq | tag seq)

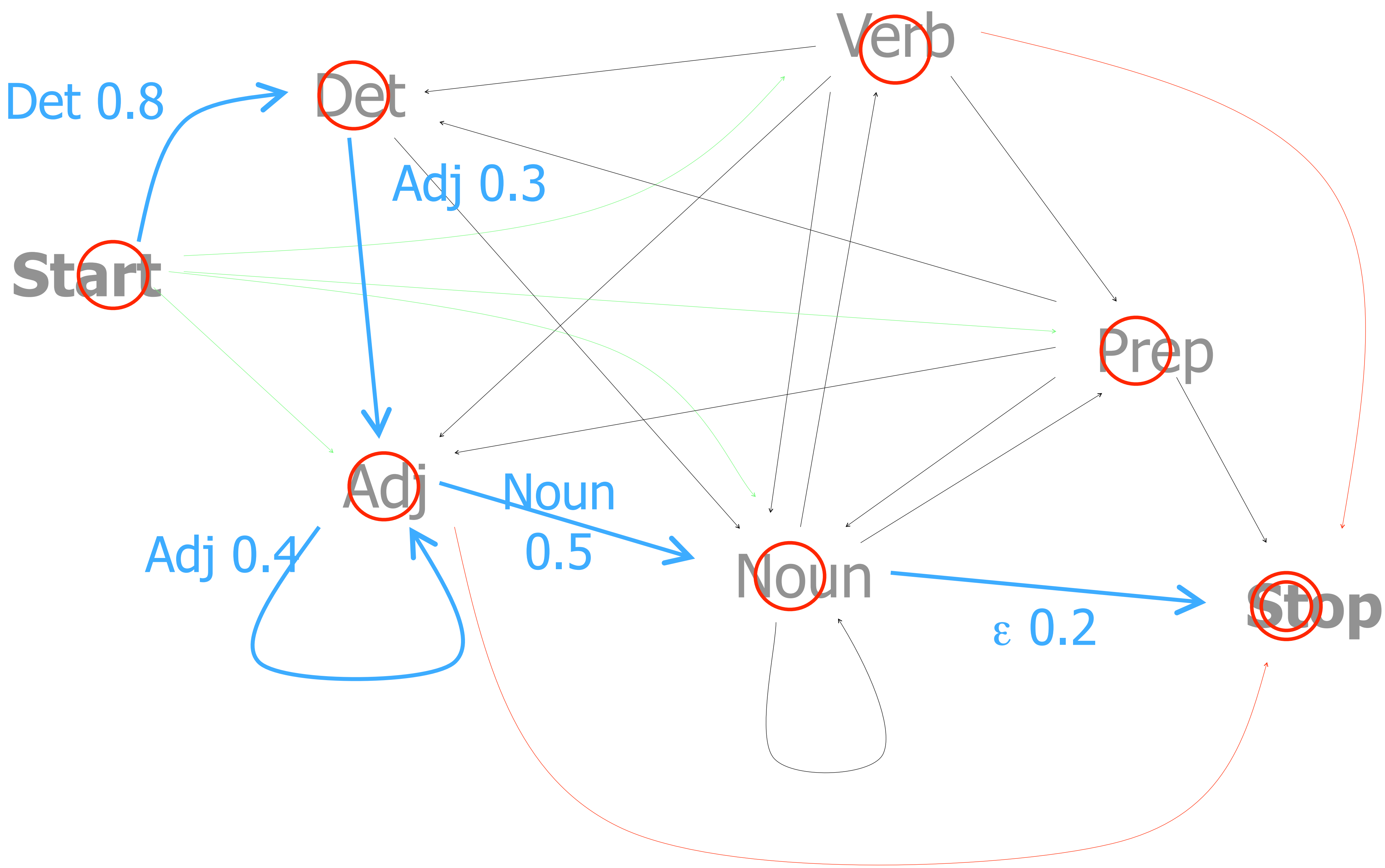


Compose

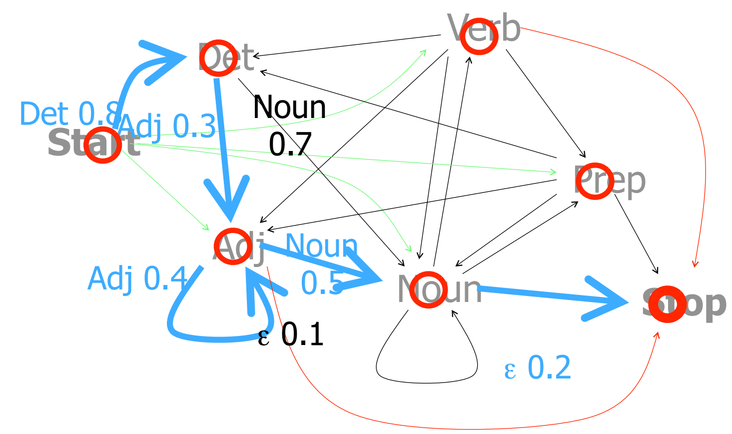


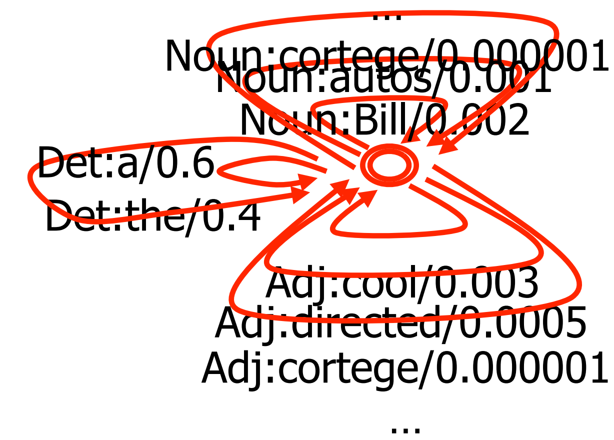


p(tag seq)

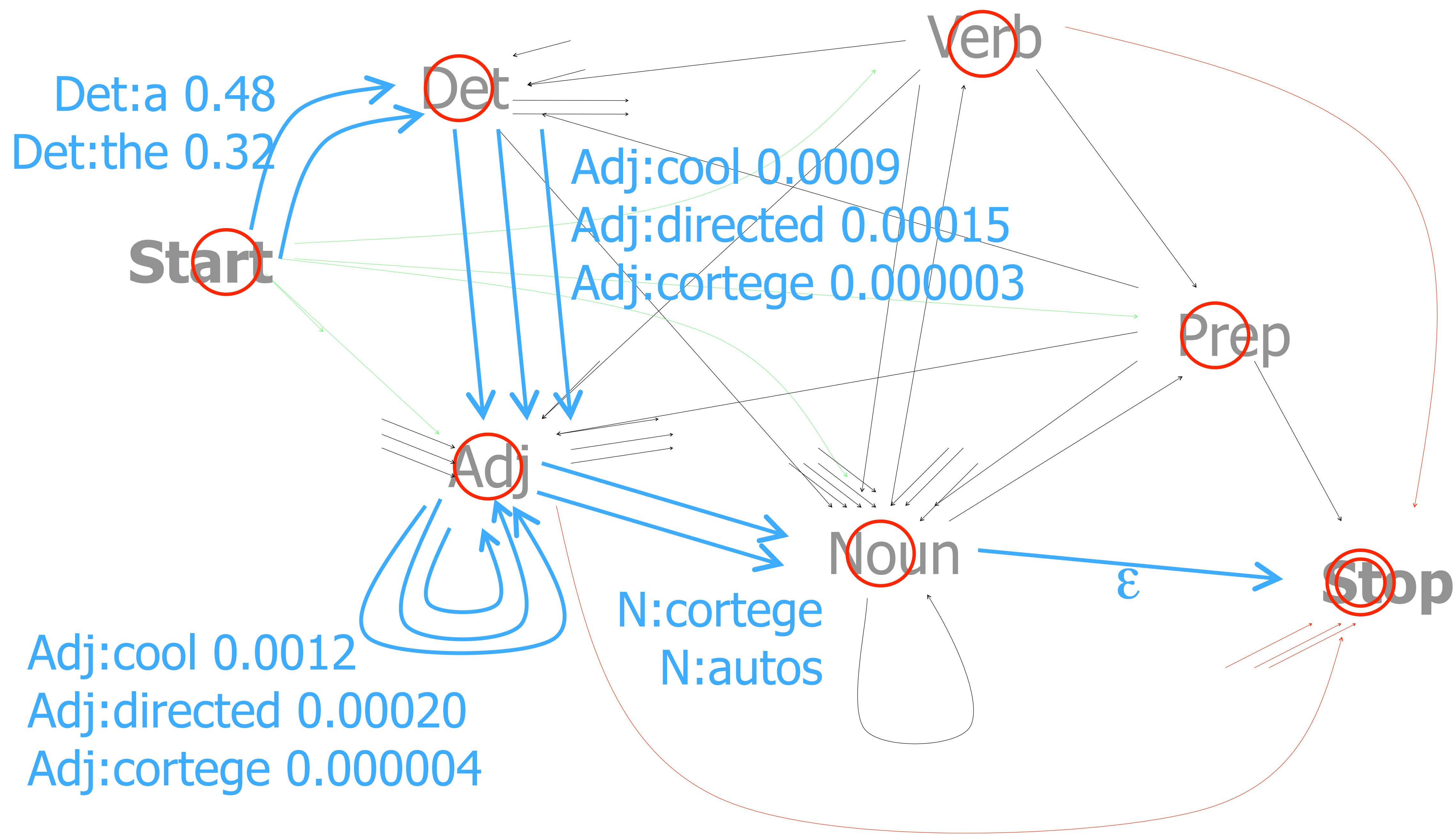


Compose



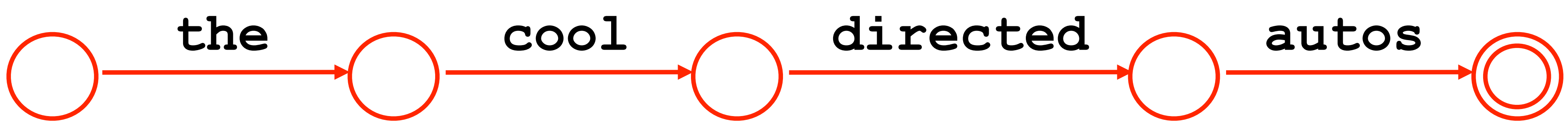


p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)

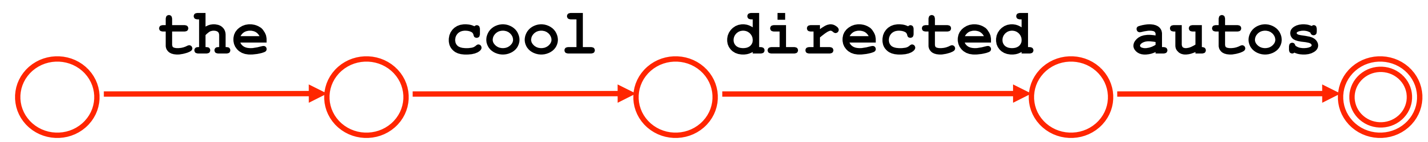


Observed Words as Straight-Line FSA

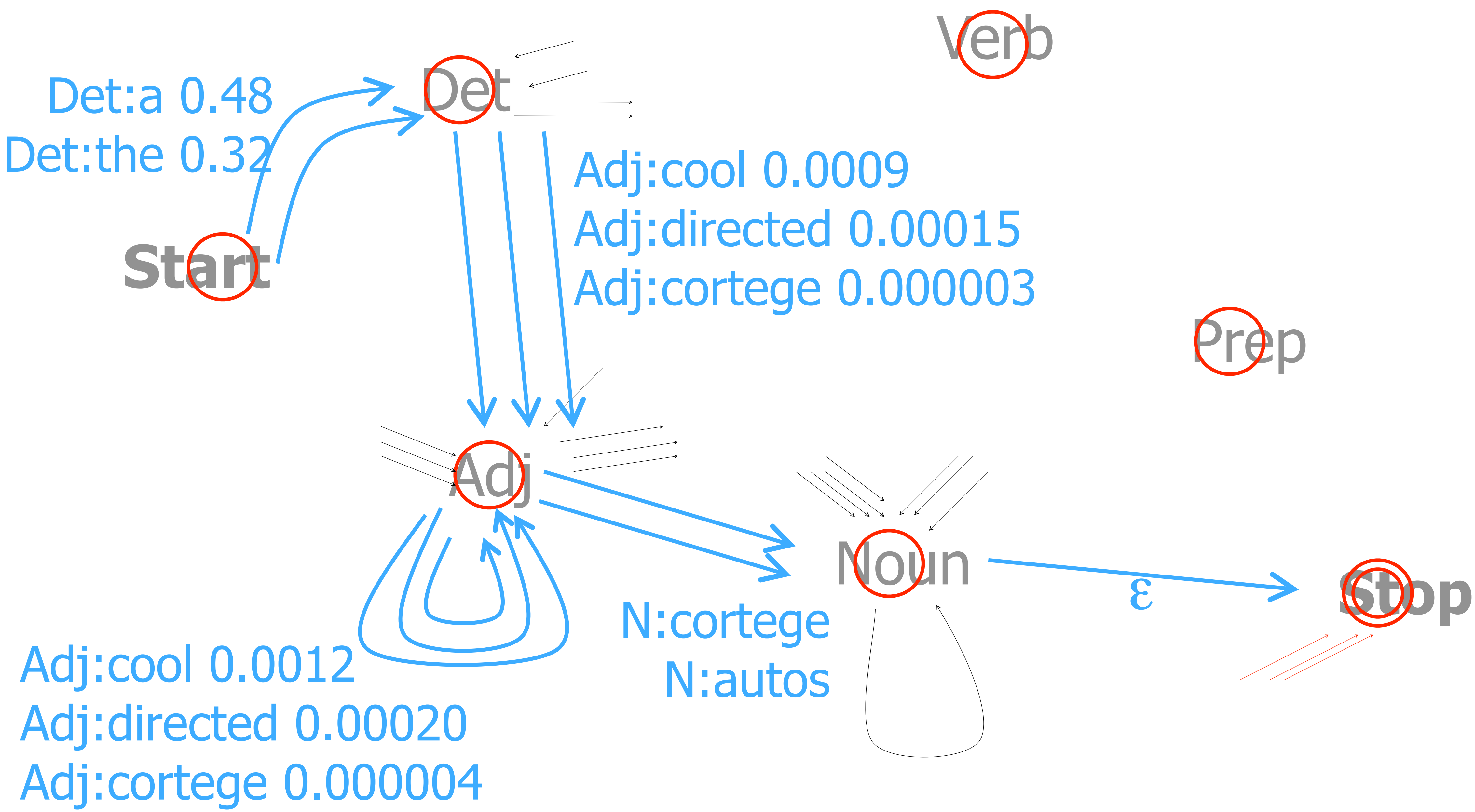
word seq



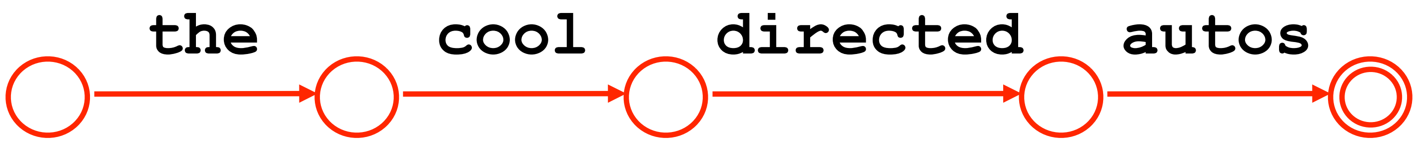
Compose with



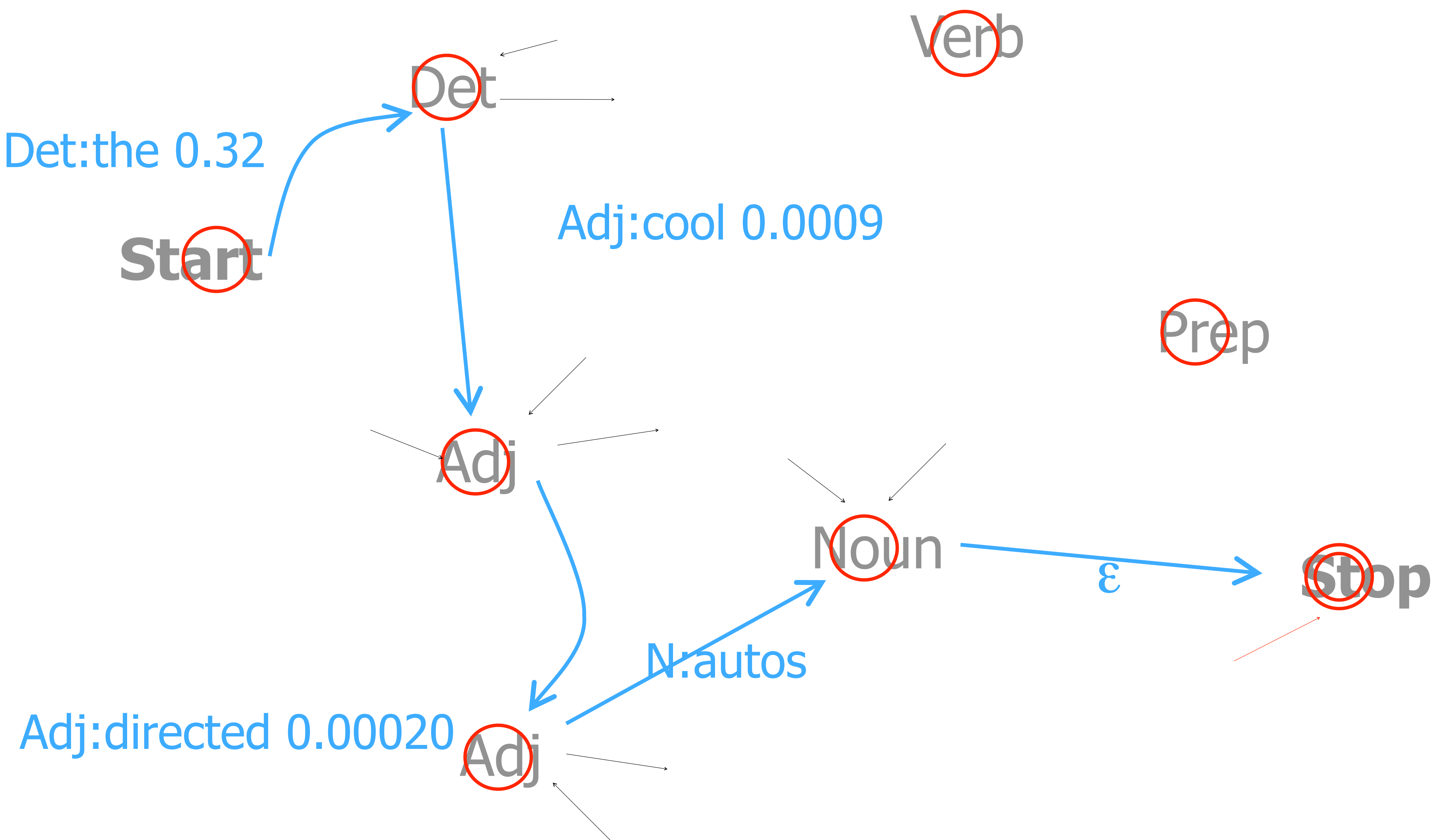
p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)



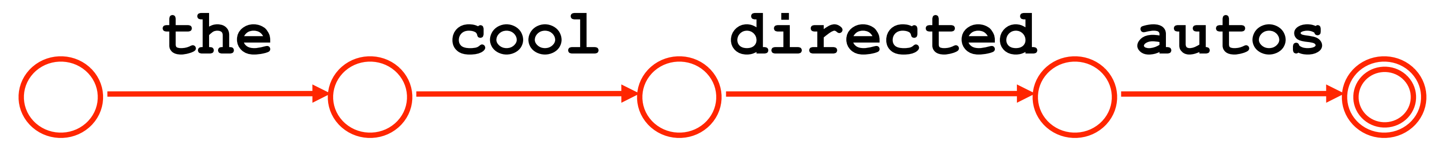
Compose with



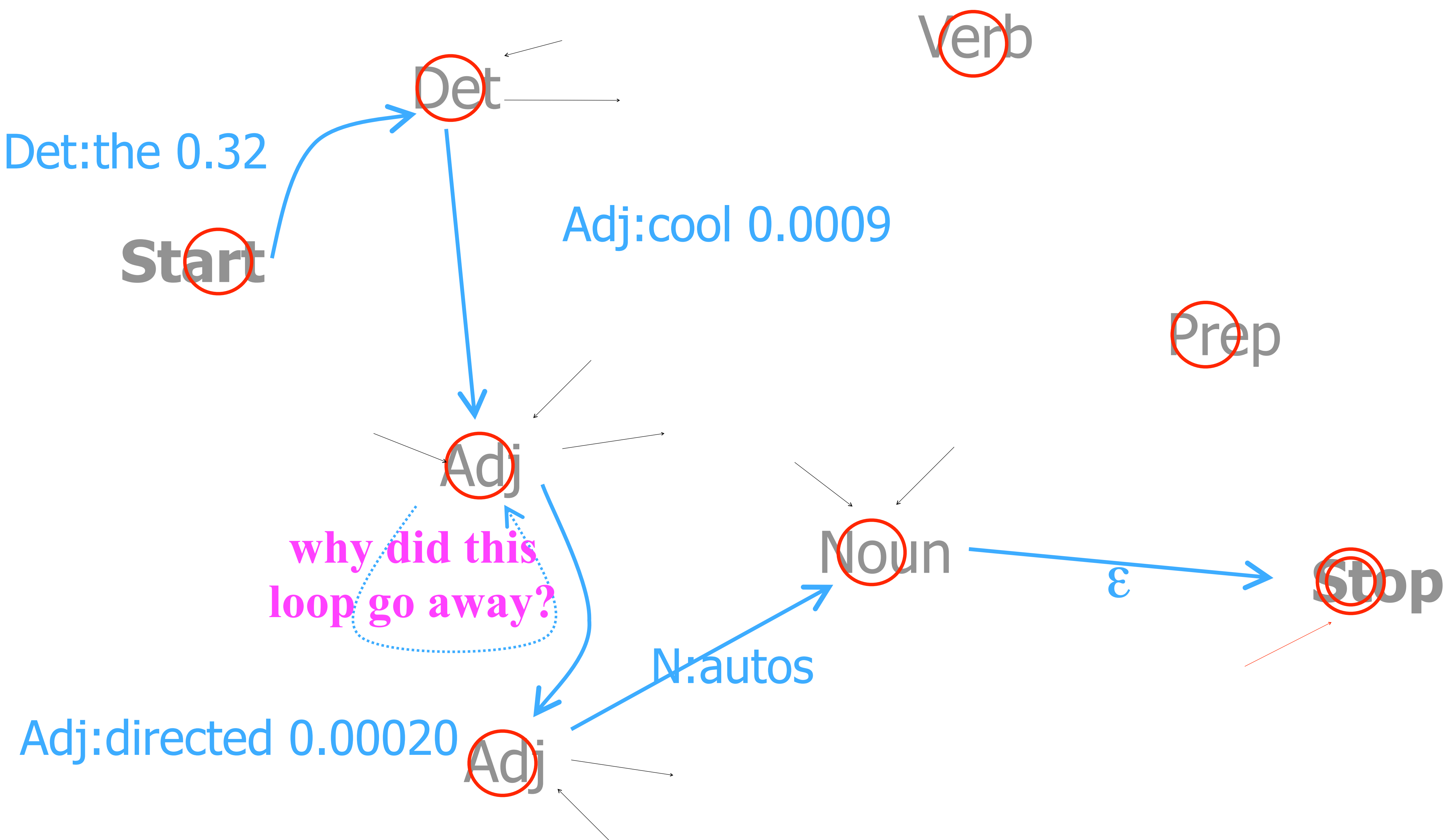
p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)



Compose with



p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)

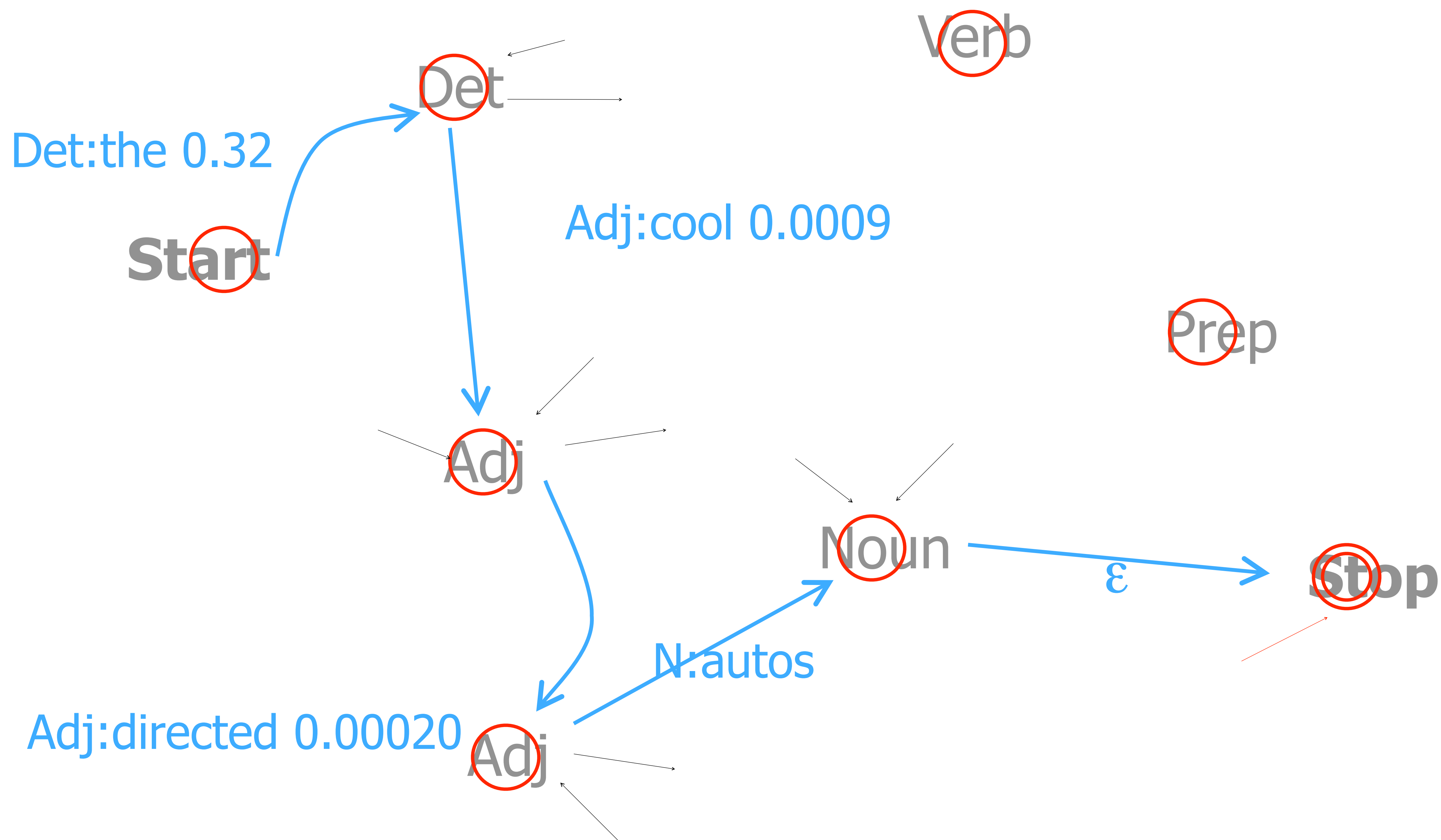


The best path:

Start Det Adj Adj Noun Stop = 0.32 * 0.0009 …

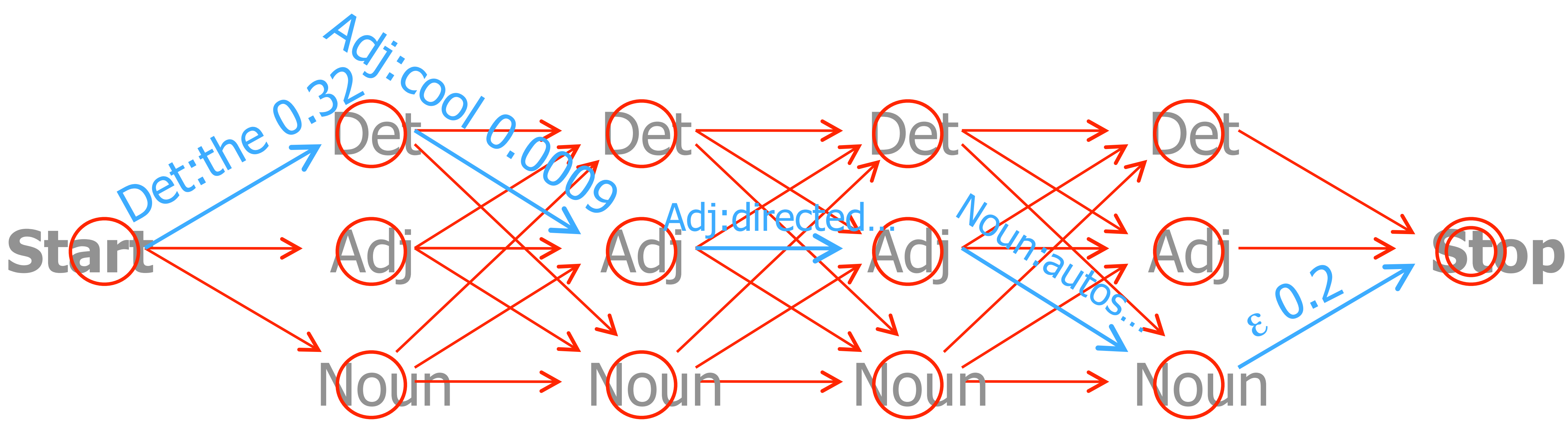
the cool directed autos

p(word seq, tag seq) = p(tag seq) * p(word seq | tag seq)



In Fact, Paths Form a “Trellis”

p(word seq, tag seq)

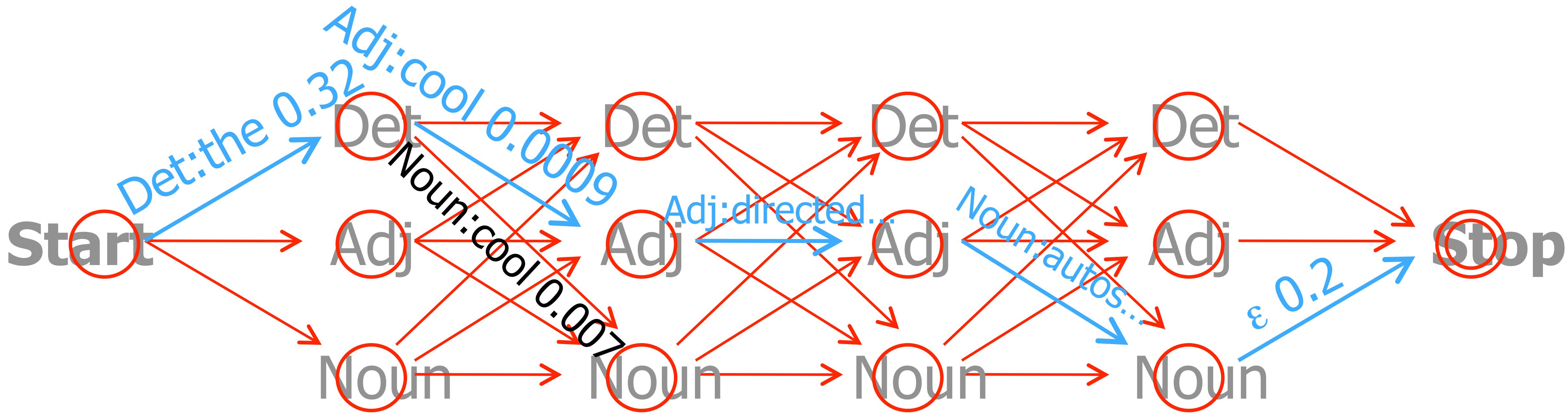


The best path:

Start	Det	Adj	Adj	Noun	**Stop**	= 0.32 * 0.0009 …
	the	cool	directed	autos		

In Fact, Paths Form a “Trellis”

p(word seq, tag seq)



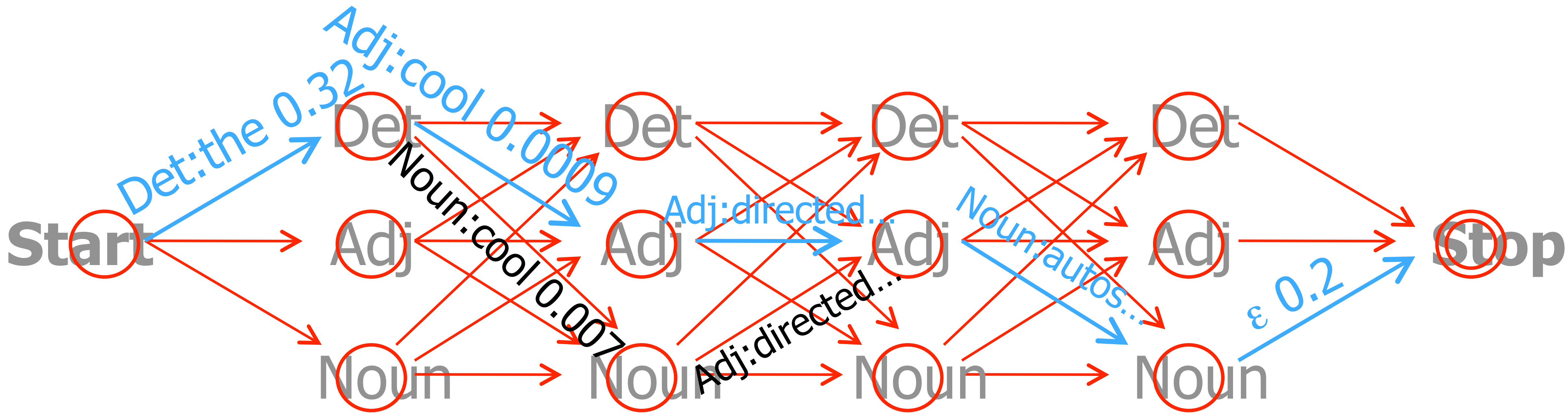
The best path:

Start Det Adj Adj Noun **Stop** = 0.32 * 0.0009 …

the cool directed autos

In Fact, Paths Form a "Trellis"

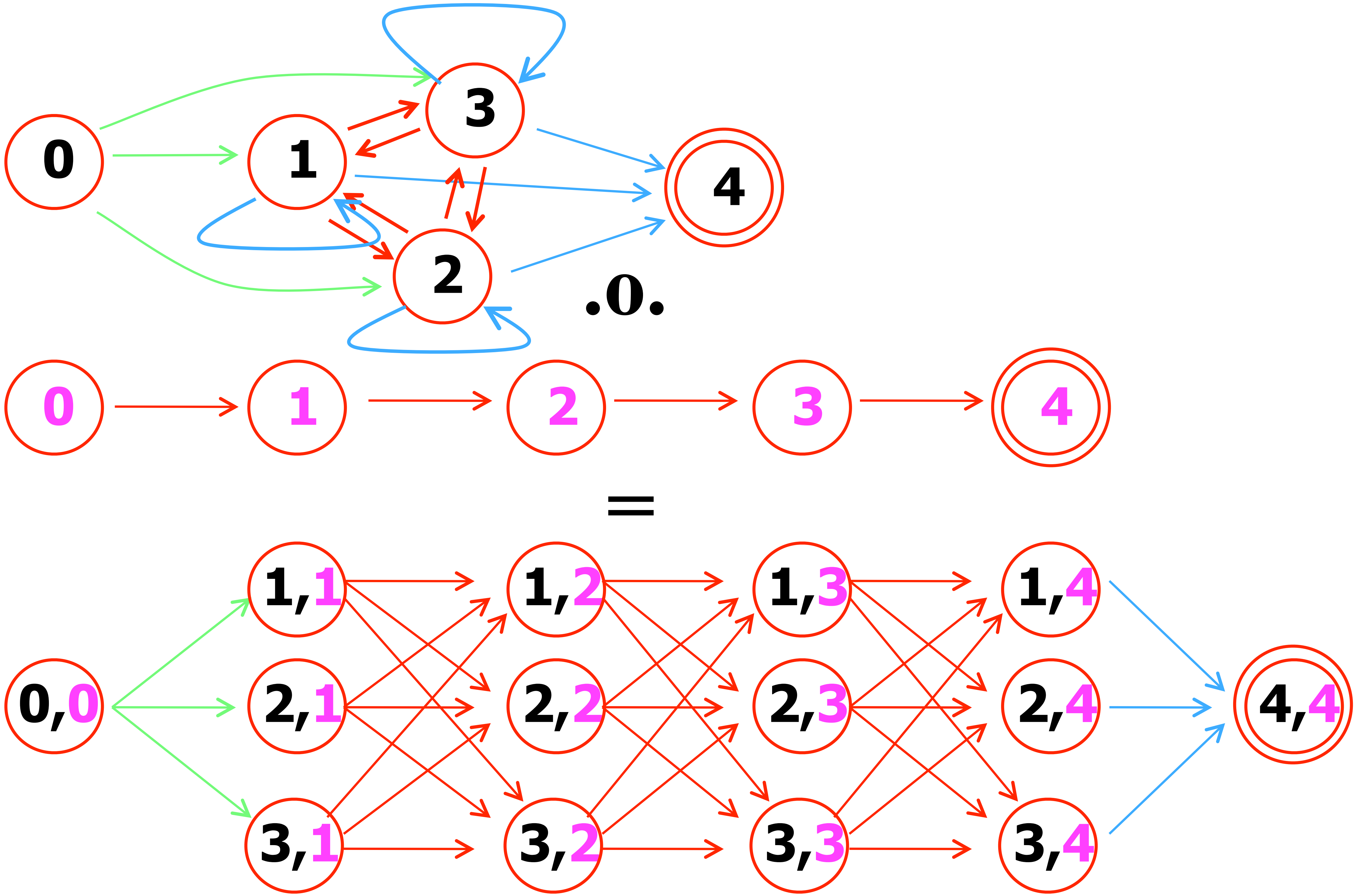
p(word seq, tag seq)



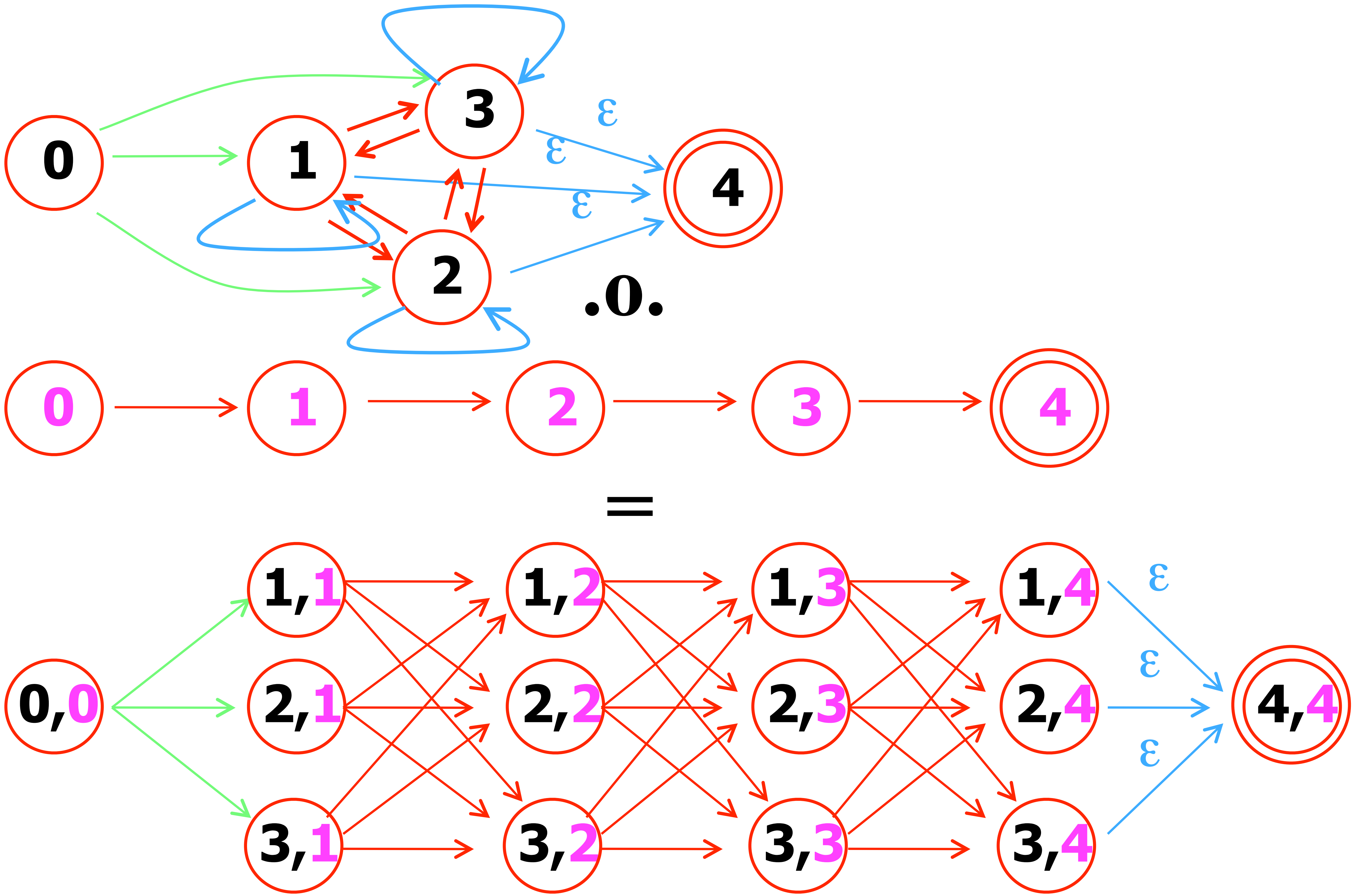
The best path:

Start	Det	Adj	Adj	Noun	**Stop**	= 0.32 * 0.0009 …
	the	cool	directed	autos		

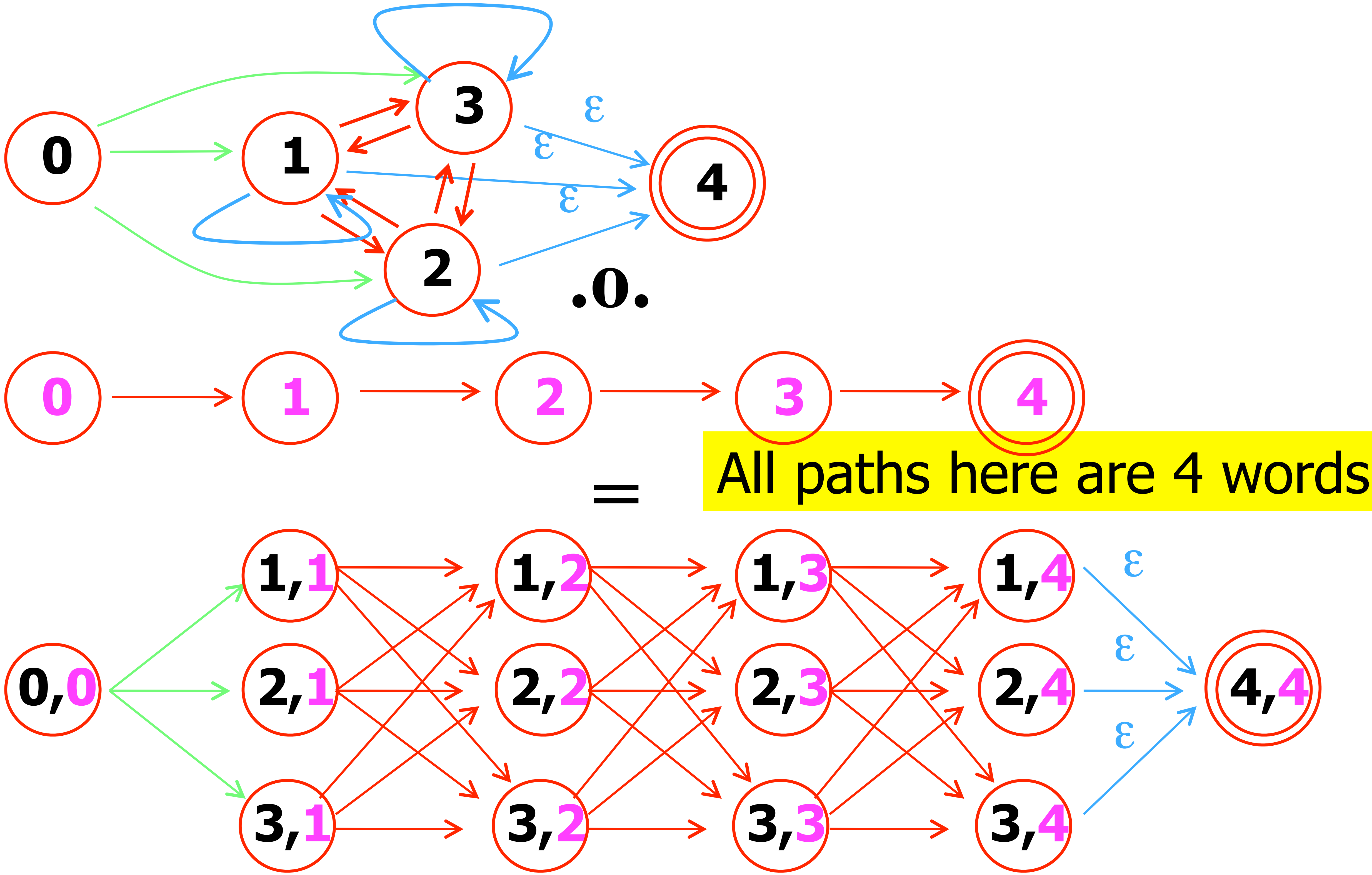
The Trellis Shape Emerges from the Cross-Product Construction for



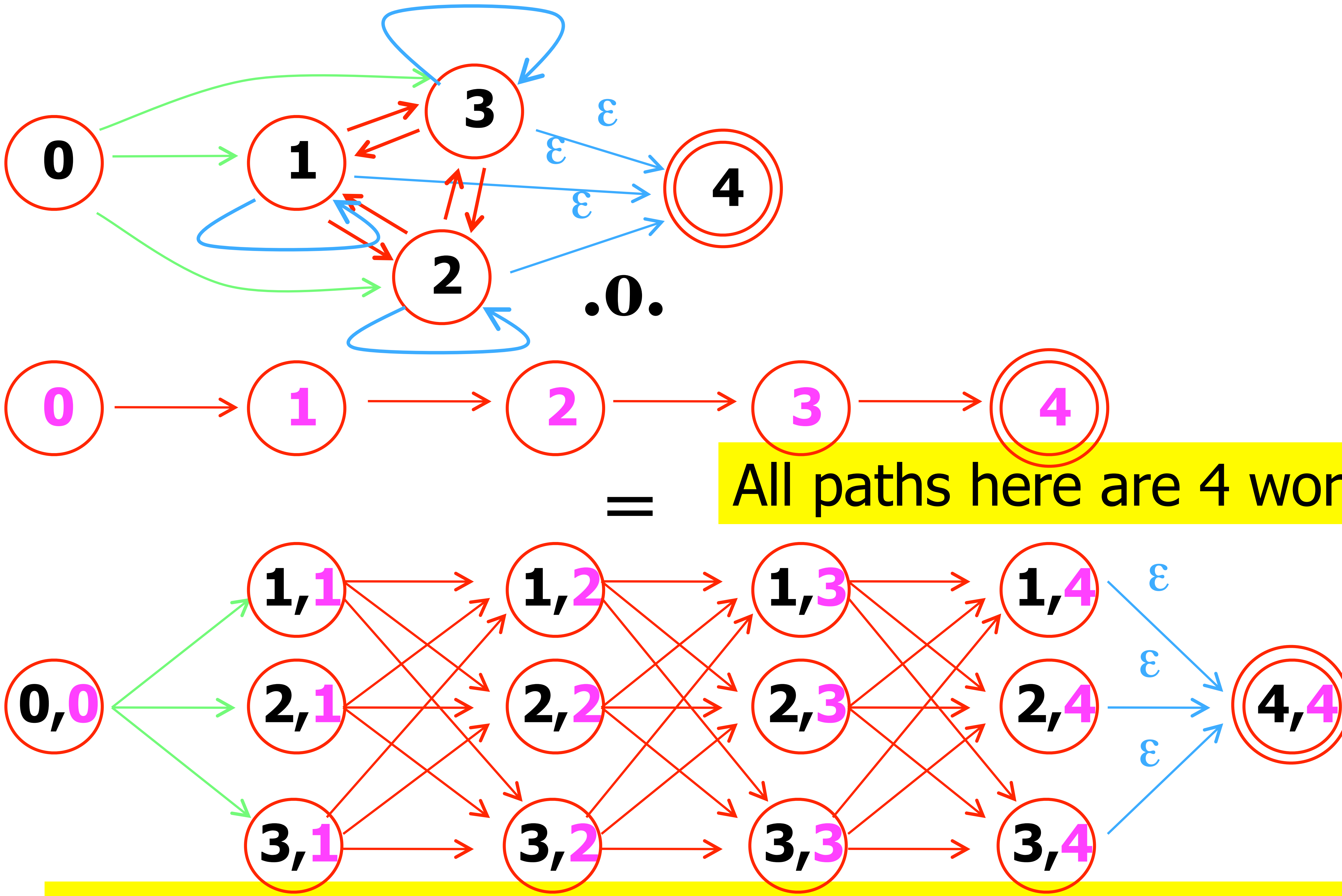
The Trellis Shape Emerges from the Cross-Product Construction for



The Trellis Shape Emerges from the Cross-Product Construction for



The Trellis Shape Emerges from the Cross-Product Construction for

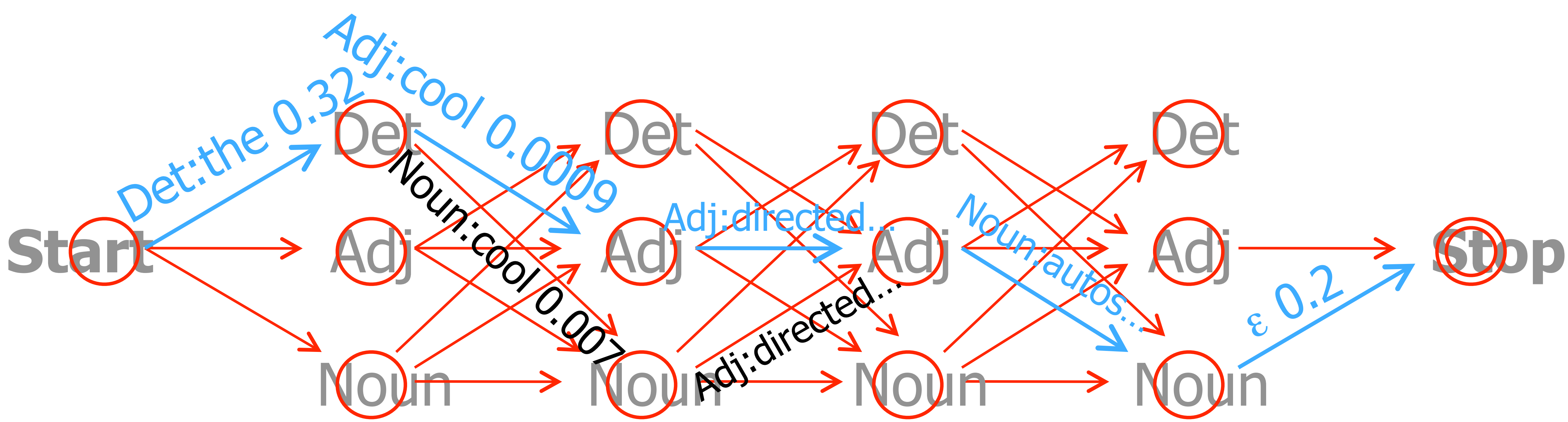


All paths here are 4 words

So all paths here must have 4 words on output side

Actually, Trellis Isn't Complete

p(word seq, tag seq)



The best path:

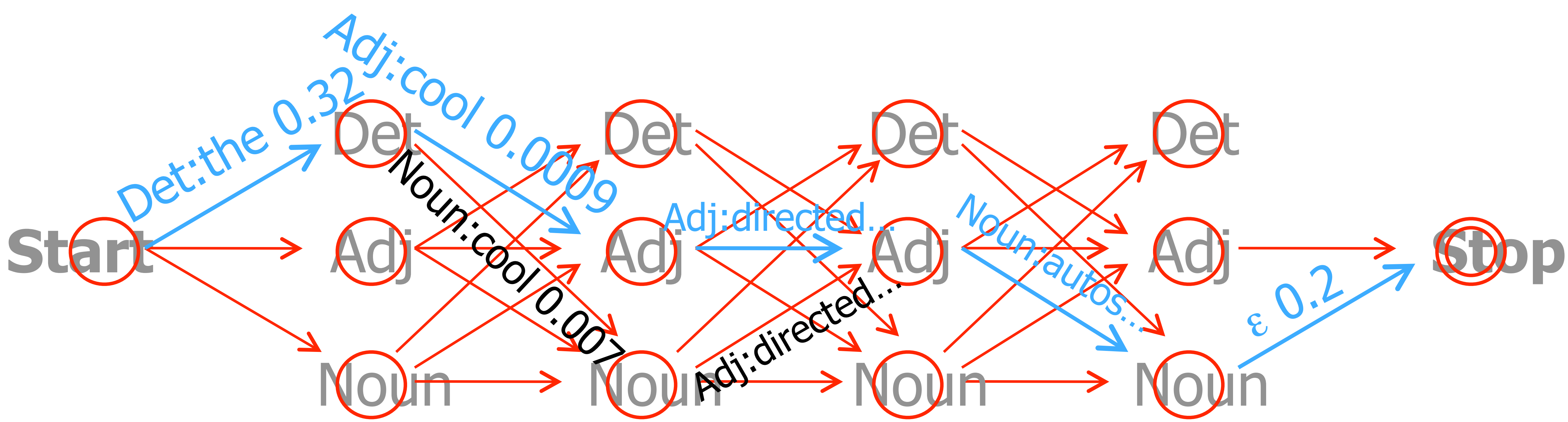
Start Det Adj Adj Noun **Stop** = 0.32 * 0.0009 …

the cool directed autos

Actually, Trellis Isn't Complete

p(word seq, tag seq)

Trellis has no Det → Det or Det →Stop arcs; why?



The best path:

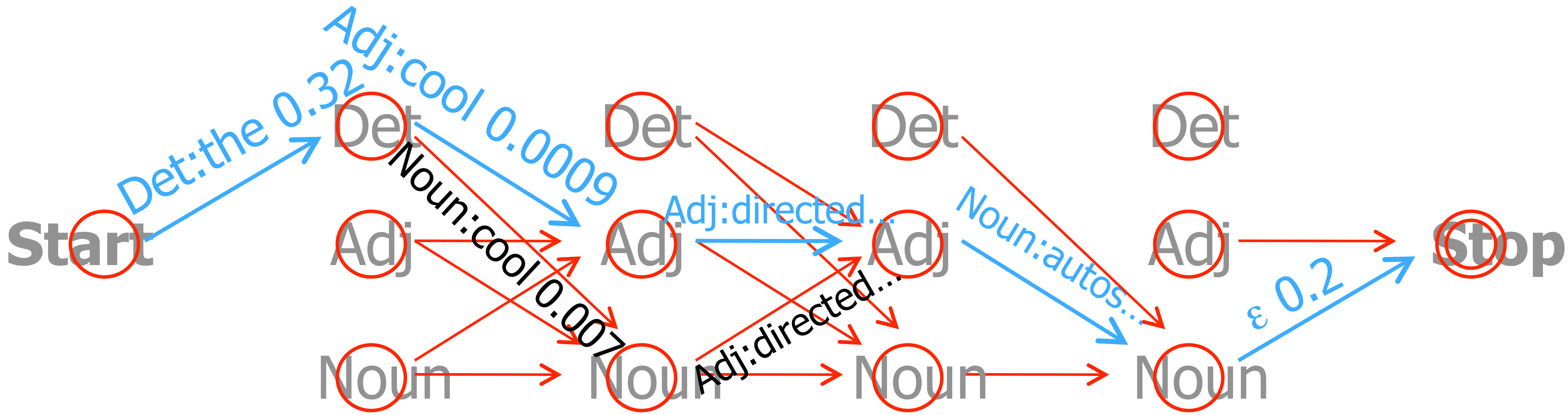
Start Det Adj Adj Noun **Stop** = 0.32 * 0.0009 …

the cool directed autos

Actually, Trellis Isn't Complete

p(word seq, tag seq)

Lattice is missing some other arcs; why?



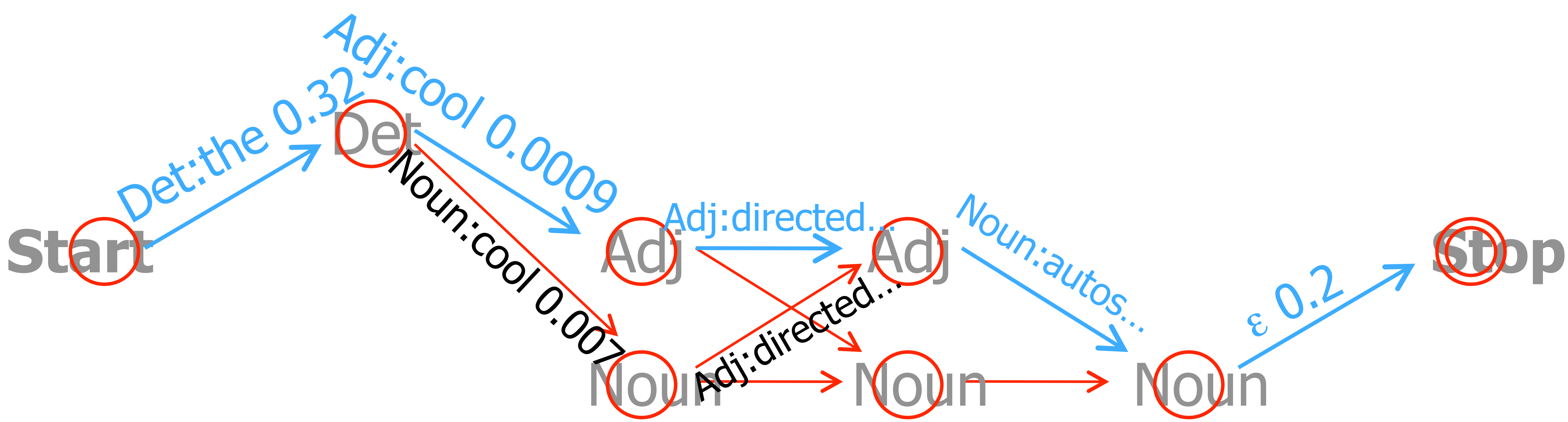
The best path:

Start Det Adj Adj Noun **Stop** = 0.32 * 0.0009 …
the cool directed autos

Actually, Trellis Isn't Complete

p(word seq, tag seq)

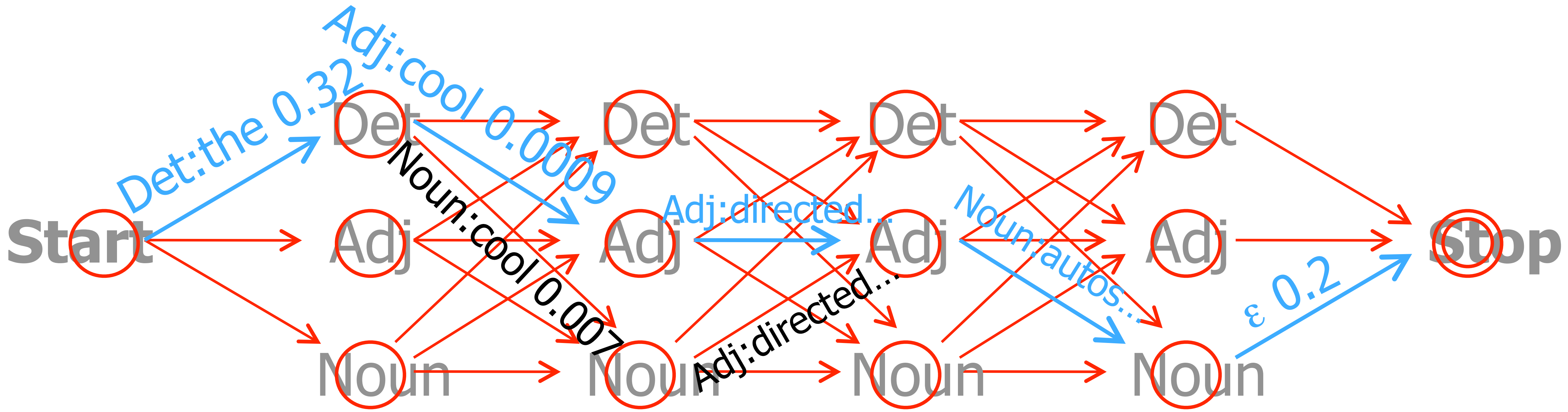
Lattice is missing some states; why?



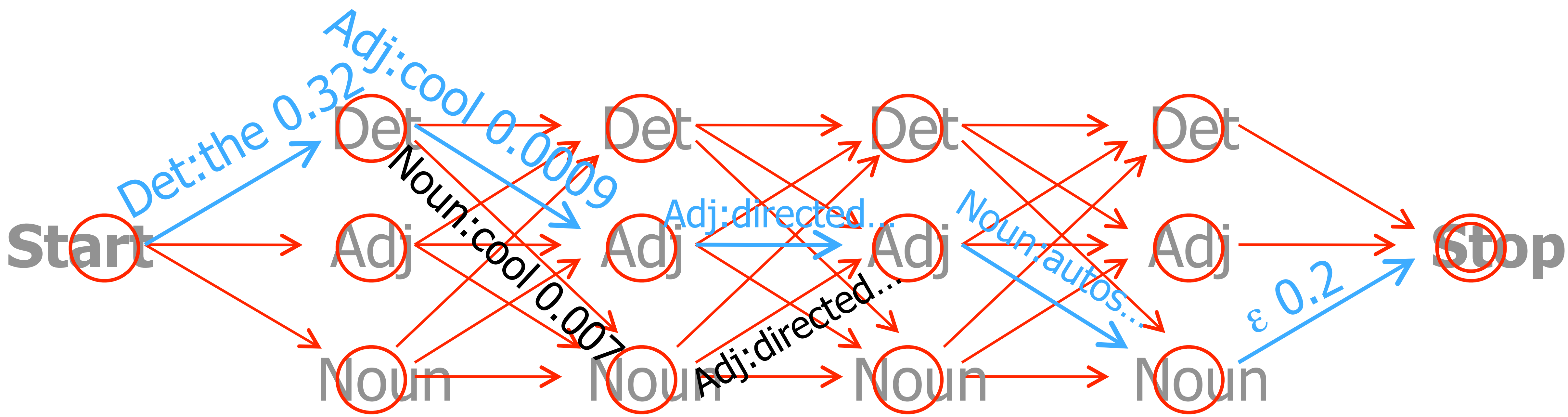
The best path:

Start Det Adj Adj Noun **Stop** = 0.32 * 0.0009 …
the cool directed autos

Find best path from Start to Stop

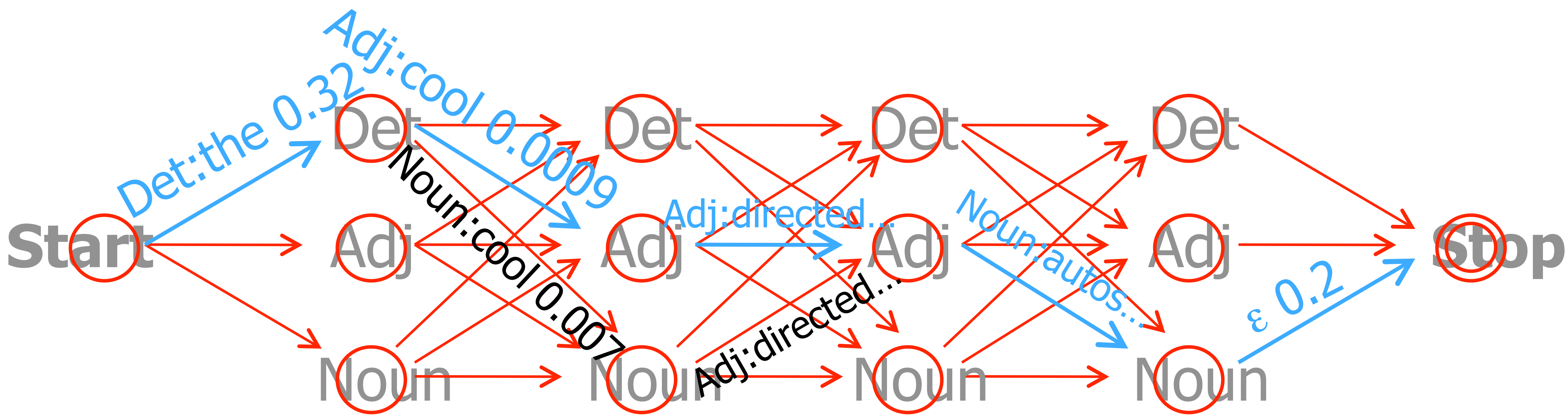


Find best path from Start to Stop



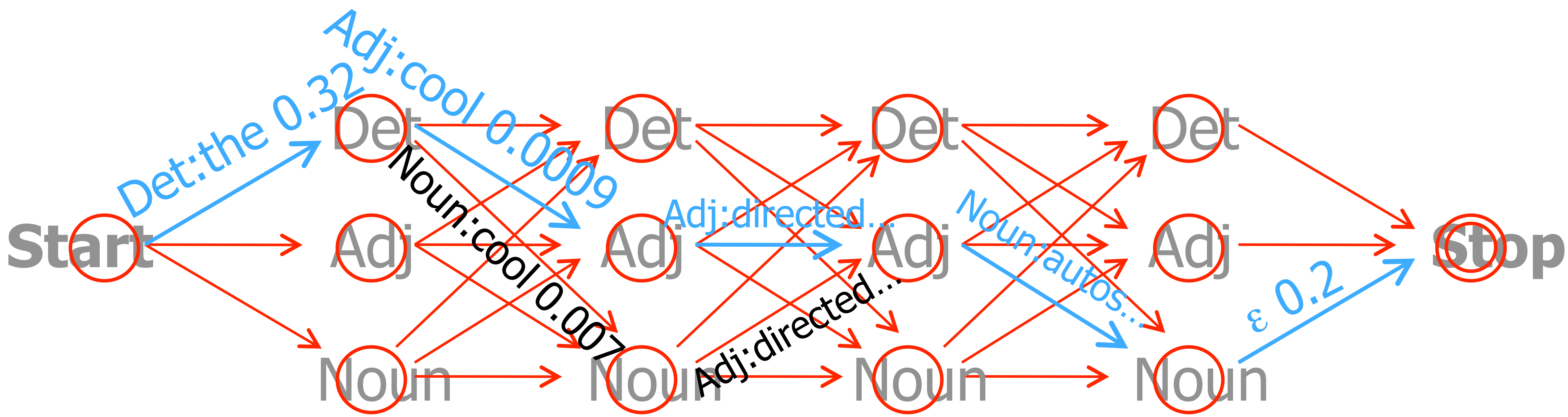
- Use dynamic programming:
  - What is best path from Start to each node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)

Find best path from Start to Stop



- Use dynamic programming:
  - What is best path from Start to each node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)
- Special acyclic case of Dijkstra's shortest-path alg.

Find best path from Start to Stop



- Use dynamic programming:
  - What is best path from Start to each node?
  - Work from left to right
  - Each node stores its best path from Start (as probability plus one backpointer)
- Special acyclic case of Dijkstra's shortest-path alg.
- Faster if some arcs/states are absent

In Summary

In Summary

- We are modeling p(word seq, tag seq)

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":

In Summary

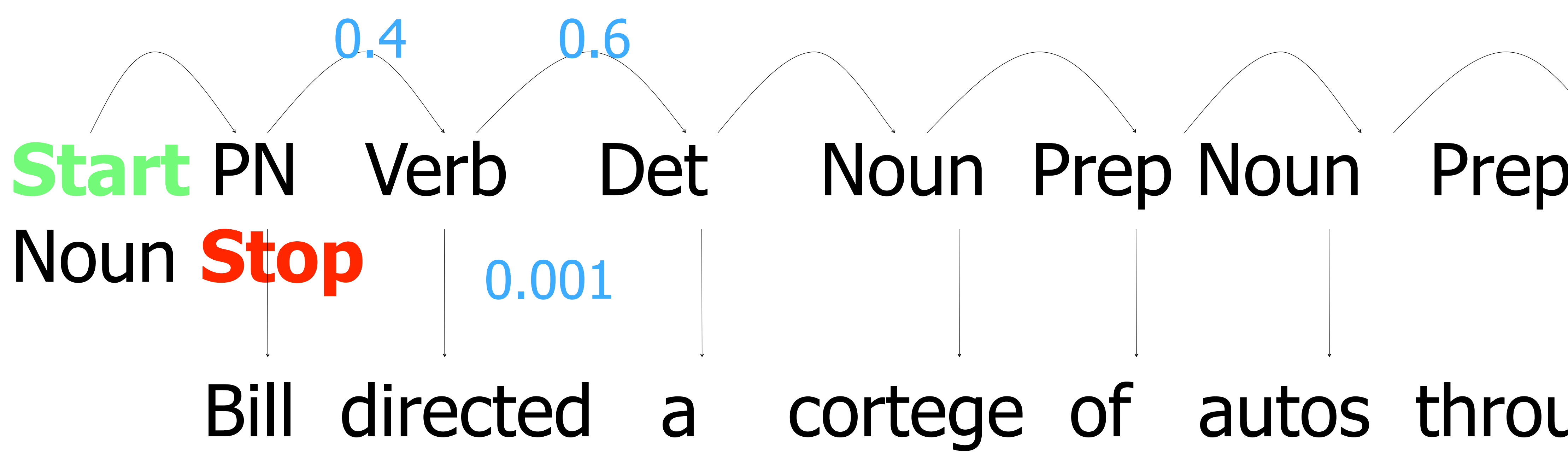
- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":

Start PN Verb Det Noun Prep Noun Prep Noun Stop

Bill directed a cortege of autos throu

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":



In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":

probs from tag bigram model

0.4 0.6

Start PN Verb Det Noun Prep Noun Prep Noun Stop

0.001

Bill directed a cortege of autos throu

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":

probs from tag bigram model

0.4 0.6

Start PN Verb Det Noun Prep Noun Prep Noun Stop

0.001

probs from unigram replacement

Bill directed a cortege of autos throu

In Summary

- We are modeling p(word seq, tag seq)
- The tags are hidden, but we see the words
- Is tag sequence X likely with these words?
- Noisy channel model is a "Hidden Markov Model":

probs from tag bigram model

0.4 0.6

Start PN Verb Det Noun Prep Noun Prep Noun Stop

probs from unigram replacement

0.001

Bill directed a cortege of autos throu

- Find X that maximizes probability product

Another Viewpoint

Another Viewpoint

- We are modeling p(word seq, tag seq)

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

p(Start PN Verb Det …
      Bill directed a …)

= p(Start) * p(PN | Start) * p(Verb | Start PN) * p(Det | Start PN Verb) * …
* p(Bill | Start PN Verb …) * p(directed | Bill, Start PN Verb Det …)
* p(a | Bill directed, Start PN Verb Det …) * …

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

p(Start PN Verb Det …
Bill directed a …)

= p(Start) * p(PN | Start) * p(Verb | Start PN) * p(Det | Start PN Verb) * …
* p(Bill | Start PN Verb …) * p(directed | Bill, Start PN Verb Det …)
* p(a | Bill directed, Start PN Verb Det …) * …

Start PN Verb Det Noun Prep Noun Prep Det Noun Stop

Bill directed a cortege of autos through the dunes

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

p(Start PN Verb Det …
        Bill directed a …)

= p(Start) * p(PN | Start) * p(Verb | ~~Start~~ PN) * p(Det | ~~Start PN~~ Verb) * …
  * p(Bill | Start PN Verb …) * p(directed | Bill, Start PN Verb Det …)
  * p(a | Bill directed, Start PN Verb Det …) * …

Start PN Verb Det Noun Prep Noun Prep Det Noun **Stop**

Bill directed a cortege of autos through the dunes

Another Viewpoint

- We are modeling p(word seq, tag seq)
- Why not use chain rule + some kind of backoff?
- Actually, we are!

p(Start PN Verb Det …
Bill directed a …)

= p(Start) * p(PN | Start) * p(Verb | ~~Start~~ PN) * p(Det | ~~Start PN~~ Verb) * …
* p(Bill | ~~Start~~ PN ~~Verb~~ …) * p(directed | ~~Bill,~~ ~~Start PN~~ Verb ~~Det~~ …)
* p(a | ~~Bill directed, Start PN Verb~~ Det …) * …

Start PN Verb Det Noun Prep Noun Prep Det Noun **Stop**

Bill directed a cortege of autos through the dunes

Variations

Variations

- Multiple tags per word

Variations

- Multiple tags per word
  - Transformations to knock some of them out

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for partly supervised learning

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for partly supervised learning
  - Supervised: You have a tagged training corpus

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for partly supervised learning
  - Supervised: You have a tagged training corpus
  - Unsupervised: You have an untagged training corpus

Variations

- Multiple tags per word
  - Transformations to knock some of them out
- How to encode multiple tags and knockouts?

- Use the above for partly supervised learning
  - Supervised: You have a tagged training corpus
  - Unsupervised: You have an untagged training corpus
  - Here: You have an untagged training corpus and a dictionary giving possible tags for each word

Applications of HMMs

- NLP
  - Part-of-speech tagging
  - Word segmentation
  - Information extraction
  - Optical character recognition
- Speech recognition
  - Modeling acoustics, with continuous emissions
- Computer Vision
  - Gesture recognition
- Biology
  - Gene finding
  - Protein structure prediction
- Economics, Climatology, Robotics, etc.

A More Traditional View of HMMs

Recipe for NLP

Input: `the lead paint is unsafe` **Observations**

Output: `the/Det lead/N paint/N is/V unsafe/Adj` **Tags**

1) **Data**: Notation, representation
2) **Problem**: Write down the problem in notation
3) **Model**: Make some assumptions, define a parametric model (often generative model of the data)
4) **Inference**: How to search through possible answers to find the best one
5) **Learning**: How to estimate parameters
6) **Implementation**: Engineering considerations for an efficient implementation

An HMM Tagger

- View sequence of tags as a Markov chain. Assumptions:
  - Limited horizon $P(x_{t+1}|x_1, ...x_t) = P(x_{t+1}|x_t)$
  - Time invariant (stationary) $P(x_{t+1}|x_t) = P(x_2|x_1)$
  - We assume that a word's tag only depends on the previous tag (limited horizon) and that his dependency does not change over time (time invariance)
  - A state (part of speech) generates a word. We assume it depends only on the state.

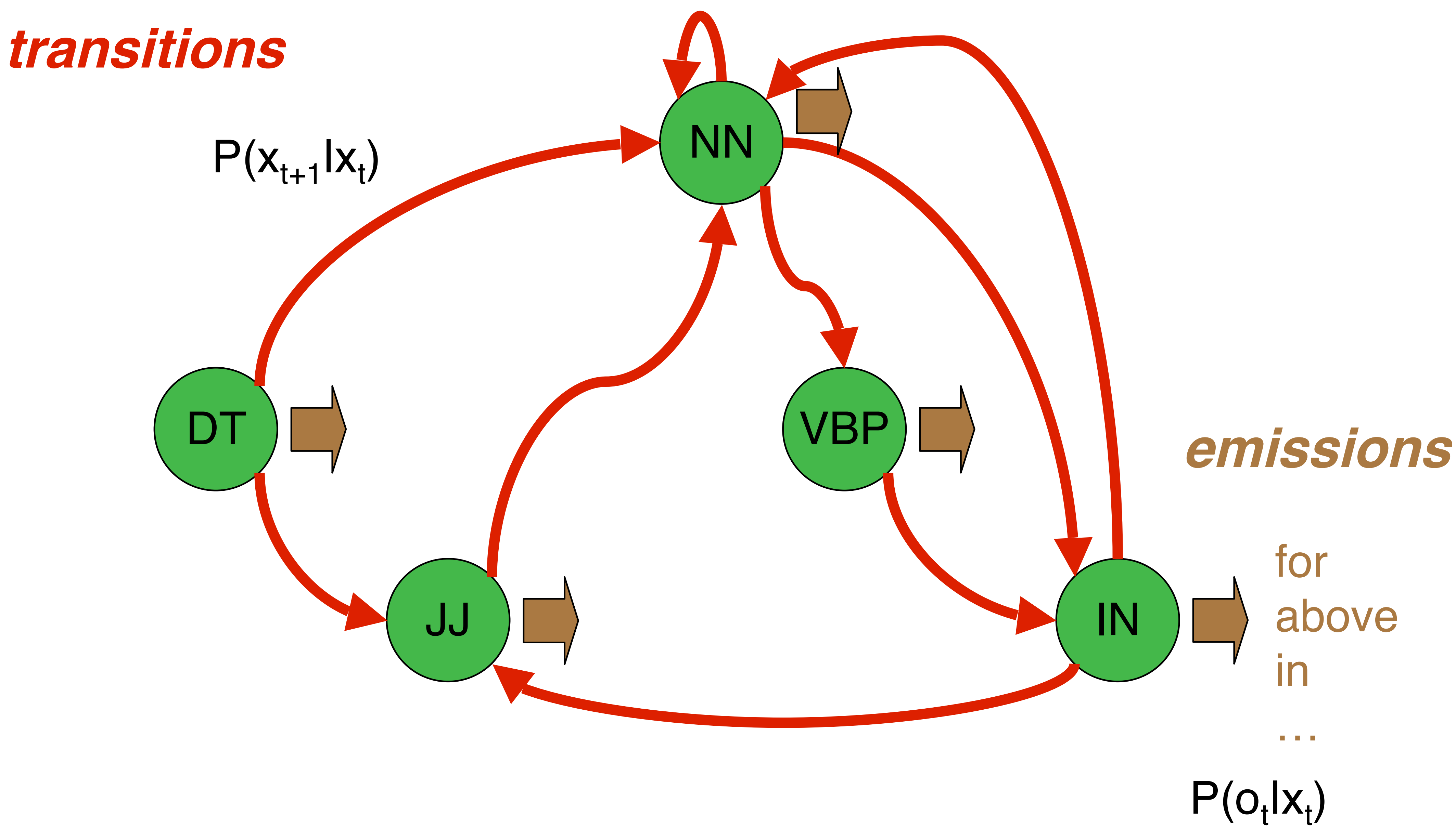
$$P(o_t|x_1, ...x_T, o_1, ...o_{t-1}) = P(o_t|x_t)$$

The Markov Property

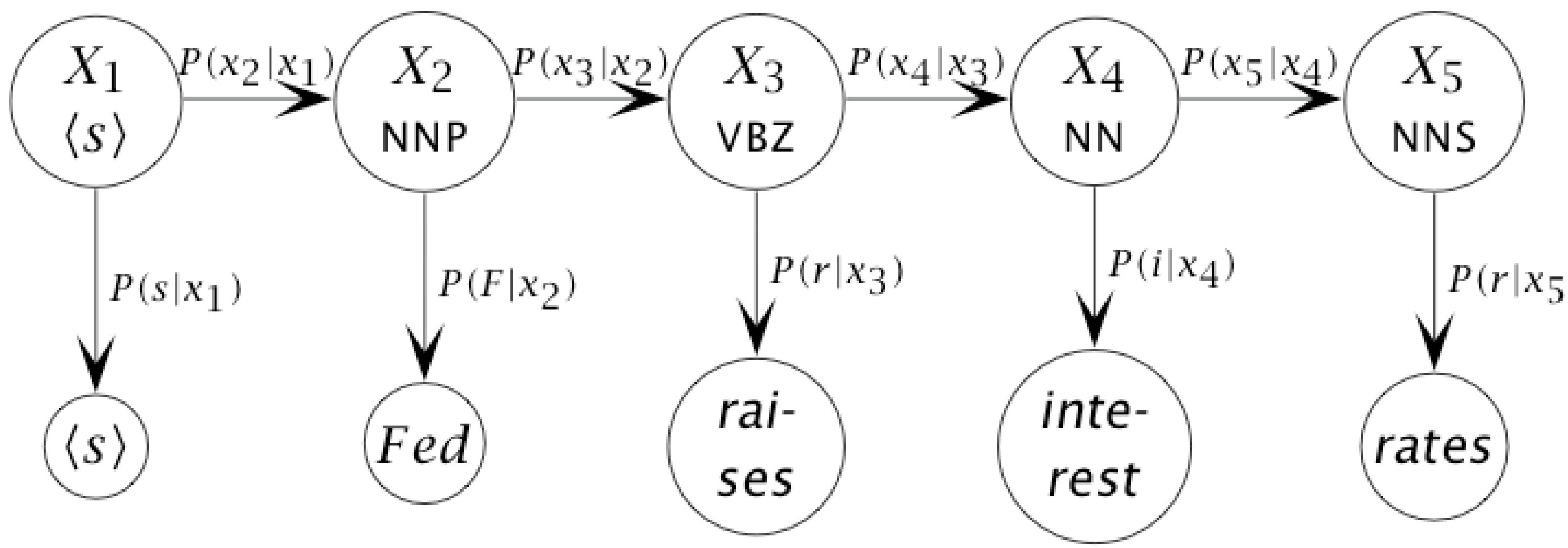
- A stochastic process has the **Markov property** if the conditional probability distribution of future states of the process, given the current state, depends only upon the current state, and conditionally independent of the past states (the *path* of the process) given the current state.
- A process with the Markov property is usually called a **Markov process**, and may be described as *Markovian*.

$$\Pr\big[X(t+h) = y \,|\, X(s) = x(s), s \leq t\big] = \Pr\big[X(t+h) = y \,|\, X(t) = x(t)\big], \quad \forall h > 0.$$

HMM w/State Emissions



HMM as Bayes Net



- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations (words)

(One) Standard HMM Formalism

- *(X, O, x_s, A, B) are all variables. Model* μ = *(A, B)*
- X is state sequence of length T; O is observation seq.
- x_s is a designated start state (with no incoming transitions). (Can also be separated into π as in book.)
- A is matrix of transition probabilities (each row is a conditional probability table (CPT)
- B is matrix of output probabilities (vertical CPTs)

$$P(X, O|\mu) = \prod_{t=1}^{T} a[x_t|x_{t-1}]\ b[o_t|x_t]$$

- HMM is a probabilistic (nondeterministic) finite state automaton, with probabilistic outputs (from vertices, not arcs, in the simple case)

HMM Inference Problems

- **Given an observation sequence, find the most likely state sequence** (tagging)
- Compute the probability of observations when state sequence is hidden (language modeling)
- Given observations and (optionally) a their corresponding states, find parameters that maximize the probability of the observations (parameter estimation)

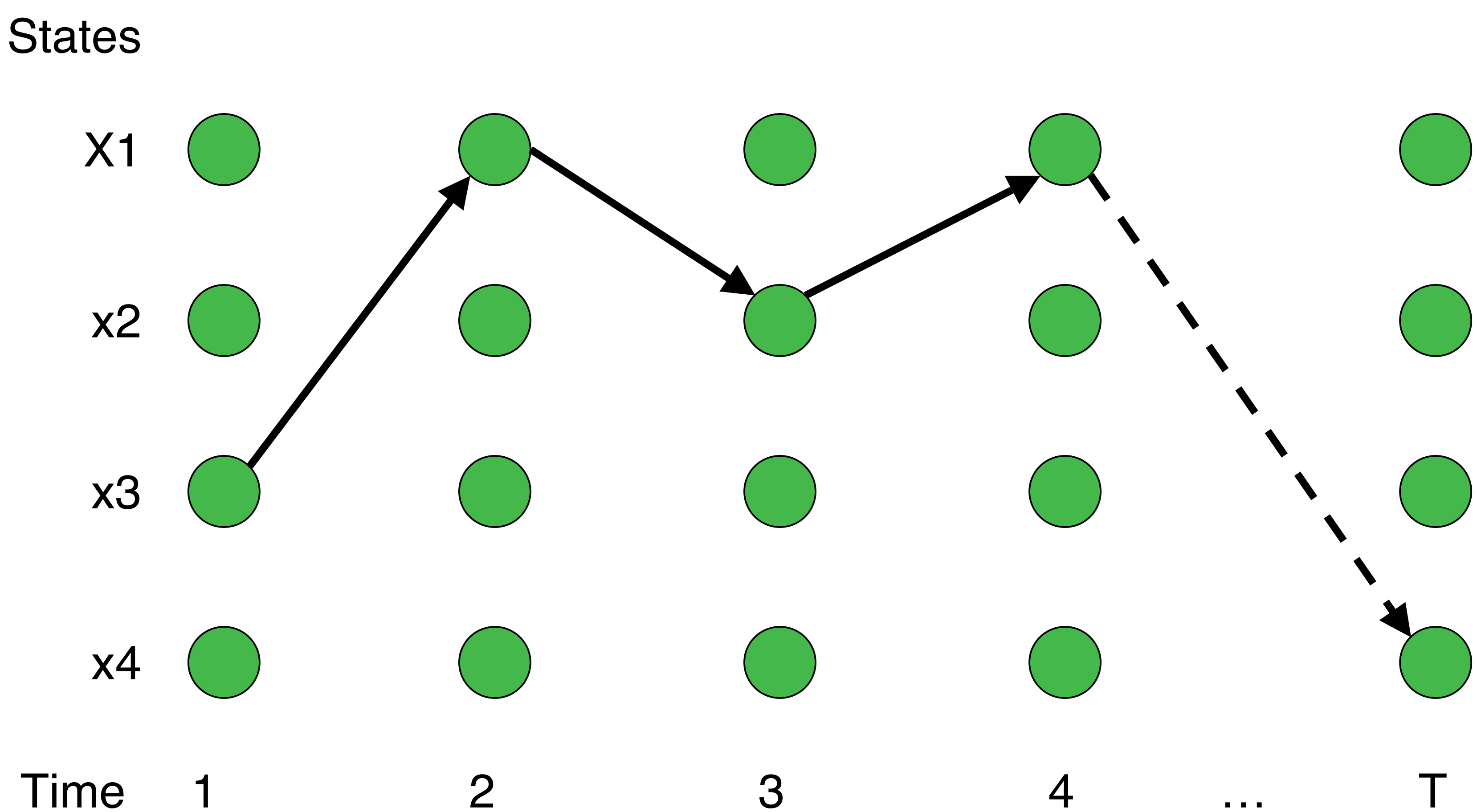
Most Likely State Sequence

- Given $O = (o_1,\ldots,o_T)$ and model $\mu = (A,B)$
- We want to find

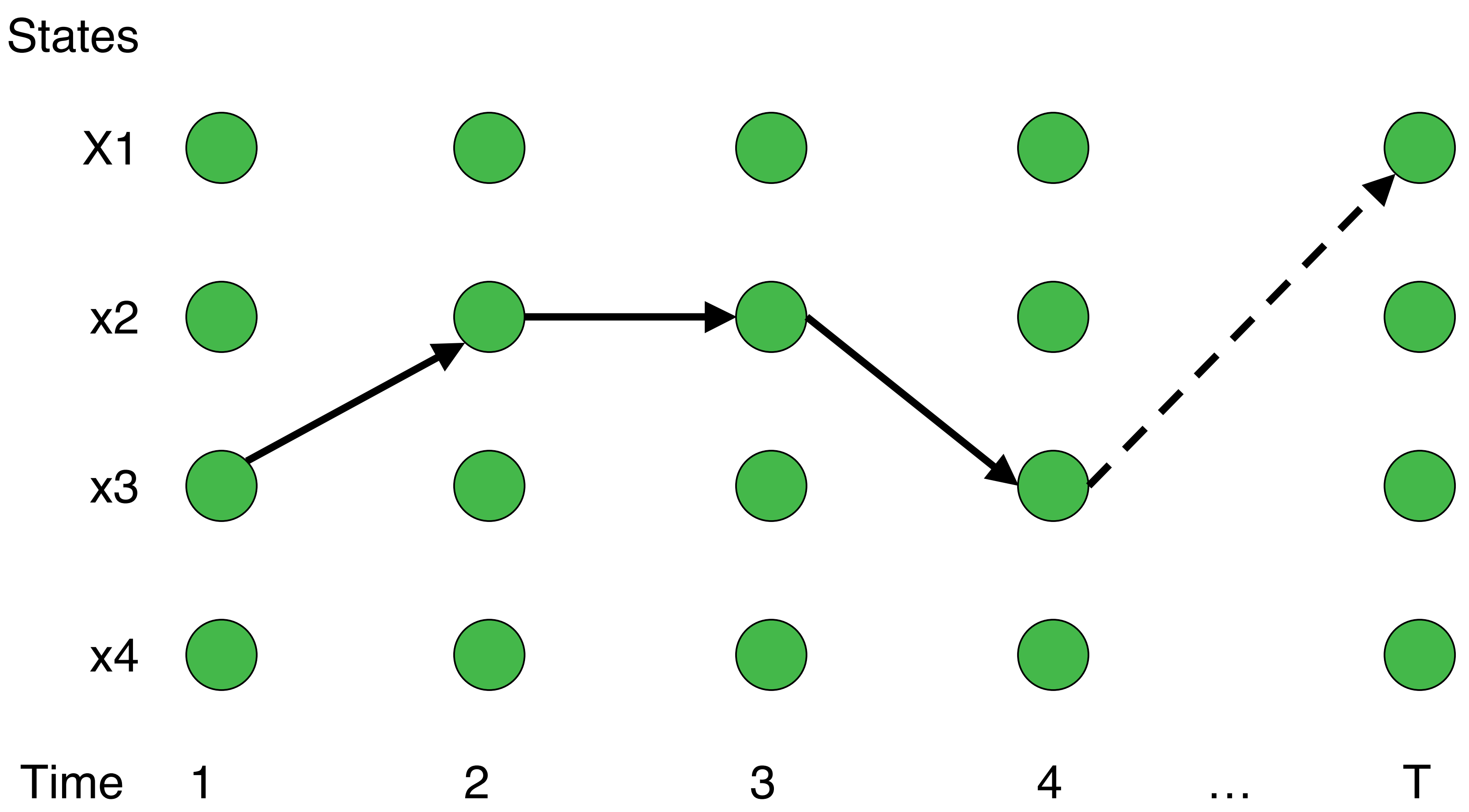
$$\arg\max_X P(X|O,\mu) = \arg\max_X \frac{P(X,O|\mu)}{P(O|\mu)} = \arg\max_X P(X,O|\mu)$$

- $P(O,X|\ \mu) = P(O|X,\ \mu)\ P(X|\ \mu\ )$
- $P(O|X,\ \mu) = b[x_1|o_1]\ b[x_2|o_2]\ \ldots\ b[x_T|o_T]$
- $P(X|\ \mu) = a[x_1|x_2]\ a[x_2|x_3]\ \ldots\ a[x_{T-1}|x_T]$
- $\arg\max_X P(O,X|\ \mu) = \arg\max x_1, x_2,\ldots x_T$
- Problem: arg max is exponential in sequence length!

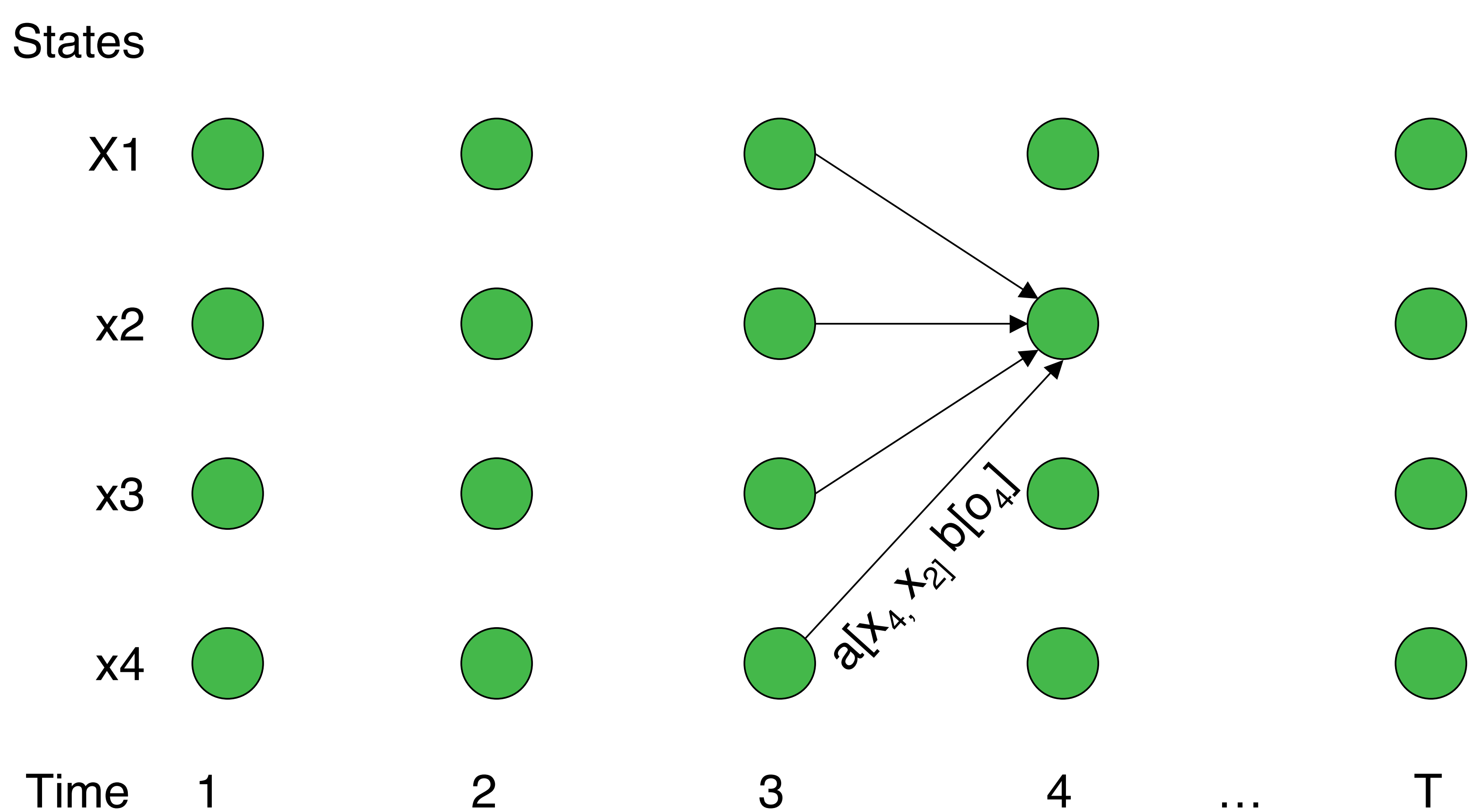
Paths in a Trellis



Paths in a Trellis



Paths in a Trellis



$\delta_i(t)$ = Probability of most likely path that ends at state *i* at time *t*.

Dynamic Programming

- Efficient computation of max over all states
- Intuition: Probability of the first *t* observations is the same for all possible *t+1* length sequences.
- Define forward score:

$$\delta_i(t) = \max_{x_1 \dots x_{t-1}} P(o_1 o_2 \dots o_t, x_1 \dots x_{t-1}, x_t = i | \mu)$$

$$\delta_j(t+1) = \max_{i=1..N} \delta_i(t) a[x_j | x_i]\, b[o_{t+1} | x_j]$$

- Compute it recursively from the beginning
- (Then must remember best paths to get arg max.)

The Viterbi Algorithm (1967)

- Used to efficiently find the state sequence that gives the highest probability to the observed outputs
- Maintains two dynamic programming tables:
  - The probability of the best path (max)

$$\delta_j(t+1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i]\ b[o_{t+1}|x_j]$$

  - The state transitions of the best path (arg)

$$\psi_j(t+1) = \arg \max_{i=1..N} \delta_i(t) a[x_j|x_i]\ b[o_{t+1}|x_j]$$

- Note that this is different from finding the most likely tag for each time *t*!

Viterbi Recipe

- Initialization

$$\delta_j(0) = 1 \text{ if } x_j = x_S. \quad \delta_j(0) = 0 \text{ otherwise.}$$

- Induction

$$\delta_j(t+1) = \max_{i=1..N} \delta_i(t) a[x_j|x_i]\ b[o_{t+1}|x_j]$$

Store backtrace

$$\psi_j(t+1) = \arg\max_{i=1..N} \delta_i(t) a[x_j|x_i]\ b[o_{t+1}|x_j]$$

- Termination and path readout

$$\hat{x}_T = \arg\max_{i=1..N} \delta_i(T)$$

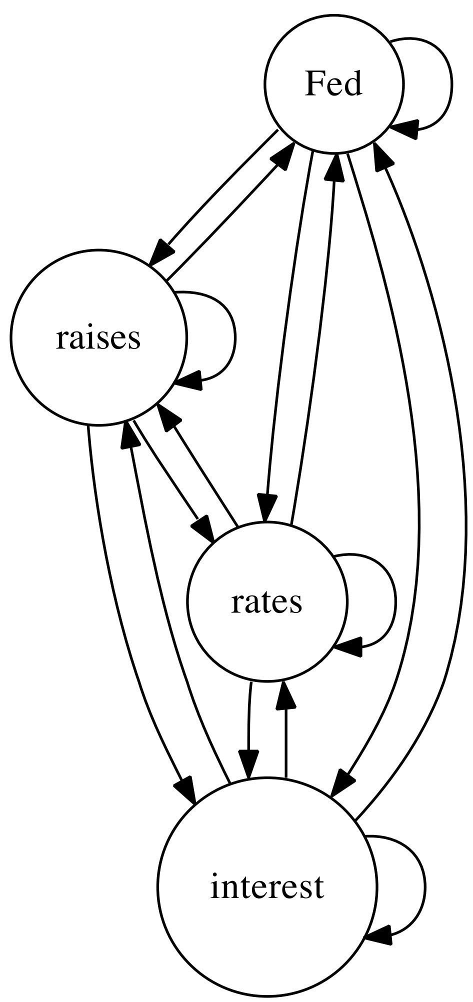
$$\hat{x}_t = \psi_{\hat{x}_{t+1}}(t+1)$$

Probability of entire best seq.

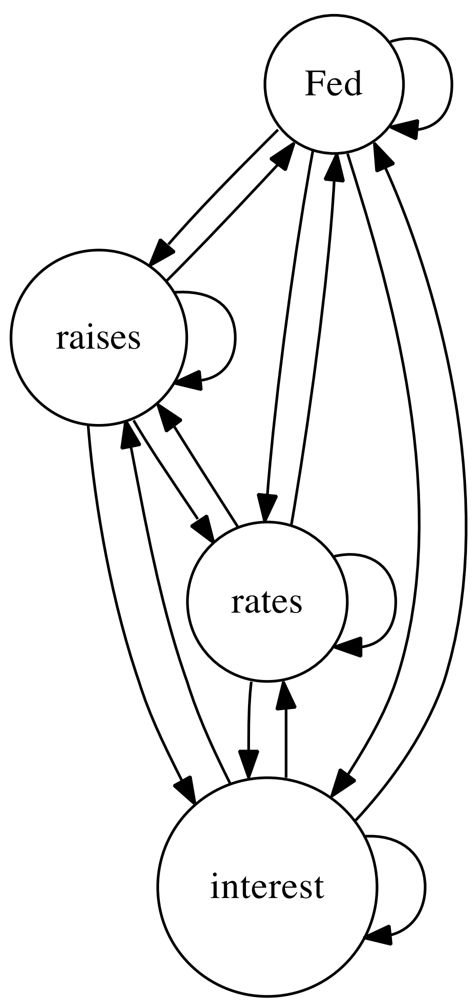
$$P(\hat{X}) = \max_{i=1..N} \delta_i(T)$$

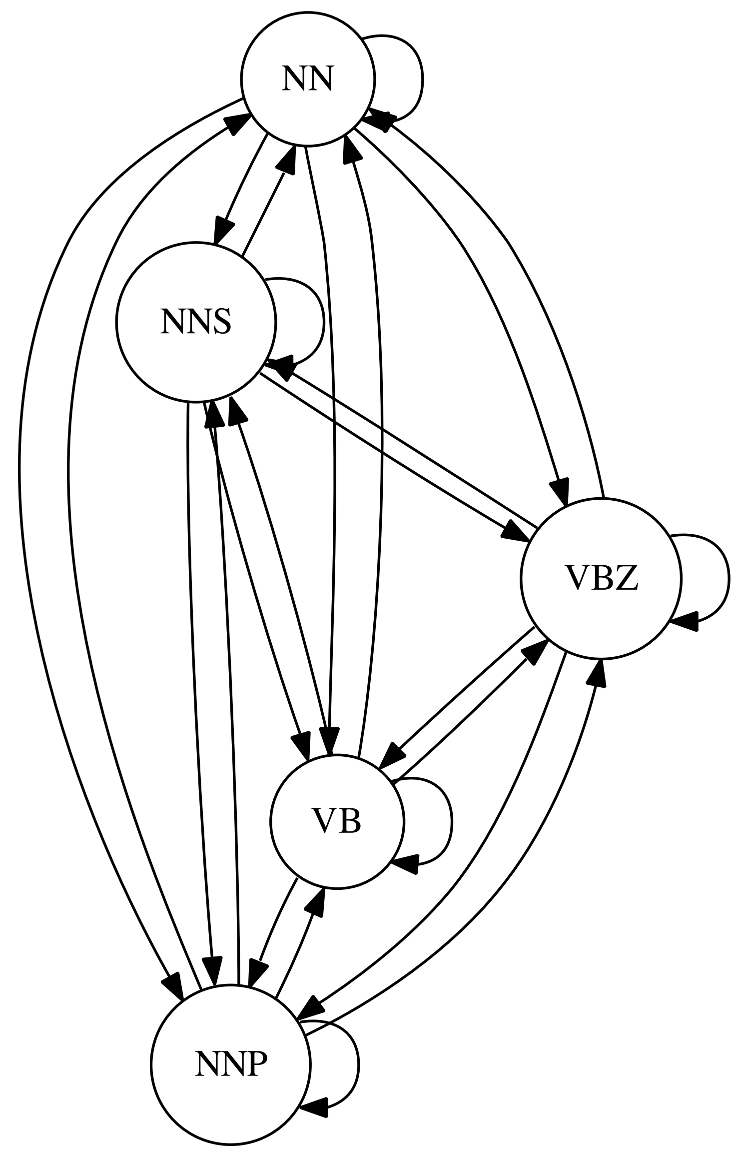
HMMs:
Maxing and Summing

Markov vs. Hidden Markov Models

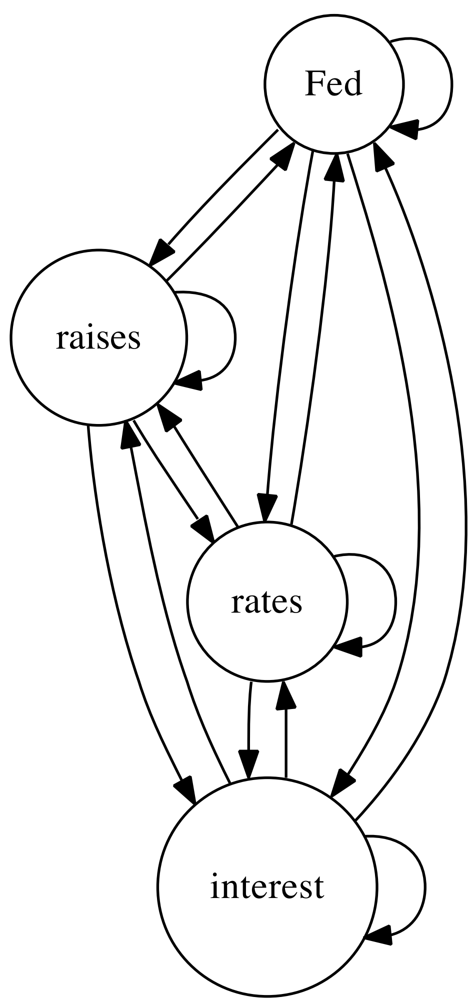


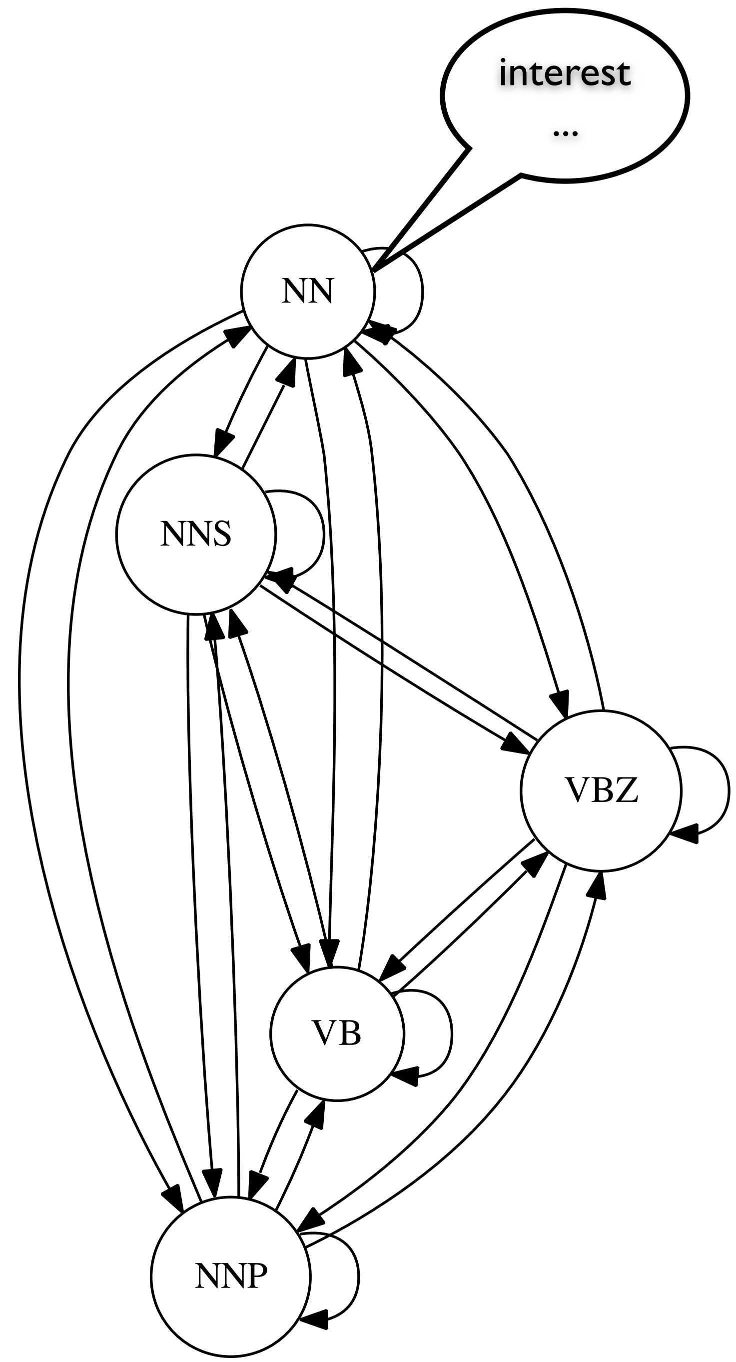
Markov vs. Hidden Markov Models



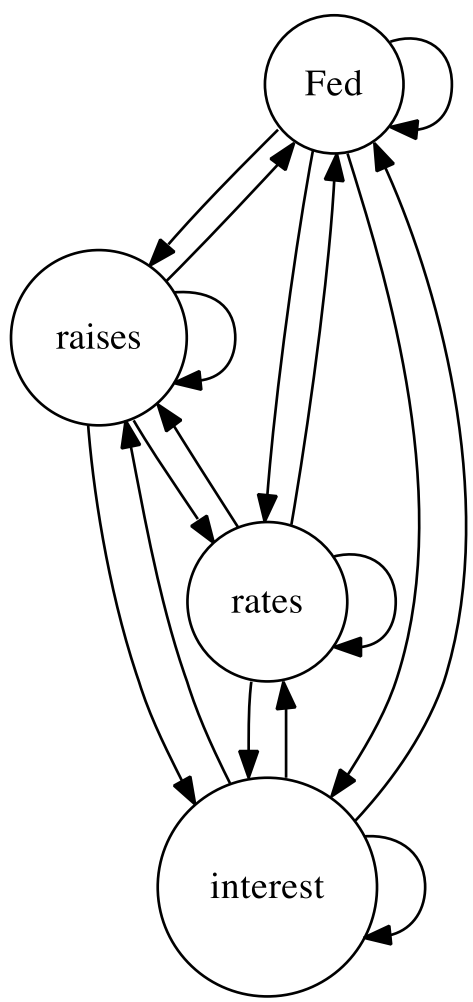


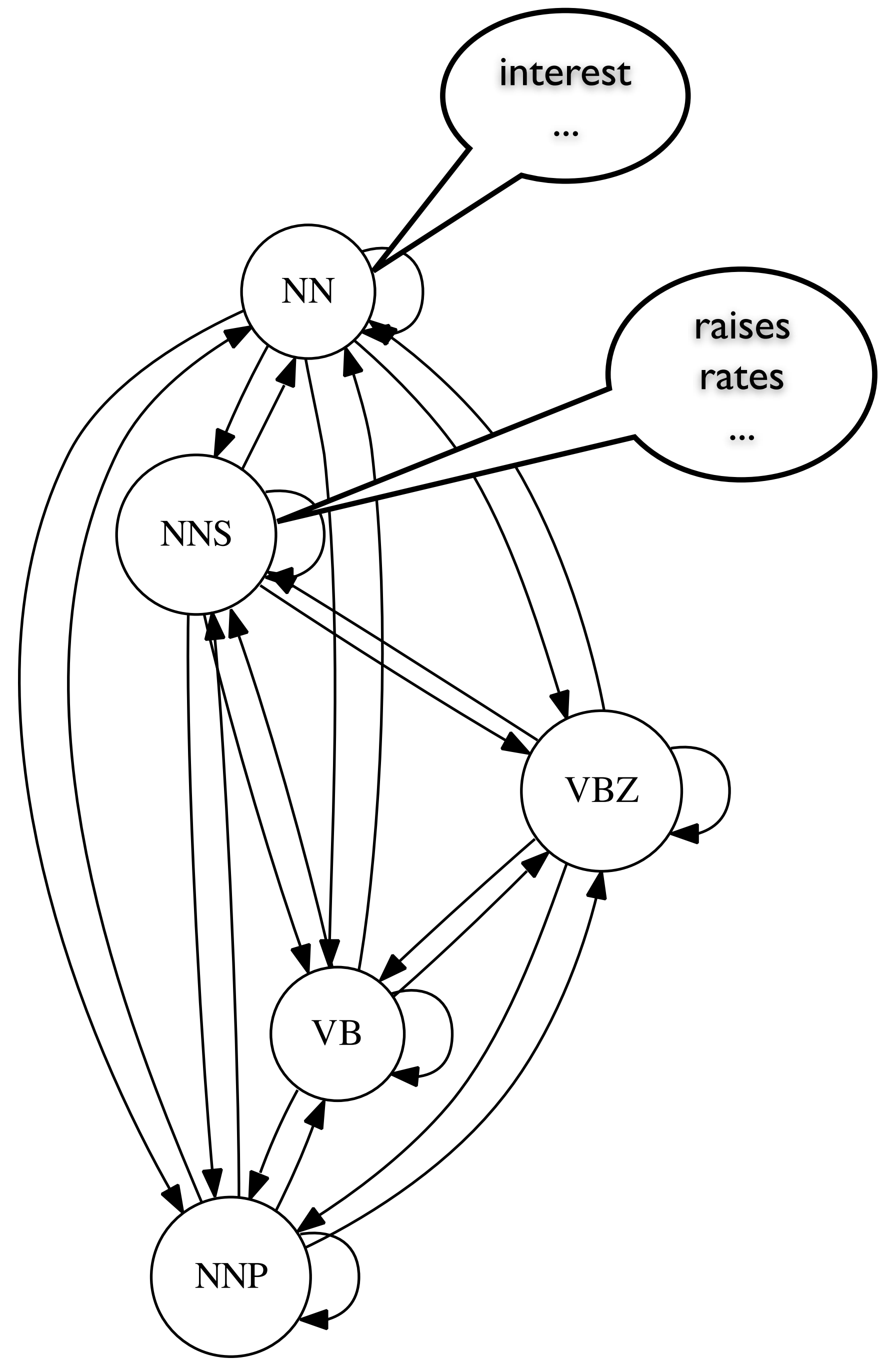
Markov vs. Hidden Markov Models



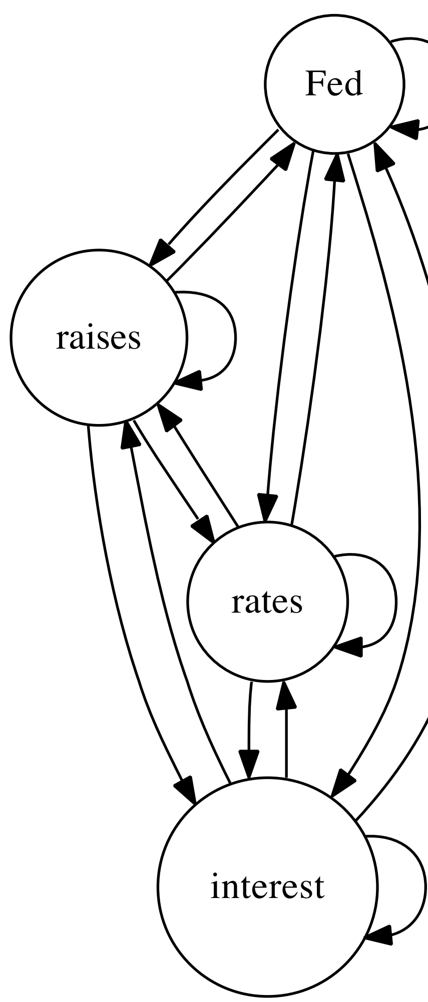


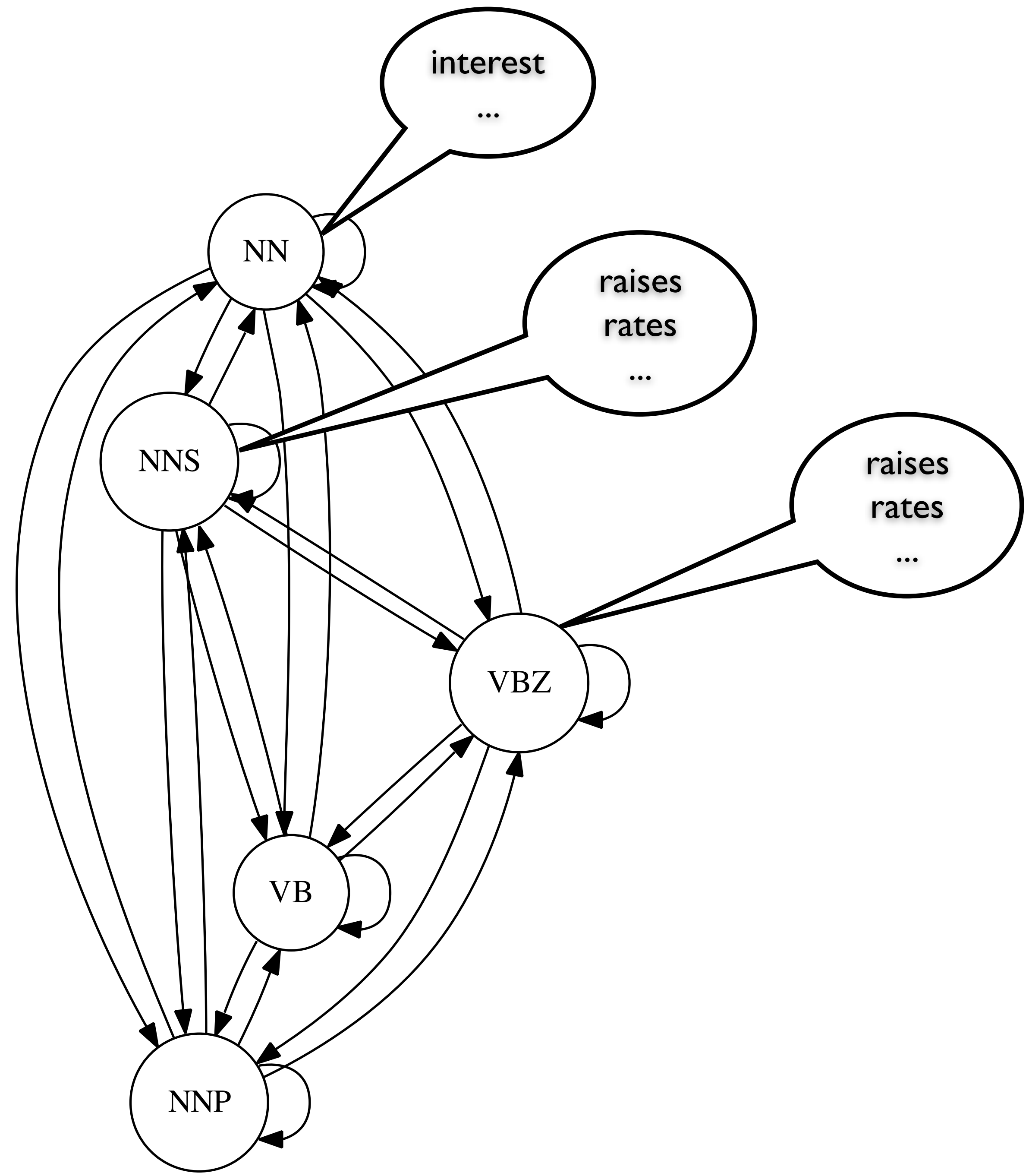
Markov vs. Hidden Markov Models



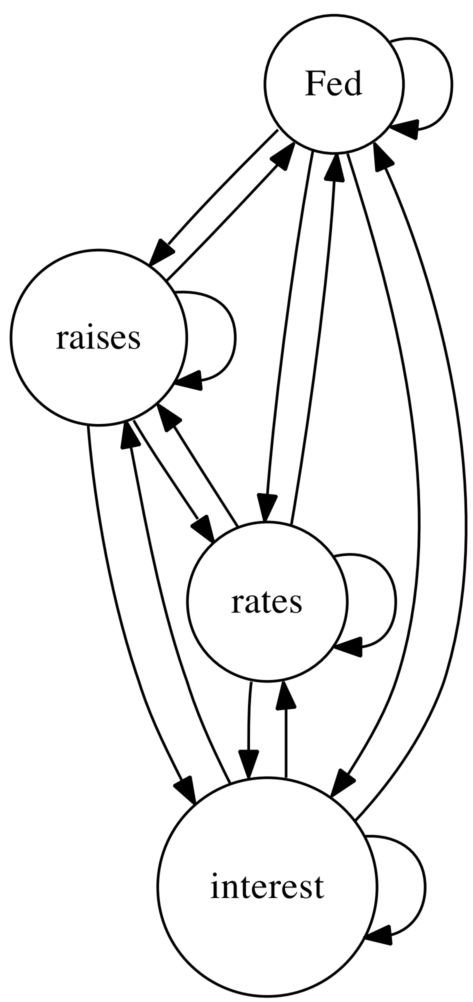


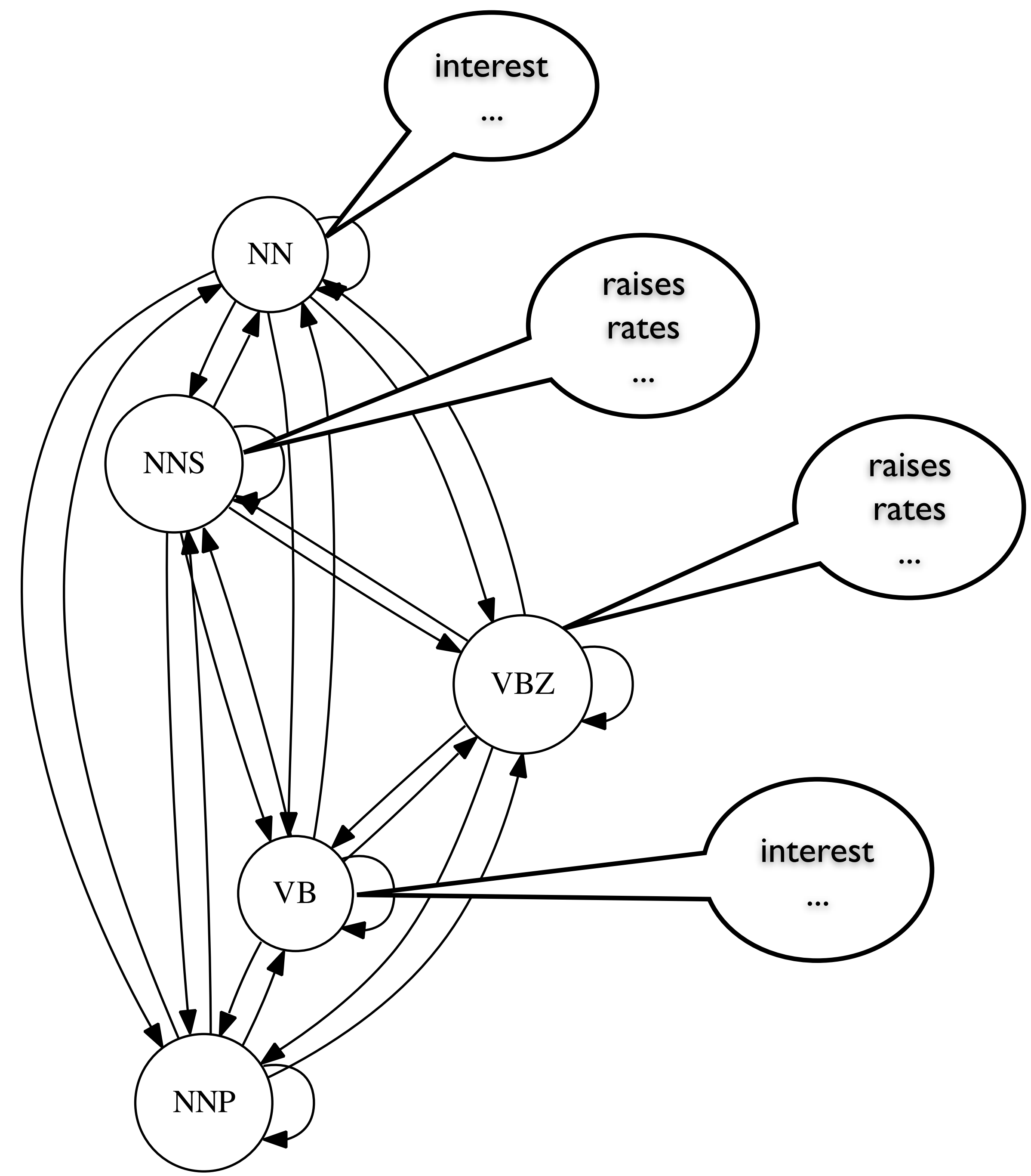
Markov vs. Hidden Markov Models



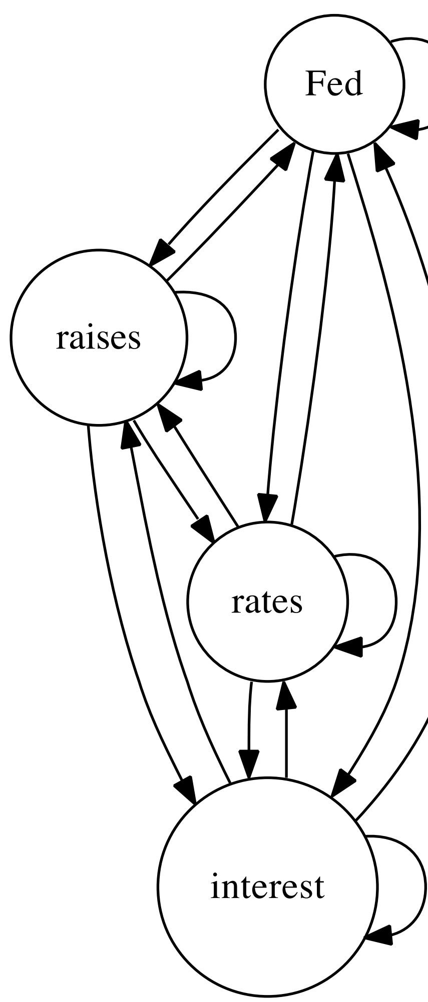


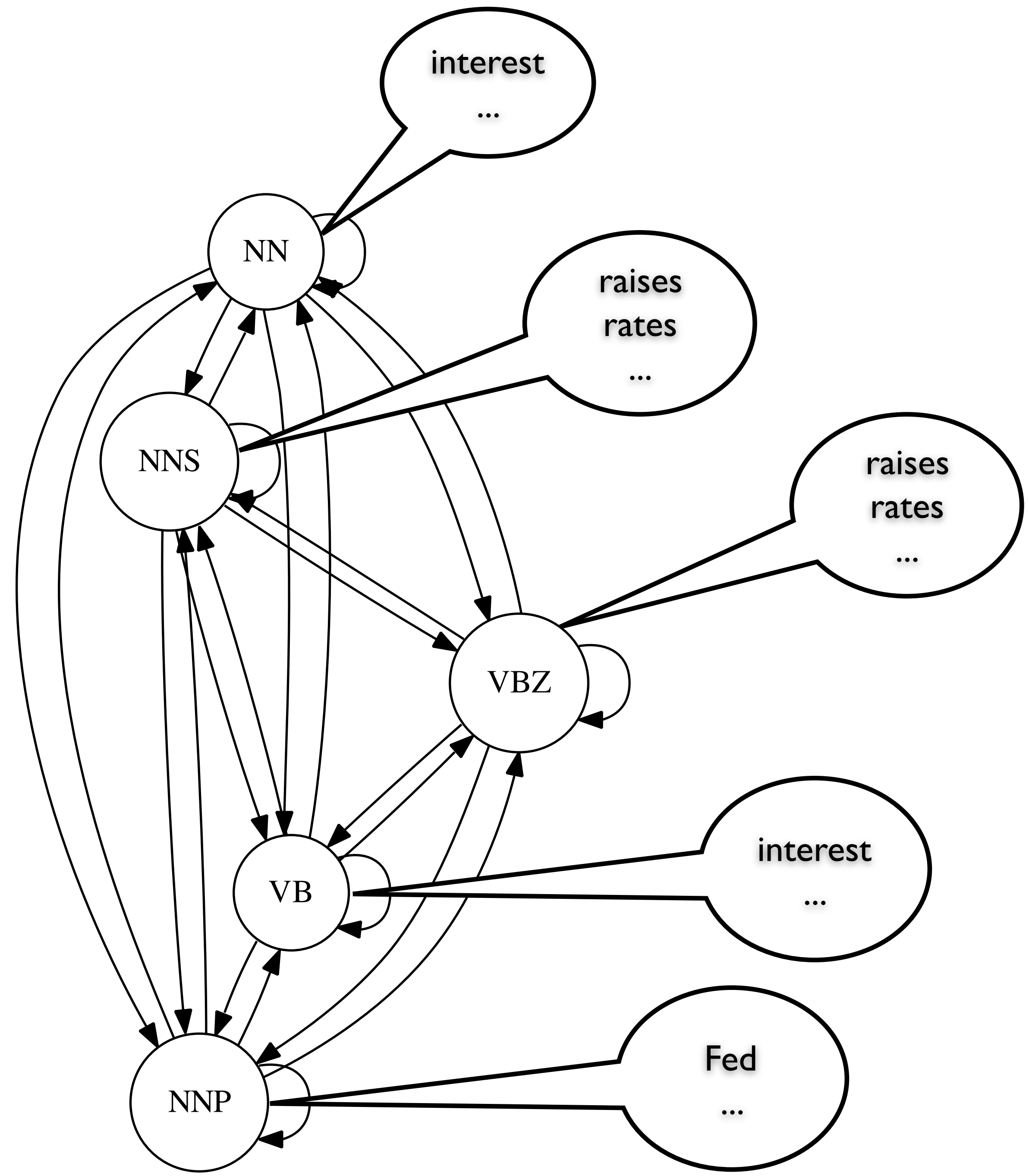
Markov vs. Hidden Markov Models



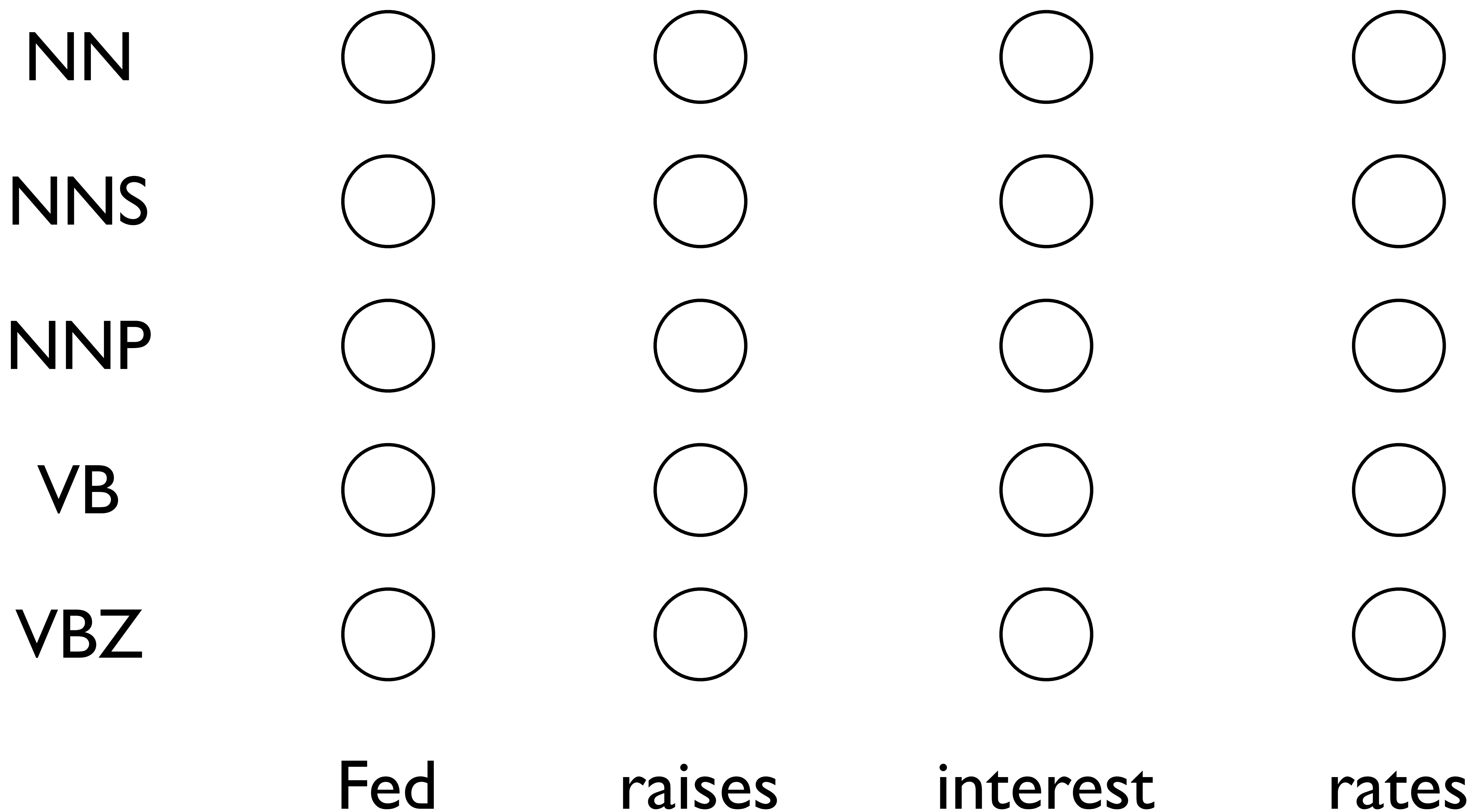


Markov vs. Hidden Markov Models





Unrolled into a Trellis



HMM Inference Problems

- Given an observation sequence, find the most likely state sequence (tagging)
- Compute the probability of observations when state sequence is hidden (language modeling)
- Given observations and (optionally) a their corresponding states, find parameters that maximize the probability of the observations (parameter estimation)

Tagging

Given an observation sequence, find the most likely state sequence.

$$\arg\max_X P(X \mid O, \mu) = \arg\max_X \frac{P(X, O \mid \mu)}{P(O \mid \mu)} = \arg\max_X P(X, O \mid \mu)$$

$$\arg\max_{x_1, x_2, \ldots x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)$$

Last time: Use dynamic programming to find highest-probability sequence (i.e. best path, like Dijsktra's algorithm)

Language Modeling

Compute the probability of observations when state sequence is hidden.

$$P(X, O \mid \mu) = P(O \mid X, \mu)P(X \mid \mu)$$

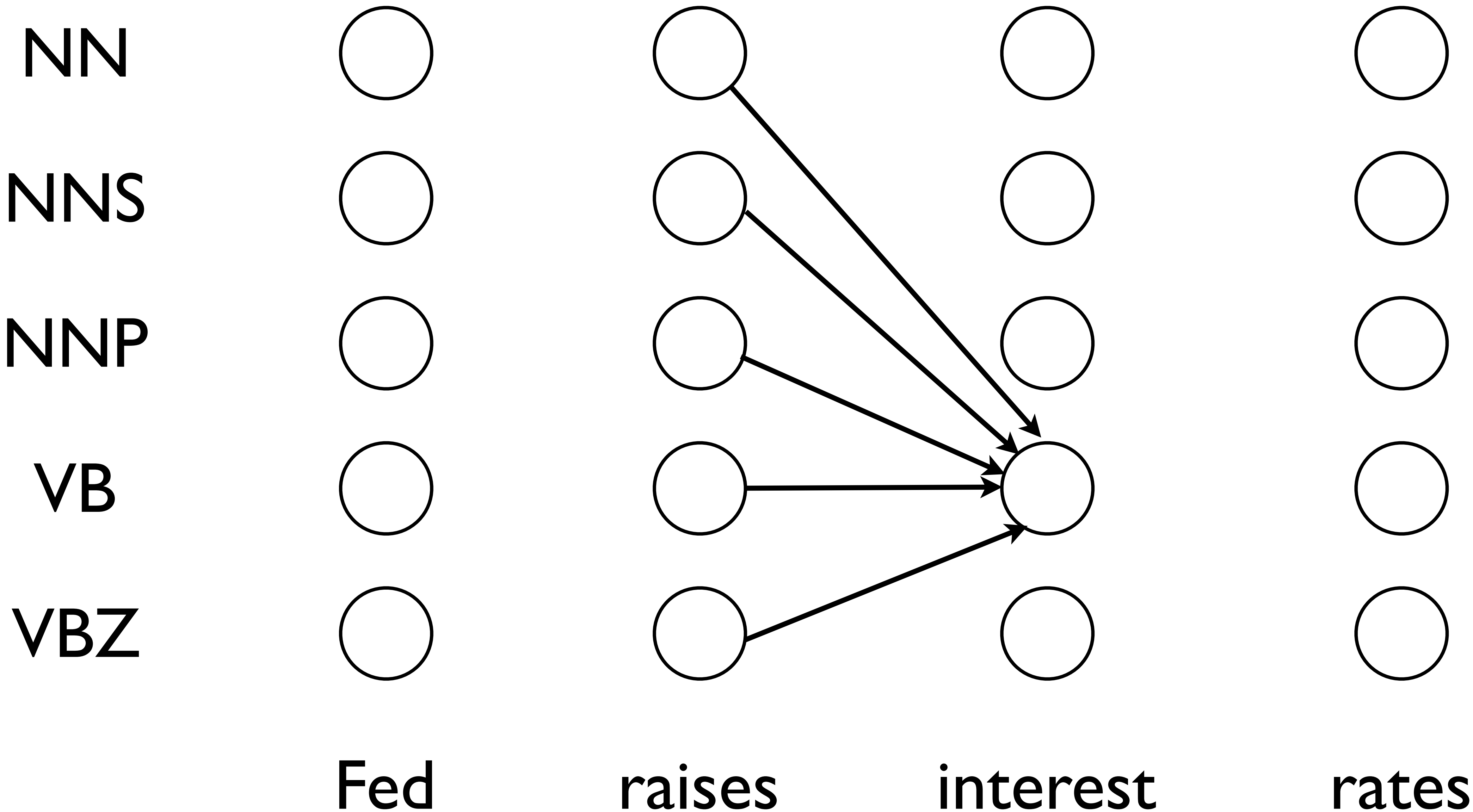
Therefore

$$P(O \mid \mu) = \sum_{X} P(O \mid X, \mu)P(X \mid \mu)$$

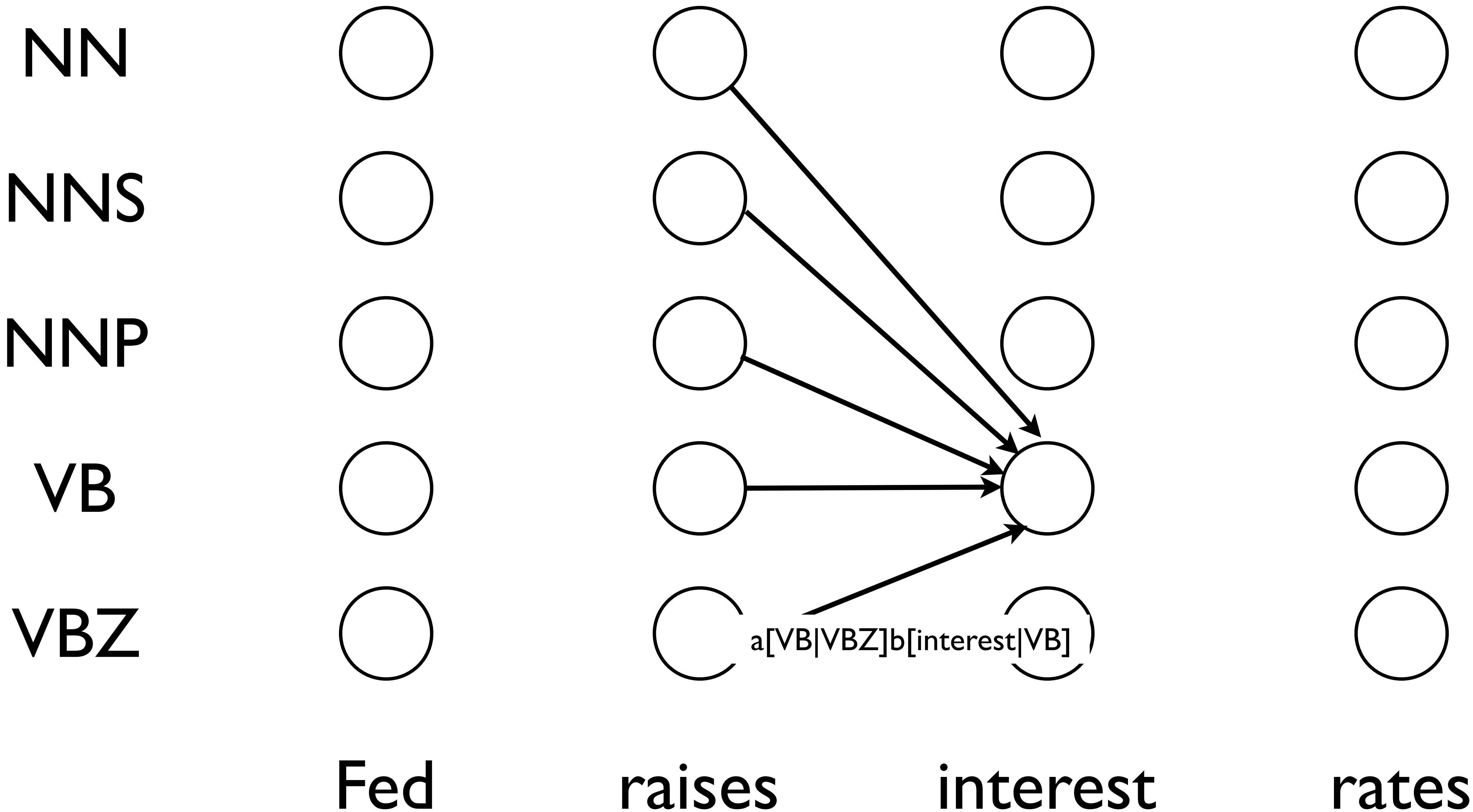
$$\sum_{x_1, x_2, \ldots x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)$$

Suspiciously similar to $\max_{x_1, x_2, \ldots x_T} P(x_1, x_2, \ldots, x_T, O \mid \mu)$

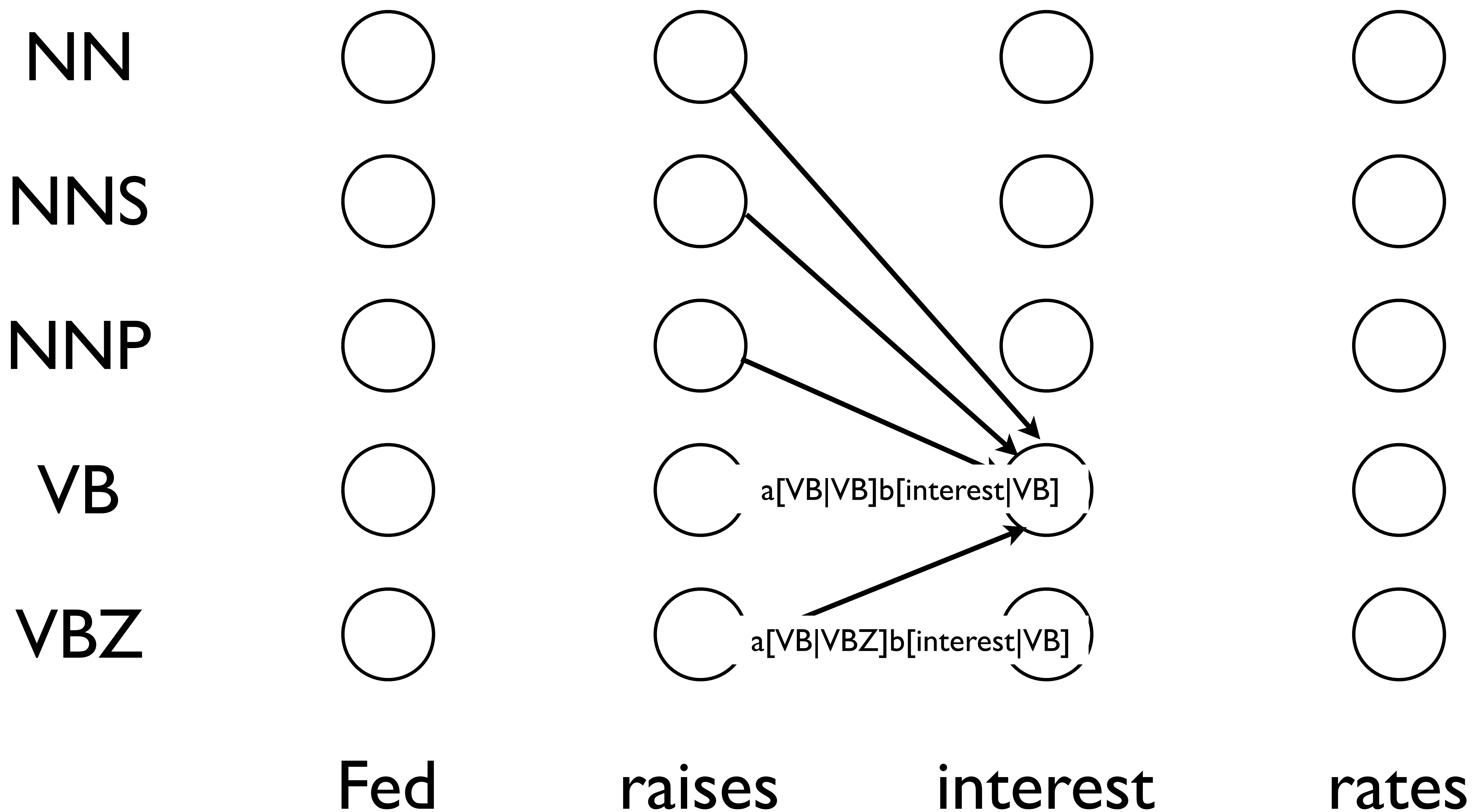
Viterbi Algorithm (Tagging)



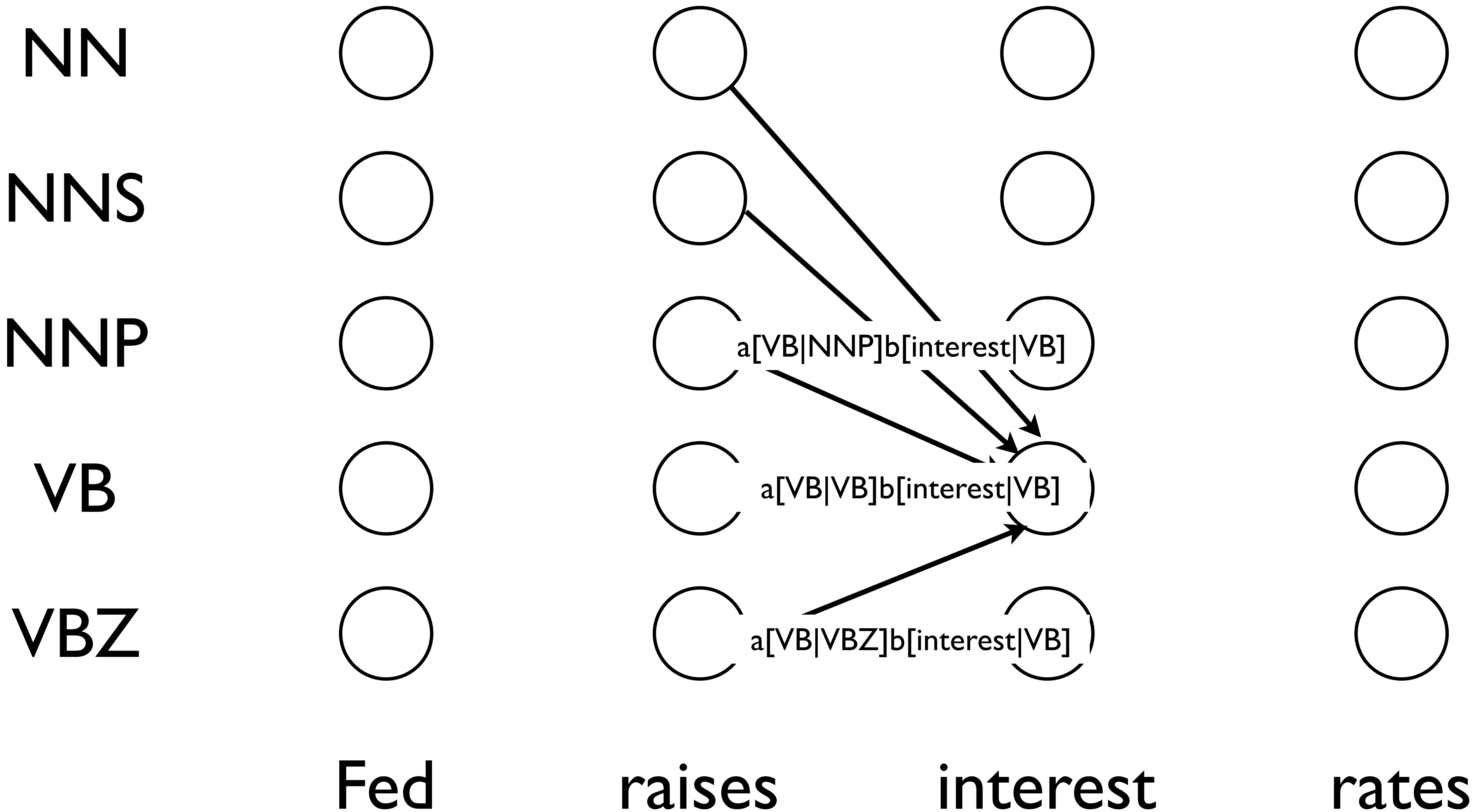
Viterbi Algorithm (Tagging)



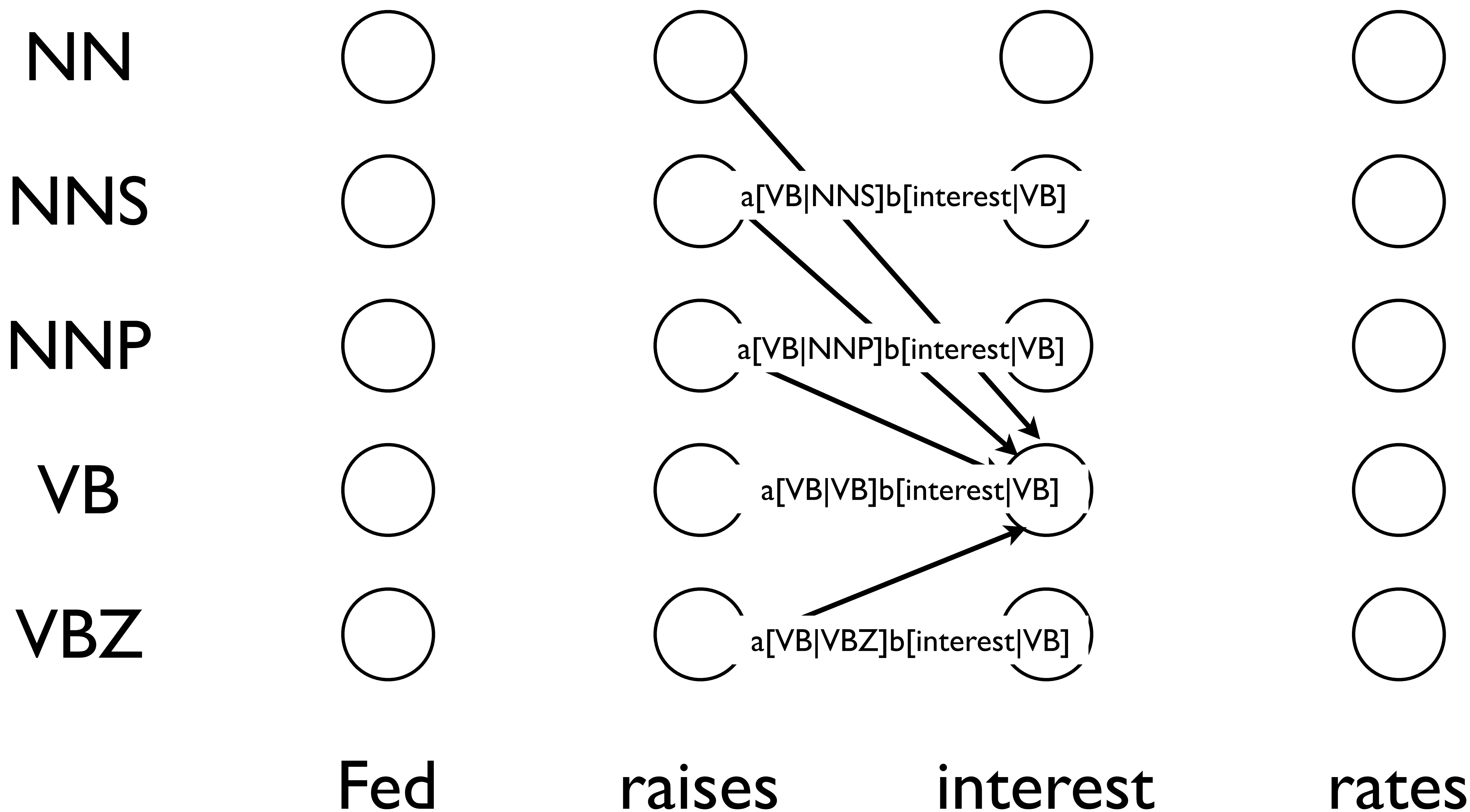
Viterbi Algorithm (Tagging)



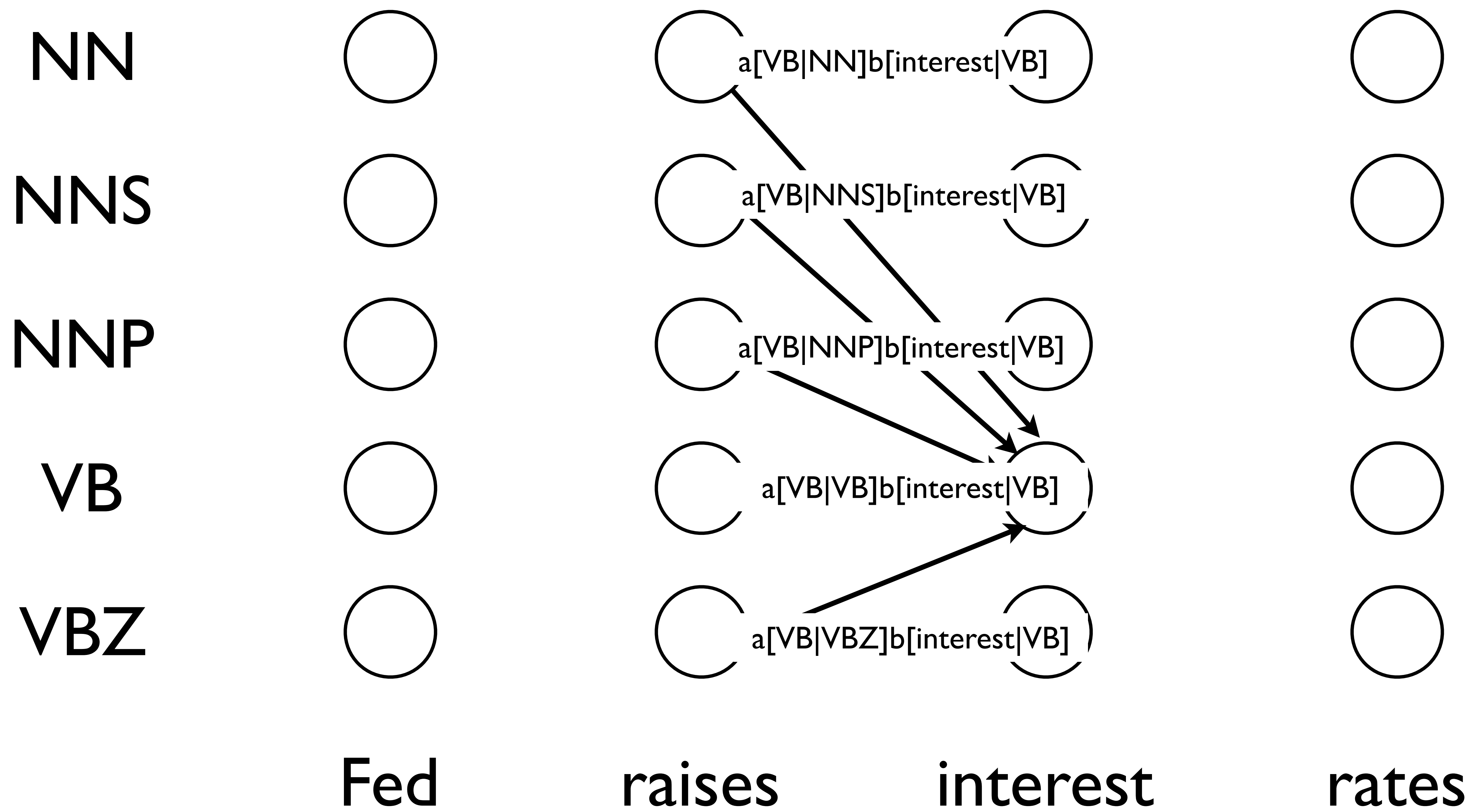
Viterbi Algorithm (Tagging)



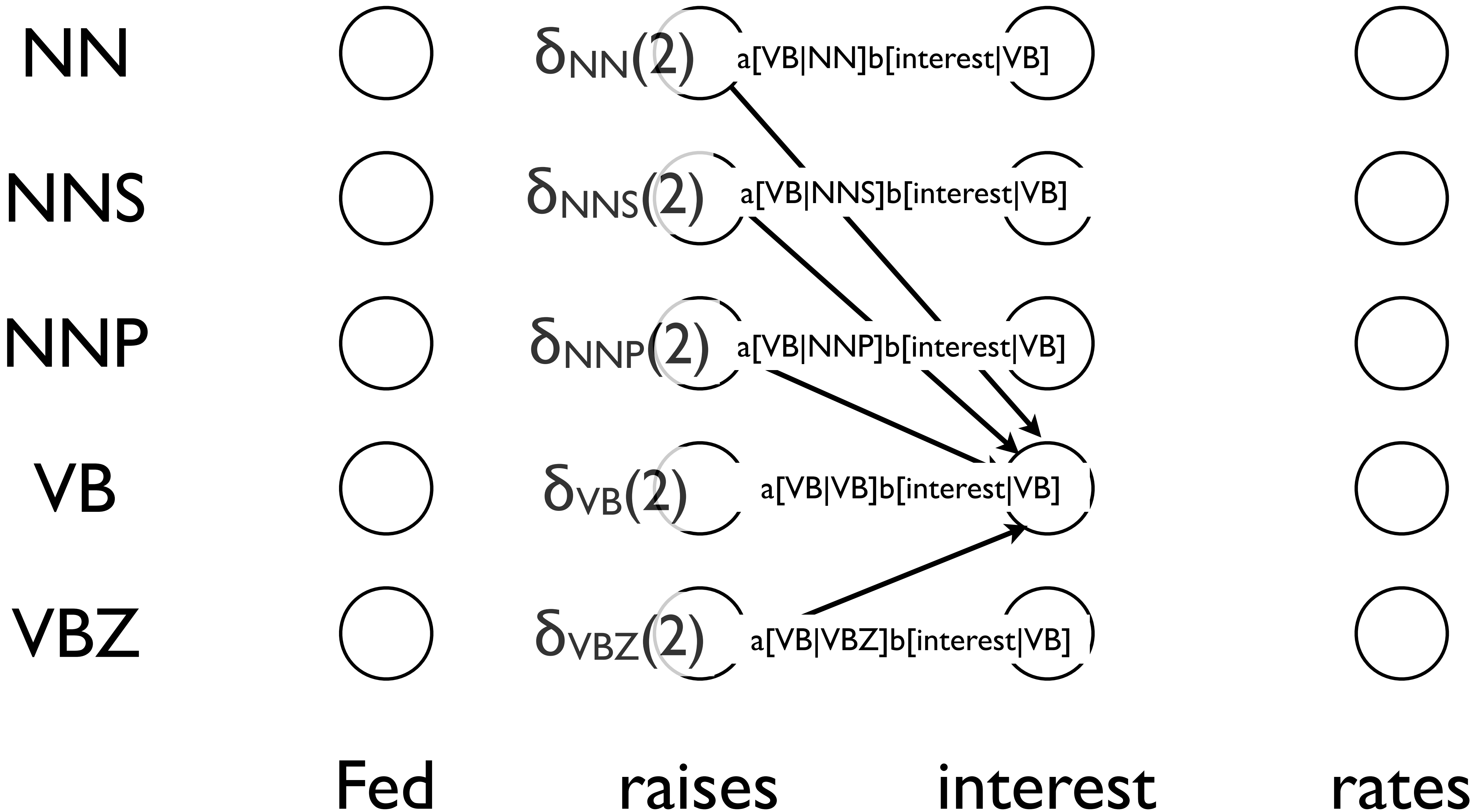
Viterbi Algorithm (Tagging)



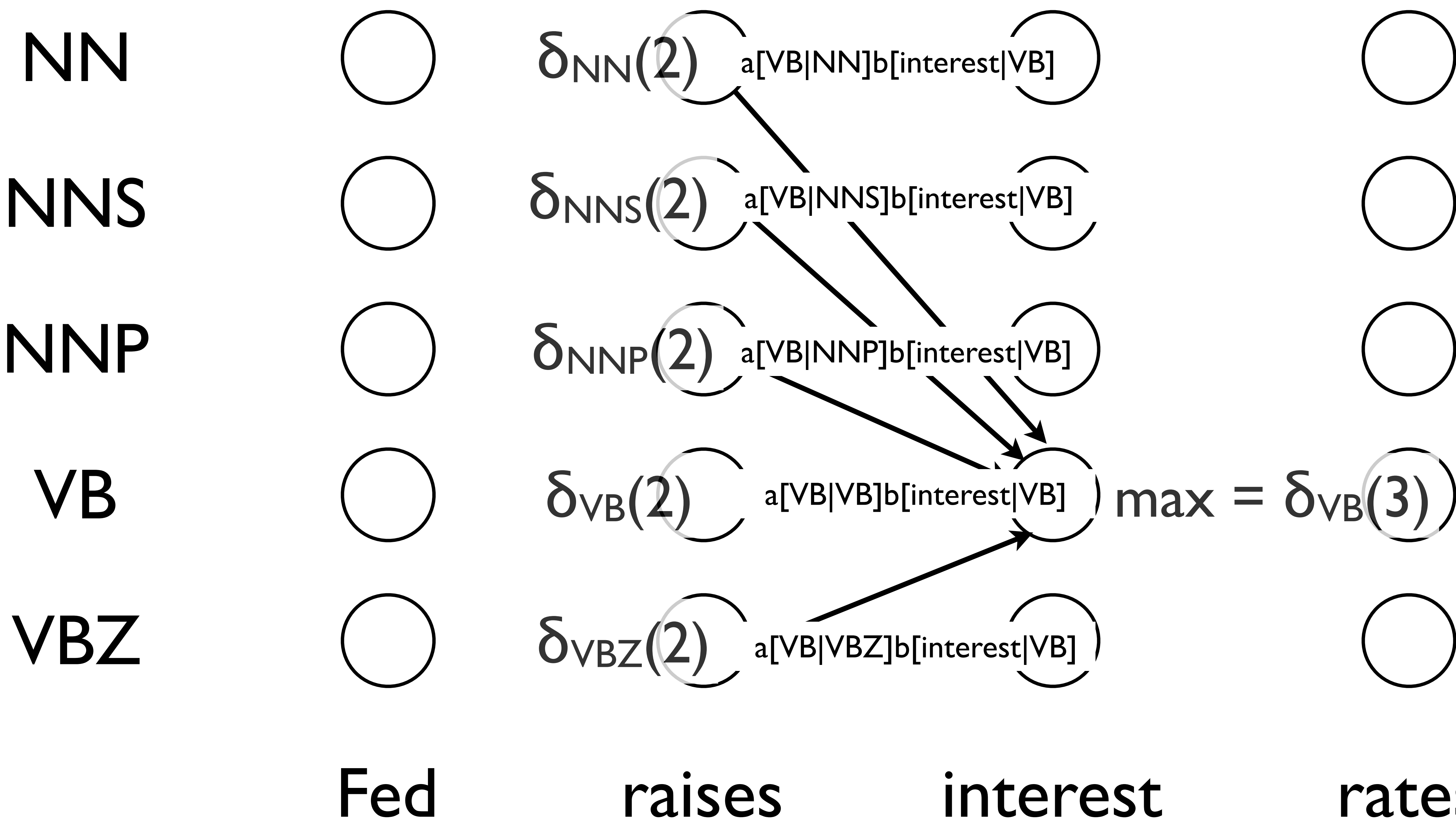
Viterbi Algorithm (Tagging)



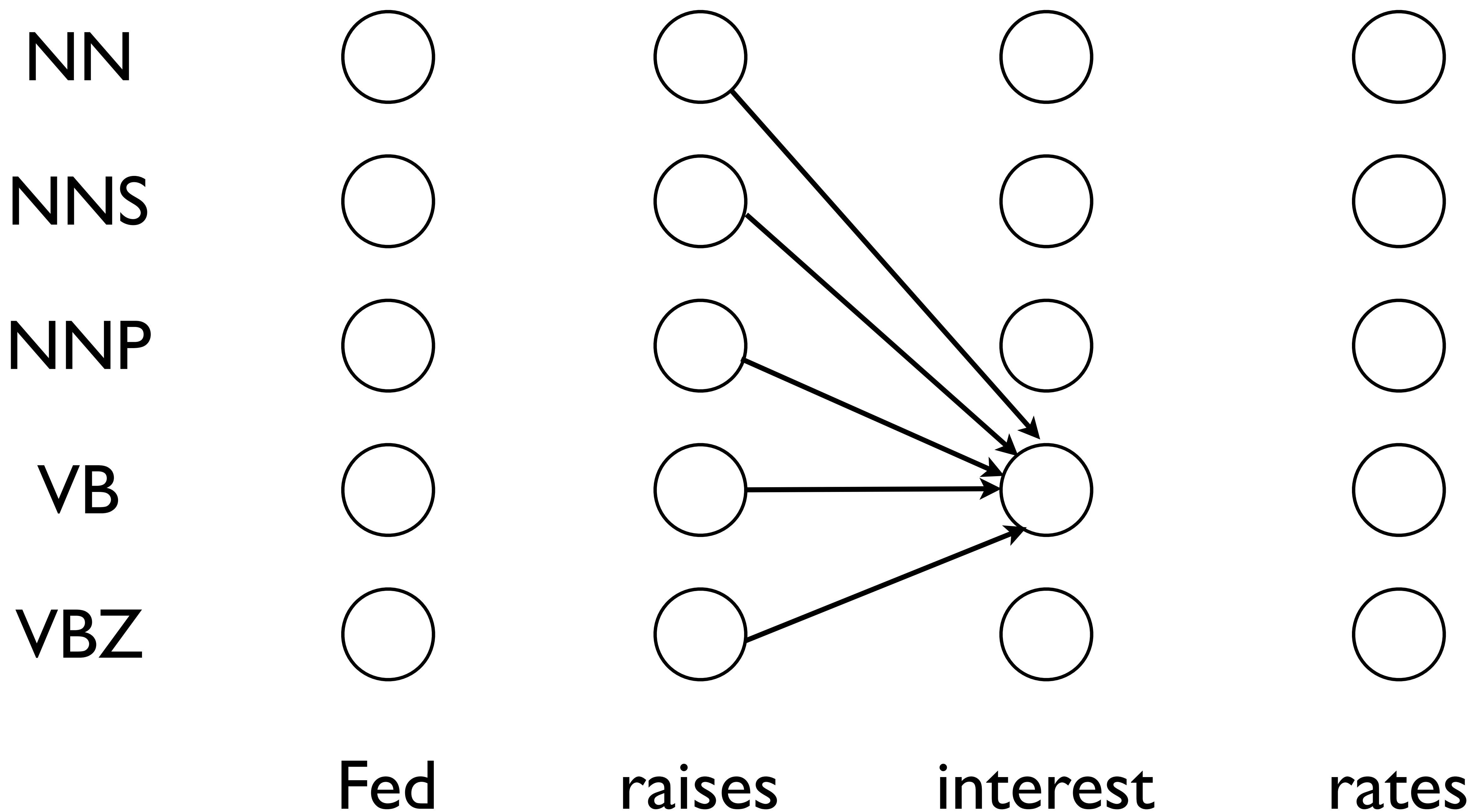
Viterbi Algorithm (Tagging)



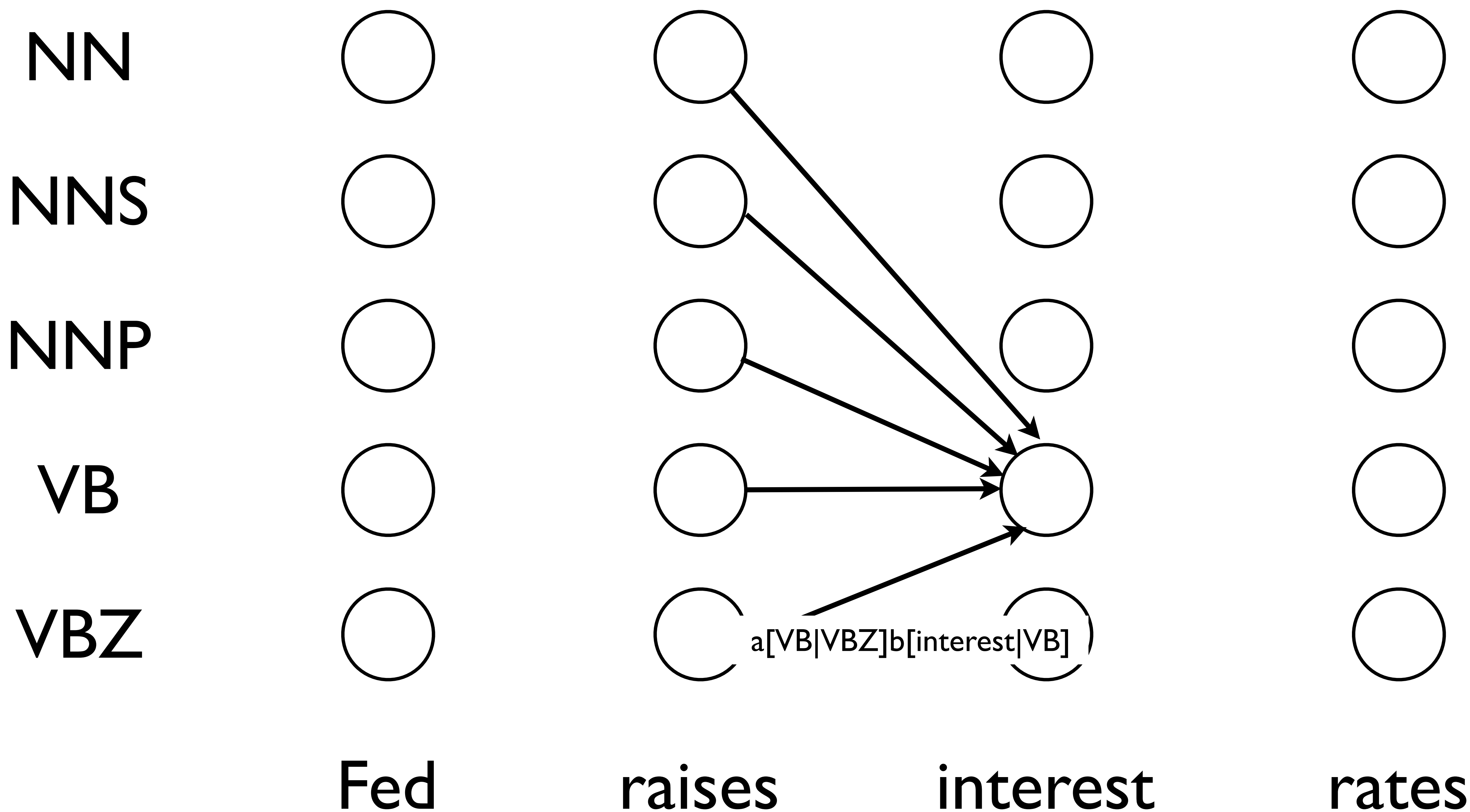
Viterbi Algorithm (Tagging)



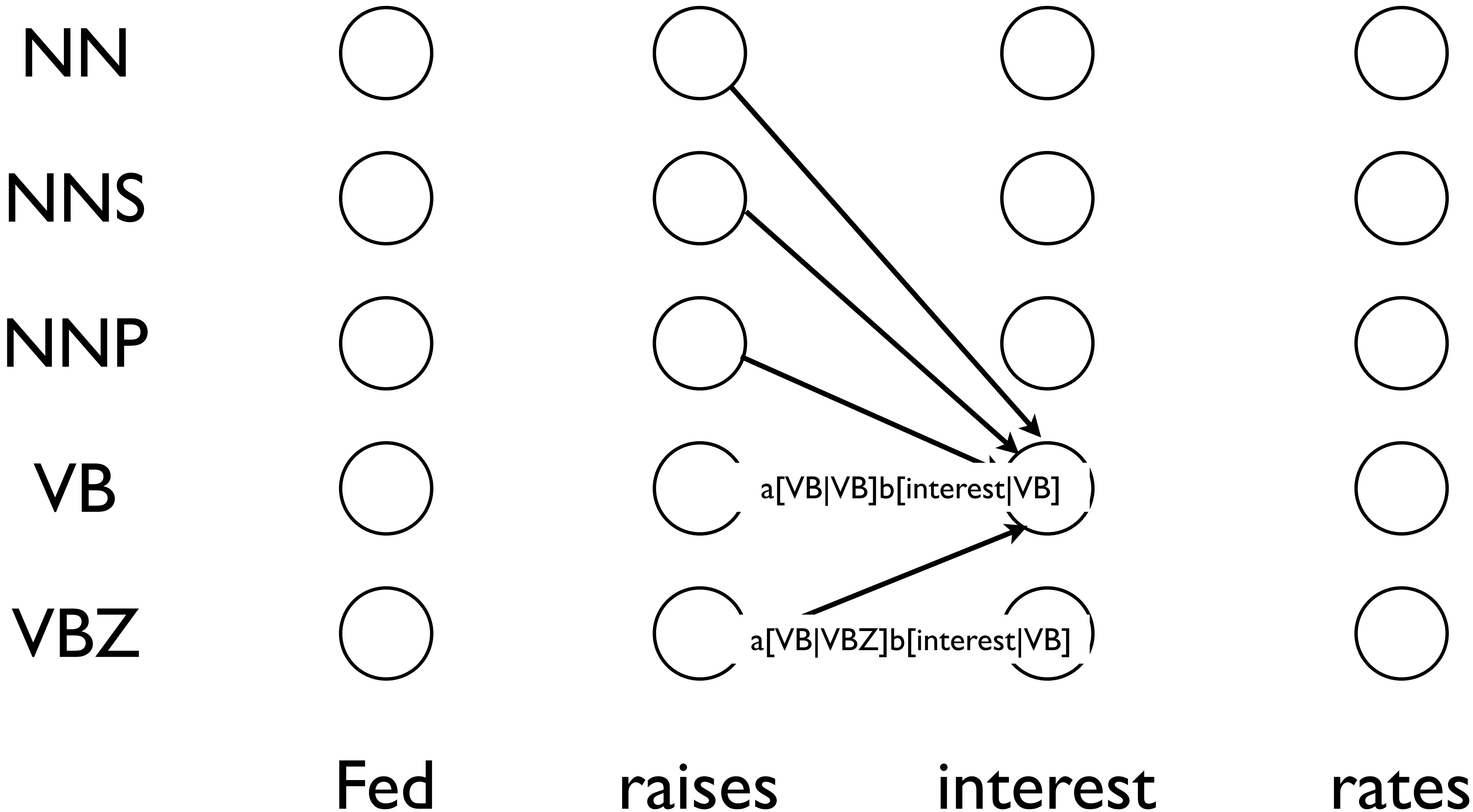
Forward Algorithm (LM)



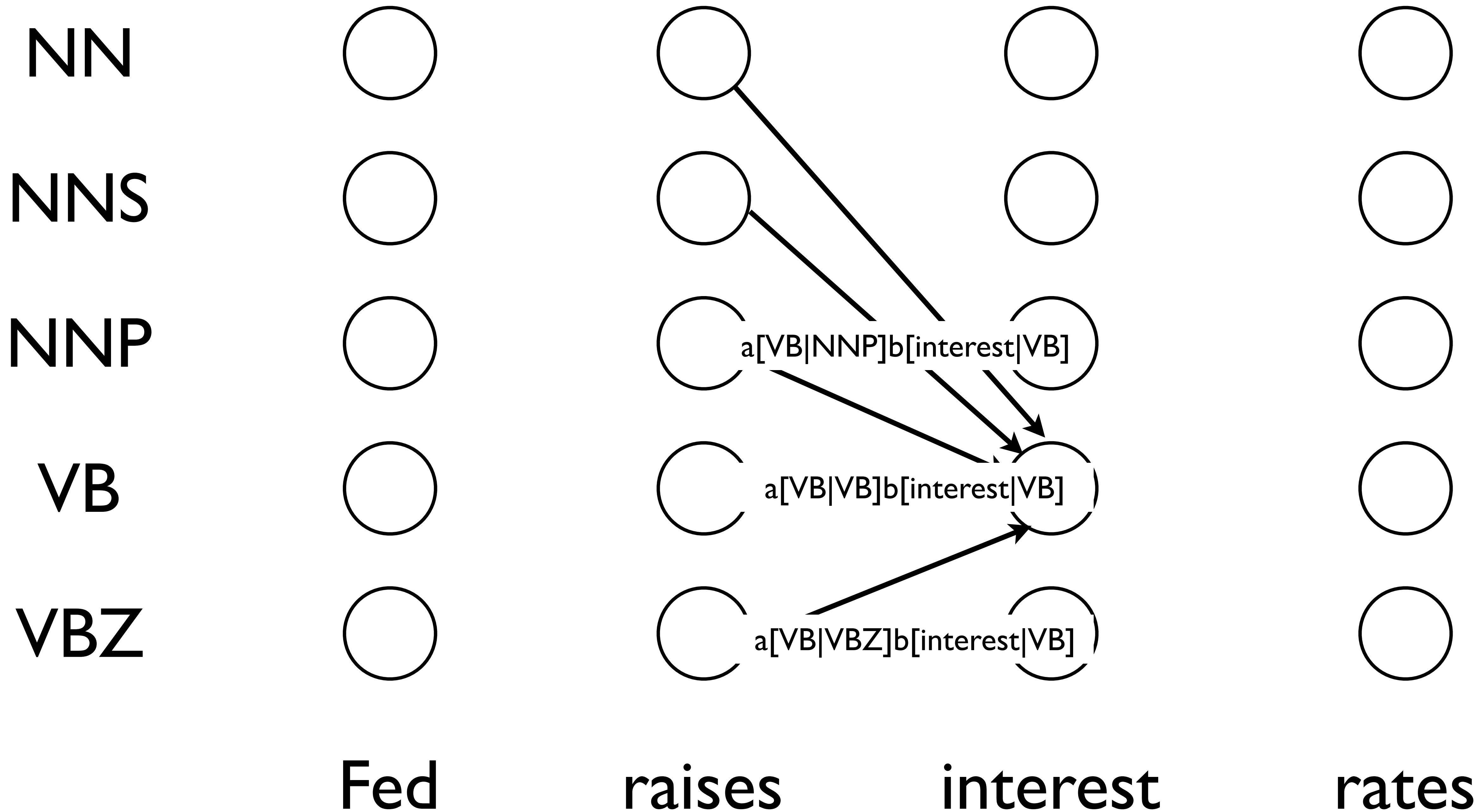
Forward Algorithm (LM)



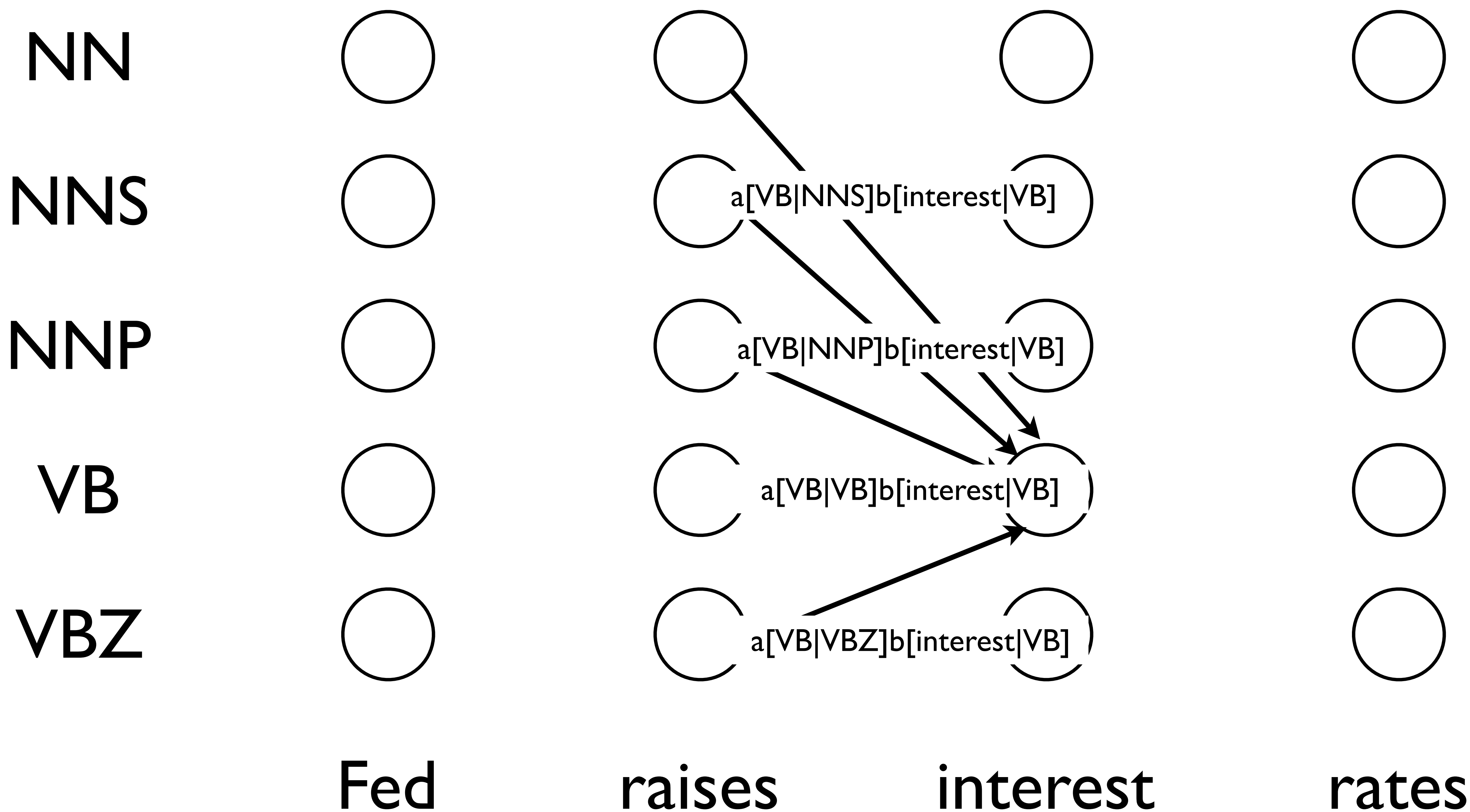
Forward Algorithm (LM)



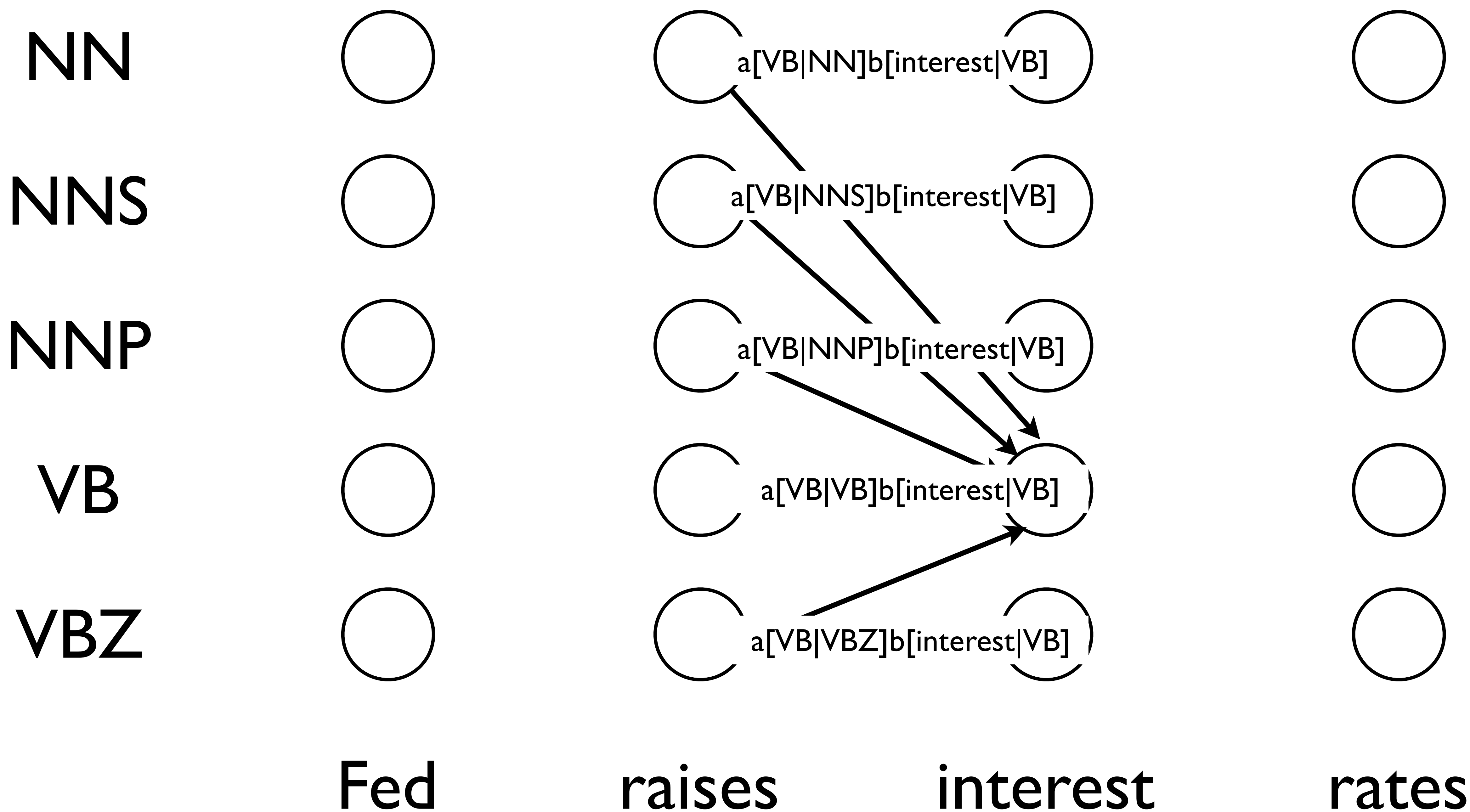
Forward Algorithm (LM)



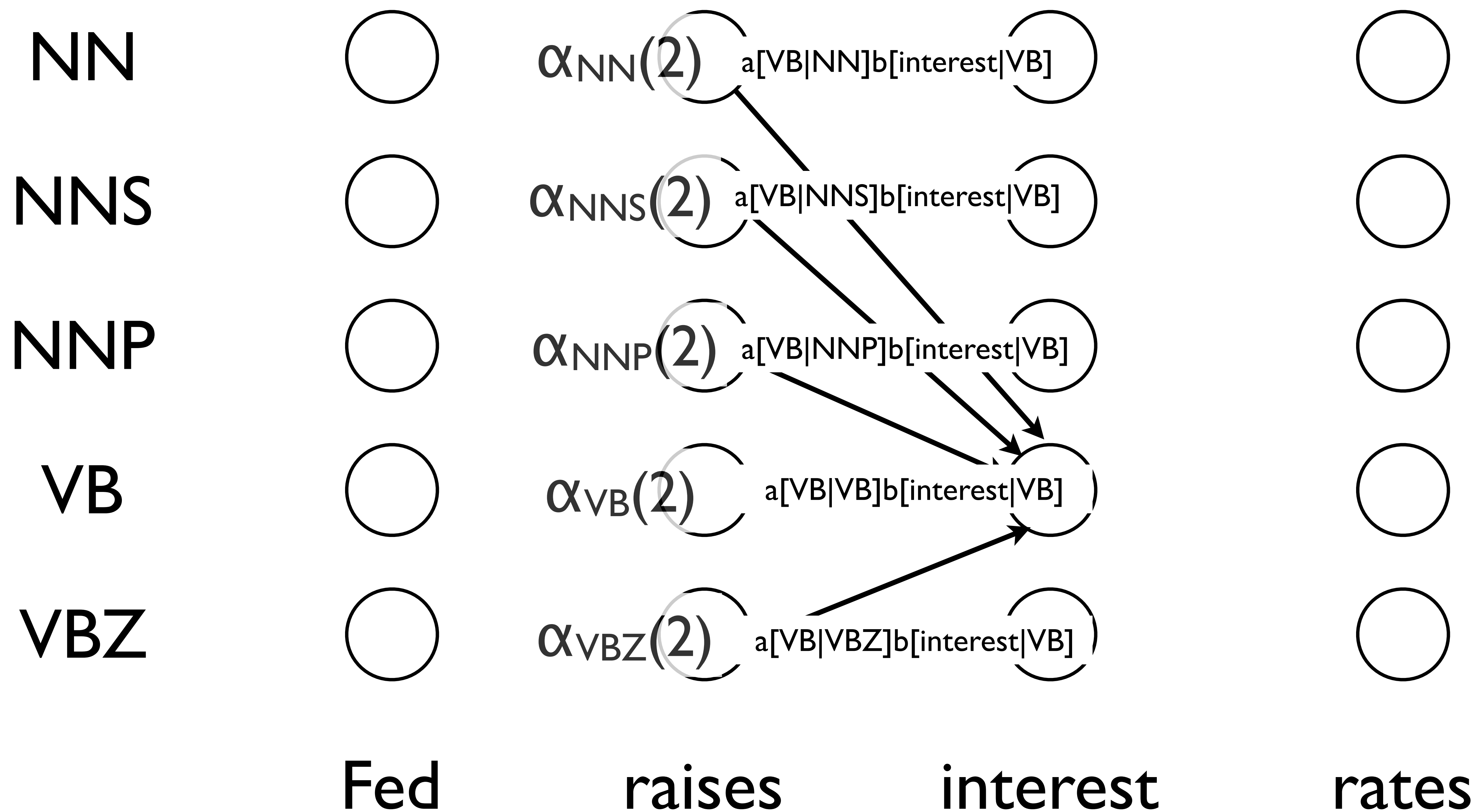
Forward Algorithm (LM)



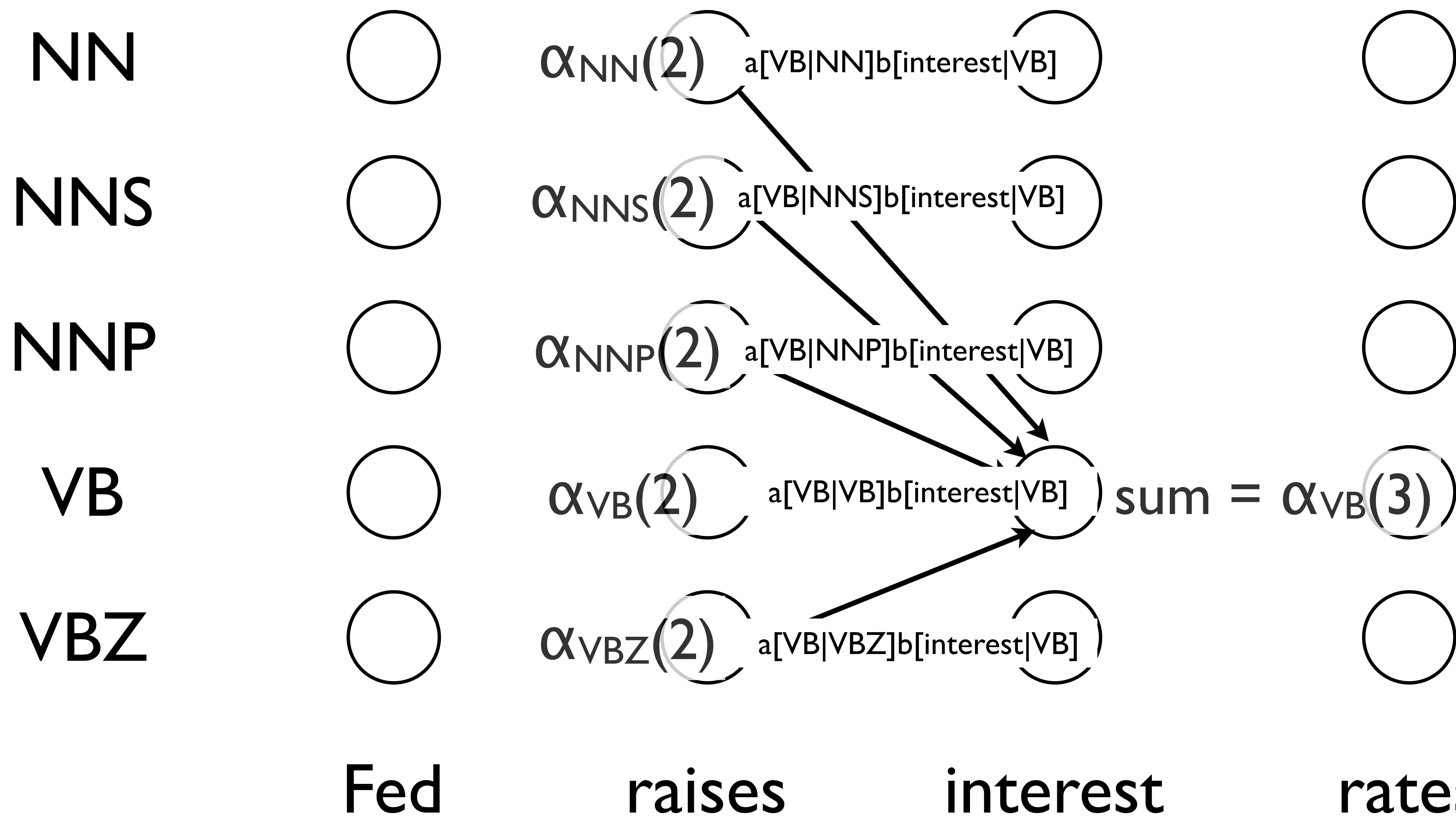
Forward Algorithm (LM)



Forward Algorithm (LM)



Forward Algorithm (LM)



What Do These Greek Letters Mean?

$$\delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)$$

$$\begin{aligned}\alpha_j(t) &= \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \\ &= P(o_1 \cdots o_{t-1}, x_t = j \mid \mu)\end{aligned}$$

What Do These Greek Letters Mean?

Probability of the best path from the beginning to word *t* such that word *t* has tag *j*

$$\delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)$$

$$\begin{aligned} \alpha_j(t) &= \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \\ &= P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) \end{aligned}$$

What Do These Greek Letters Mean?

Probability of the best path from the beginning to word *t* such that word *t* has tag *j*

$$\delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)$$

Probability of all paths from the beginning to word *t* such that word *t* has tag *j*

$$\begin{aligned}\alpha_j(t) &= \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \\ &= P(o_1 \cdots o_{t-1}, x_t = j \mid \mu)\end{aligned}$$

What Do These Greek Letters Mean?

Probability of the best path from the beginning to word *t* such that word *t* has tag *j*

$$\delta_j(t) = \max_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu)$$

Probability of all paths from the beginning to word *t* such that word *t* has tag *j*

$$\begin{aligned} \alpha_j(t) &= \sum_{x_1 \cdots x_{t-1}} P(x_1 \cdots x_{t-1}, o_1 \cdots o_{t-1}, x_t = j \mid \mu) \\ &= P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) \end{aligned}$$

NOT the probability of tag *j* at time *t*

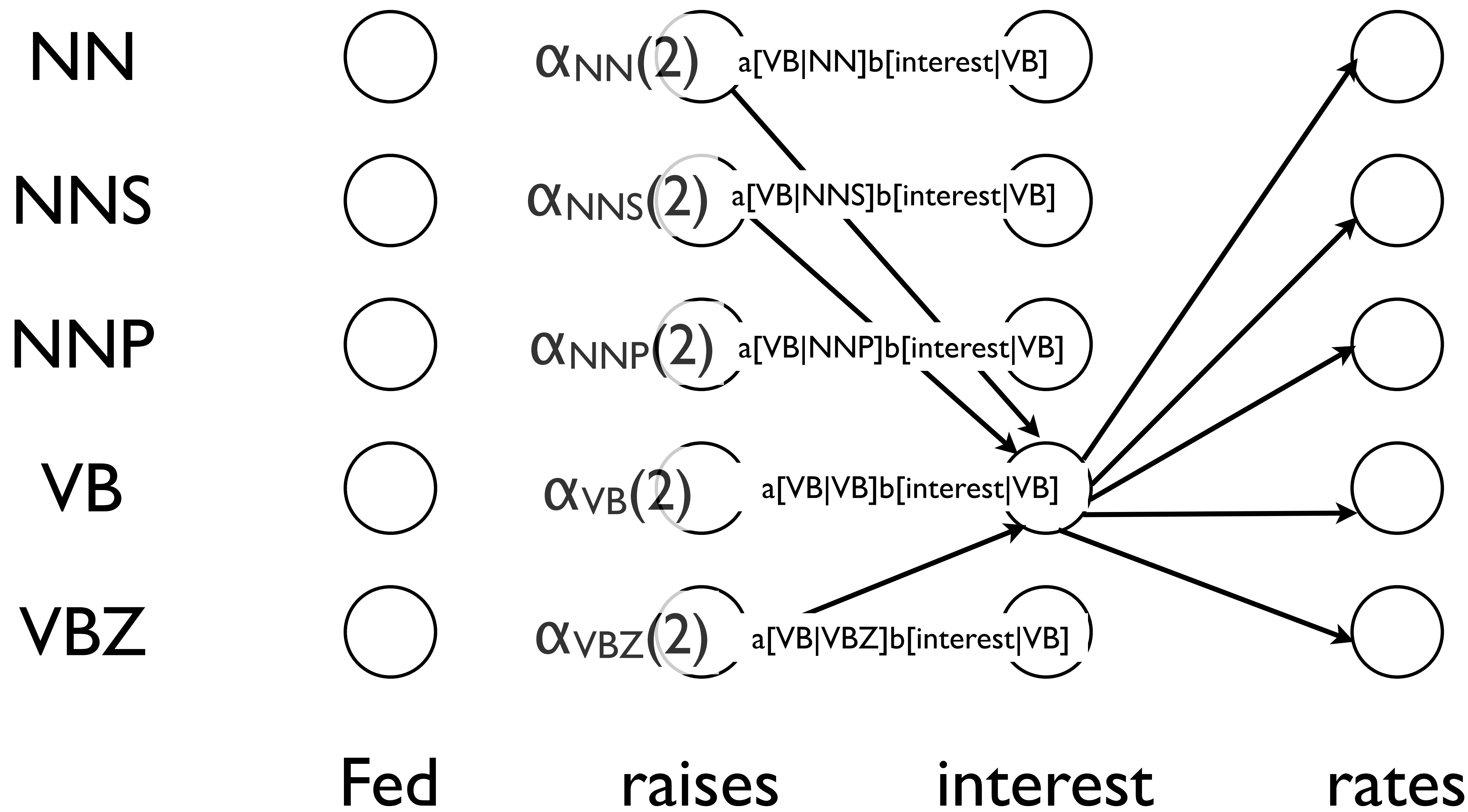
HMM Language Modeling

- Probability of observations, summed over all possible ways of tagging that observation: $$\sum_i \alpha_i(T)$$
- This is the sum of all path probabilities in the trellis

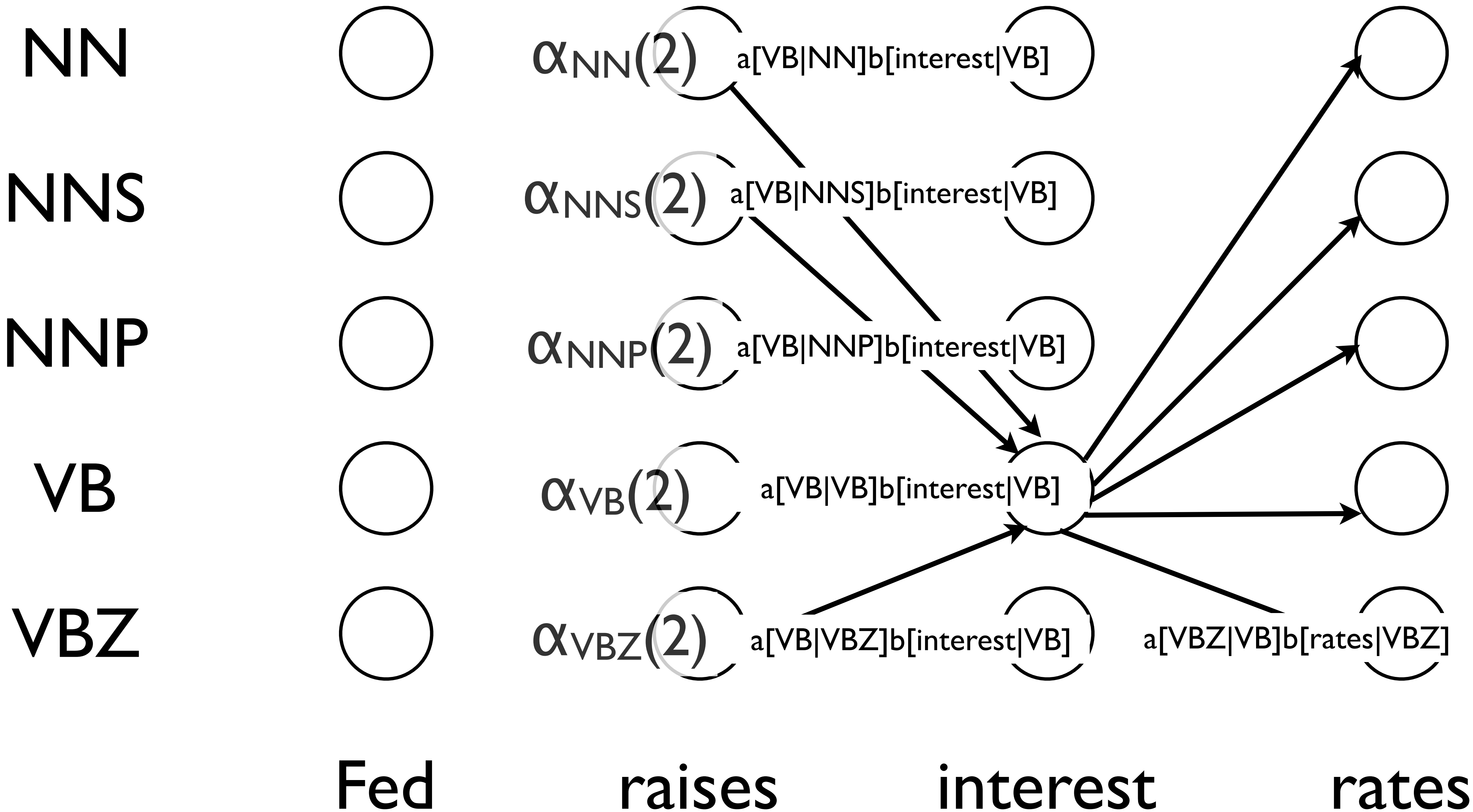
HMM Parameter Estimation

- Supervised
  - Train on tagged text, test on plain text
  - Maximum likelihood (can be smoothed):
    - a[VBZ | NN] = C(NN,VBZ) / C(NN)
    - b[rates |VBZ] = C(VBZ,rates) / C(VBZ)
- Unsupervised
  - Train and test on plain text
  - What can we do?

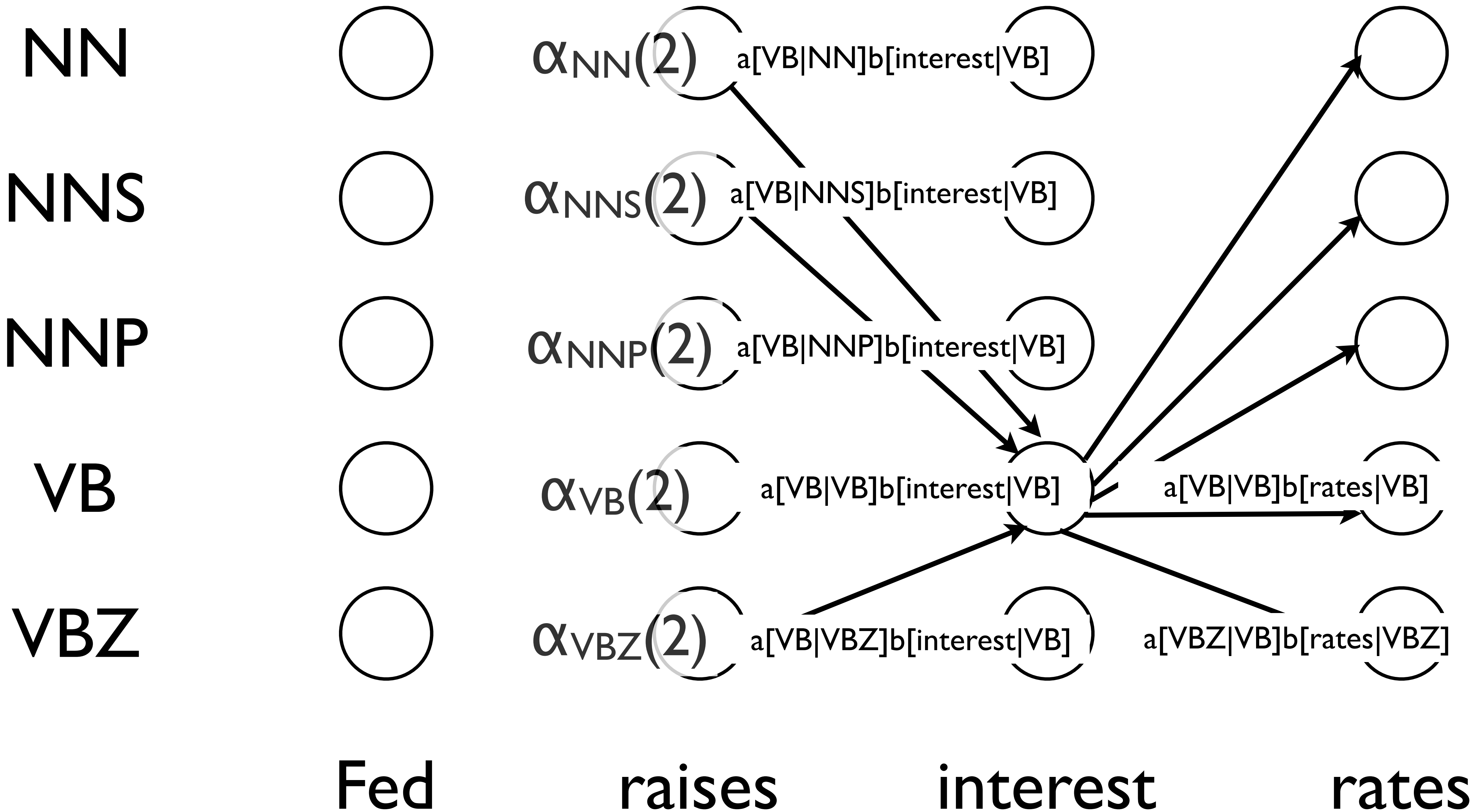
Forward-Backward Algorithm



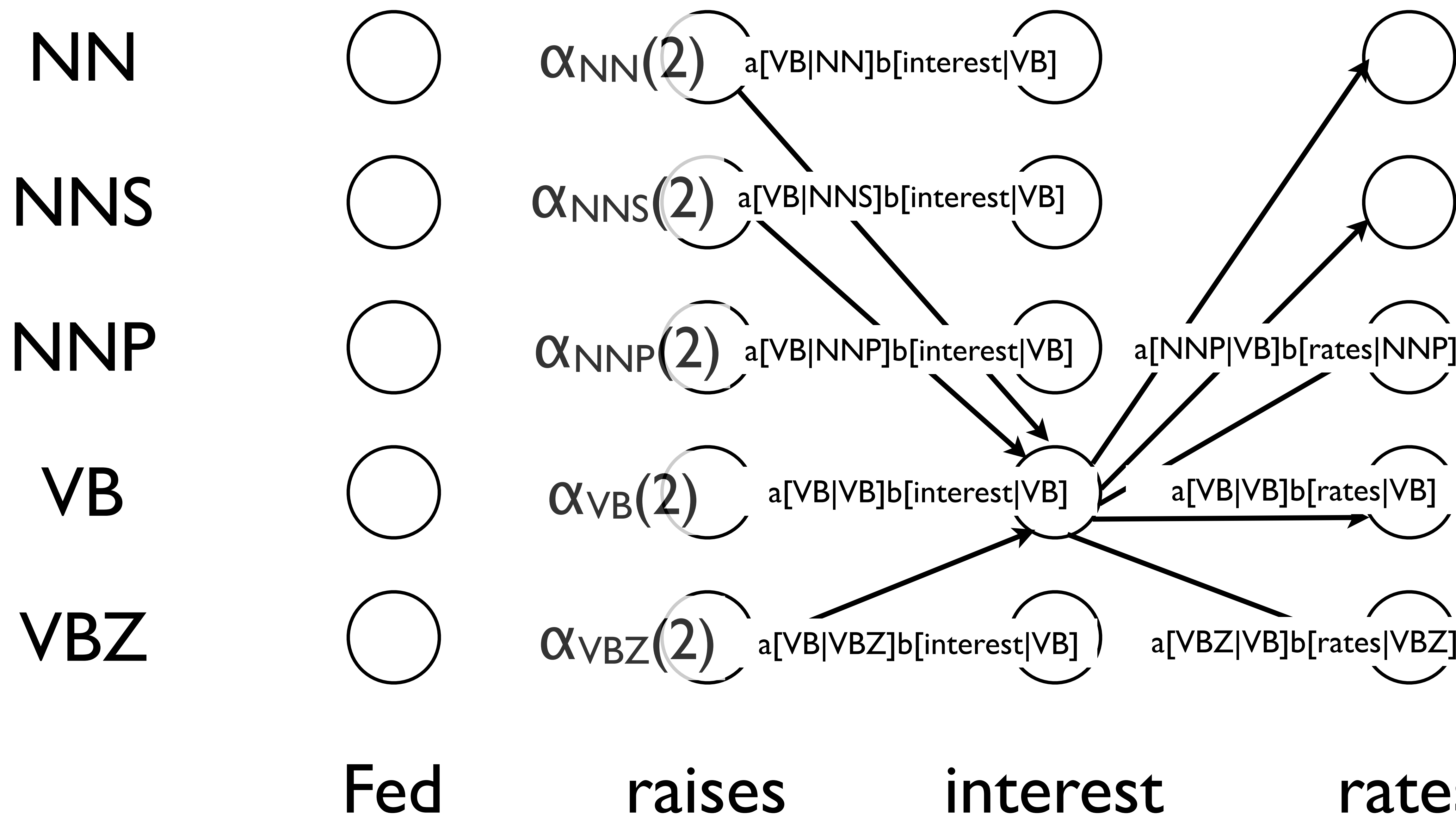
Forward-Backward Algorithm



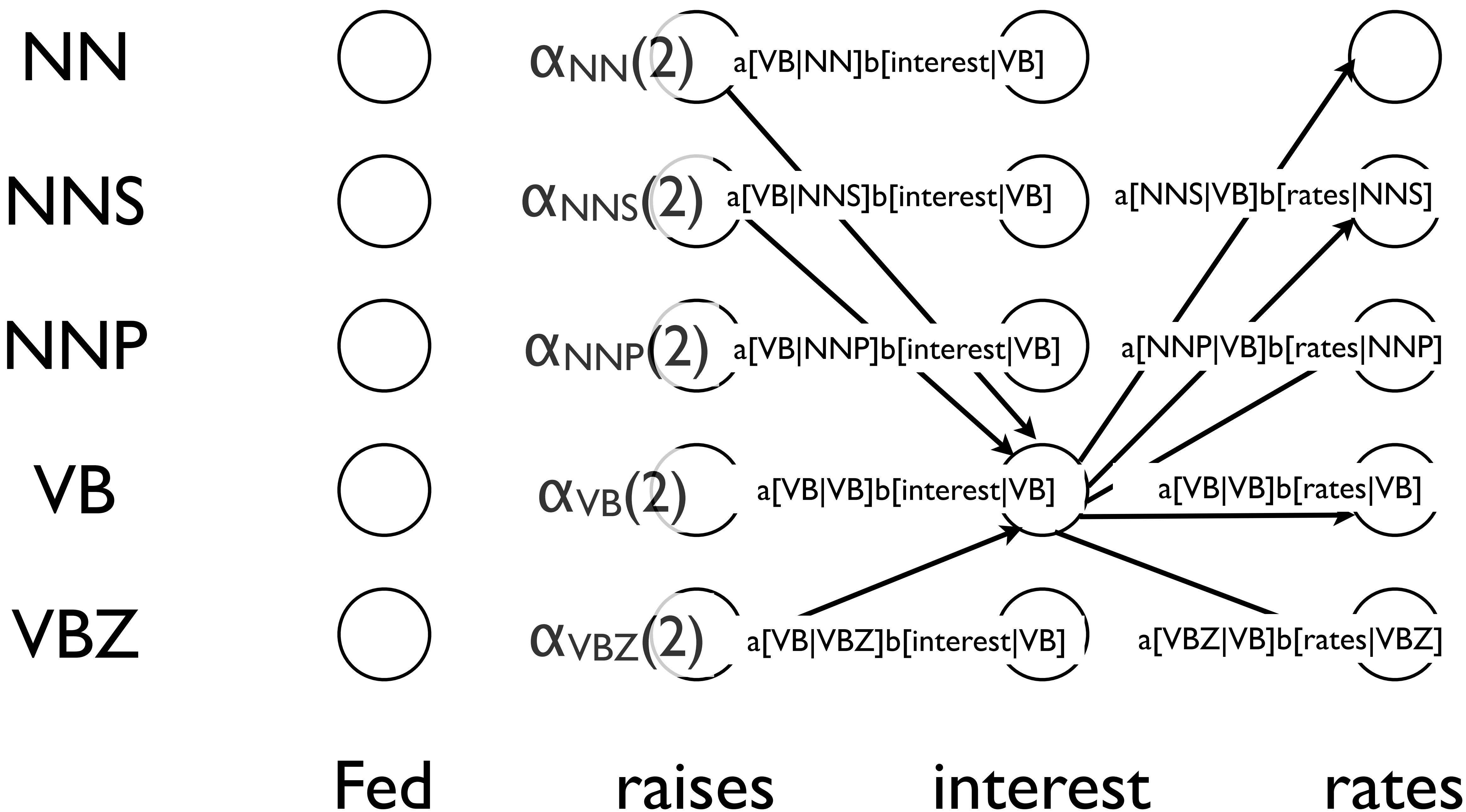
Forward-Backward Algorithm



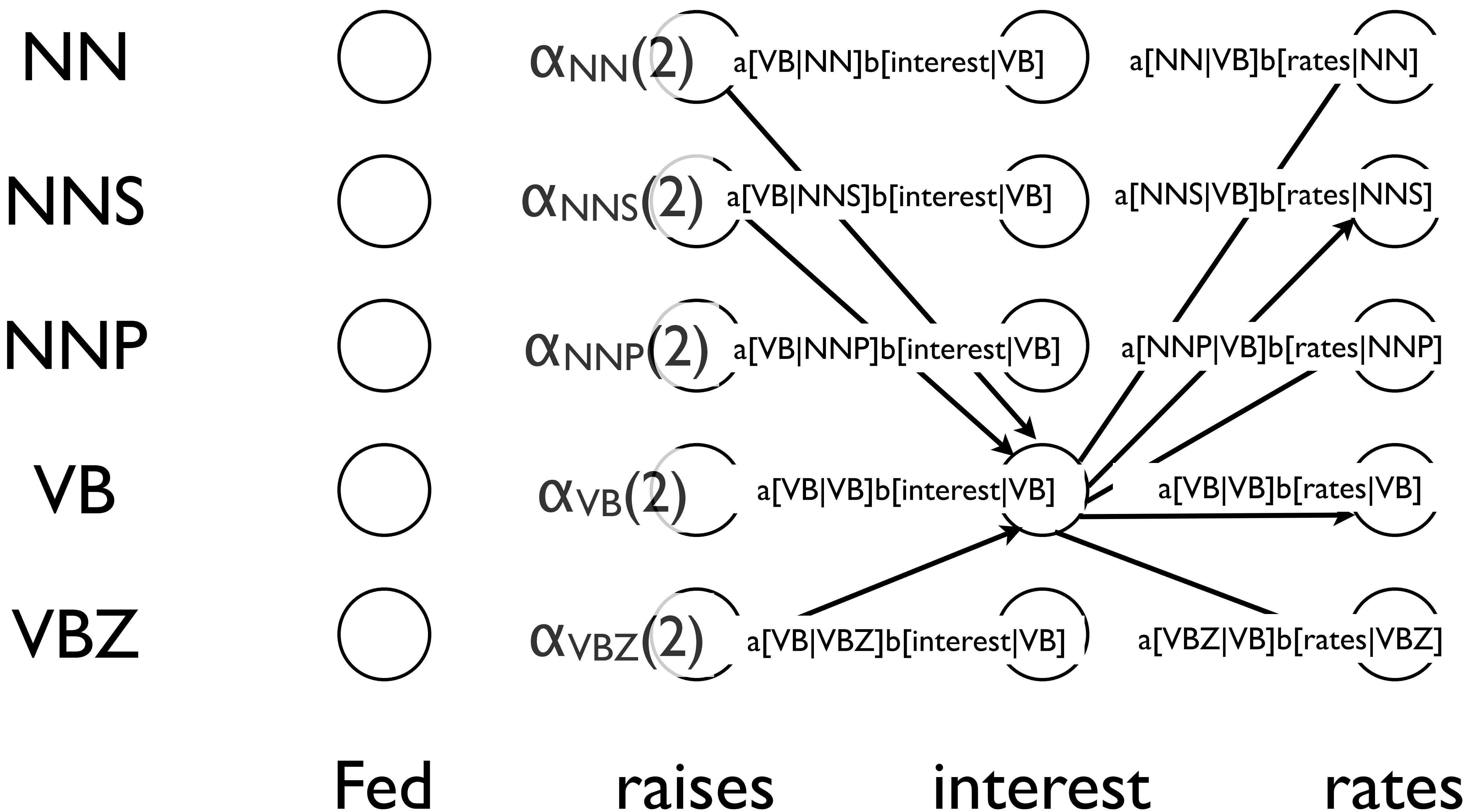
Forward-Backward Algorithm



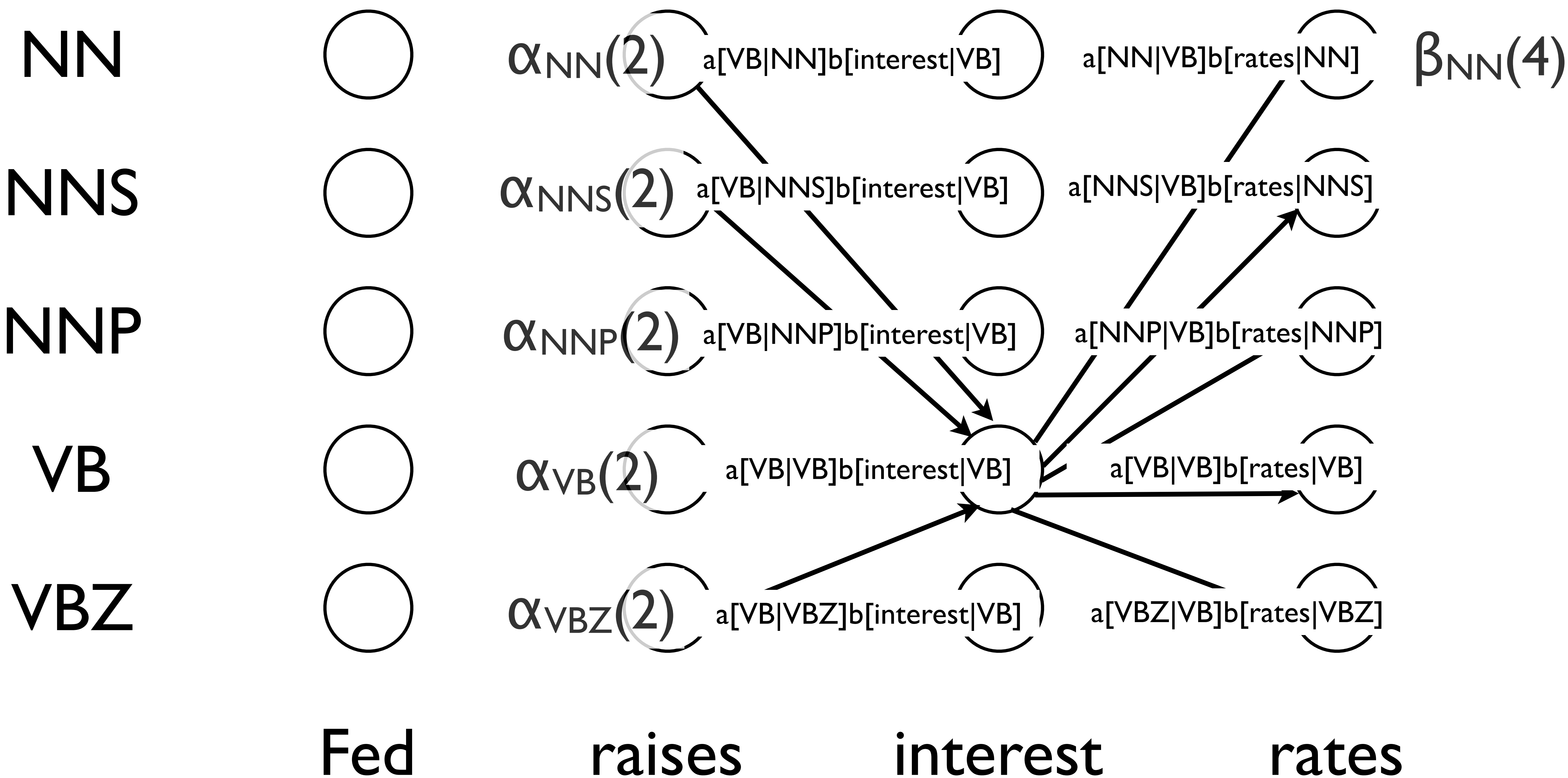
Forward-Backward Algorithm



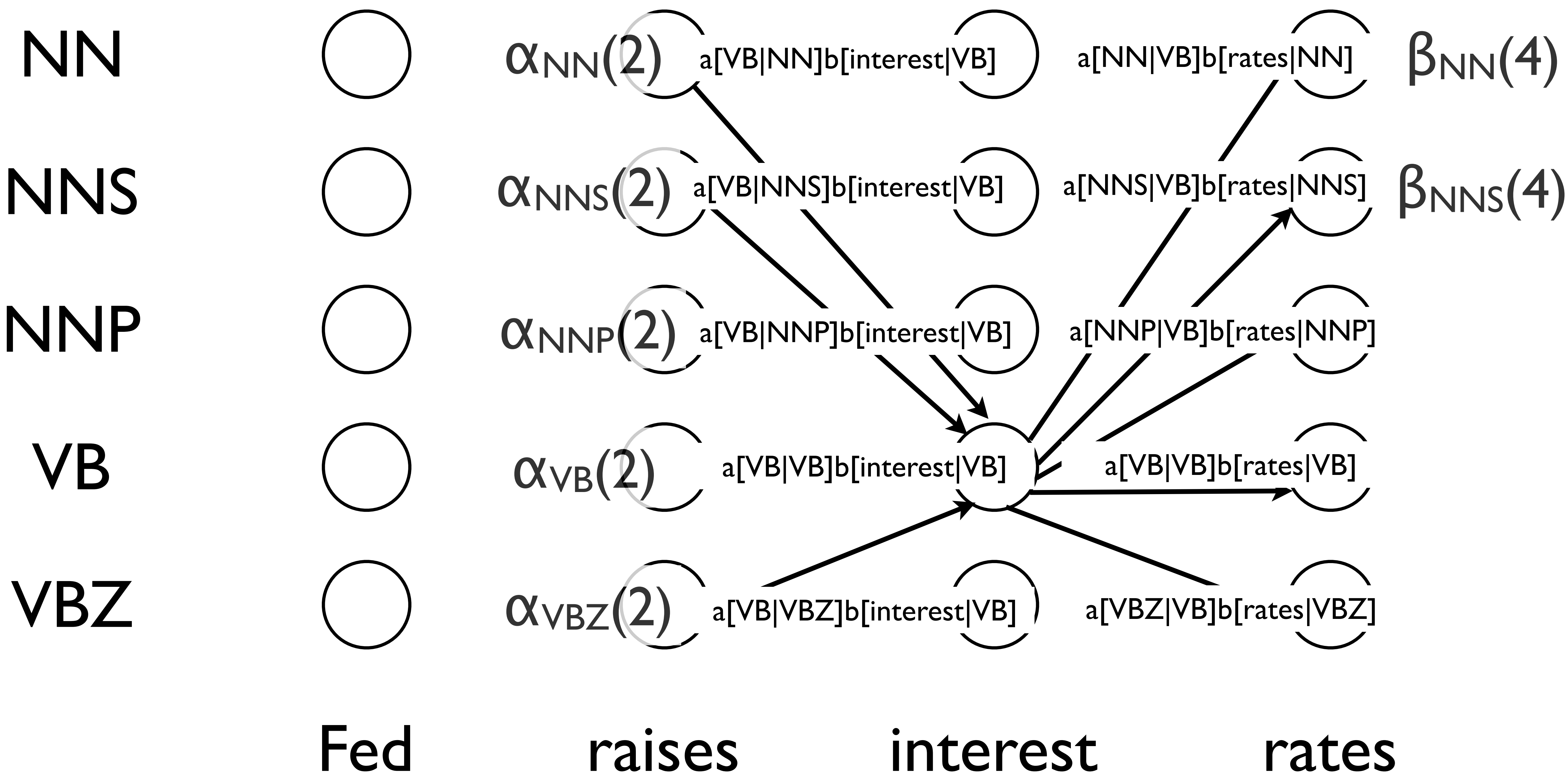
Forward-Backward Algorithm



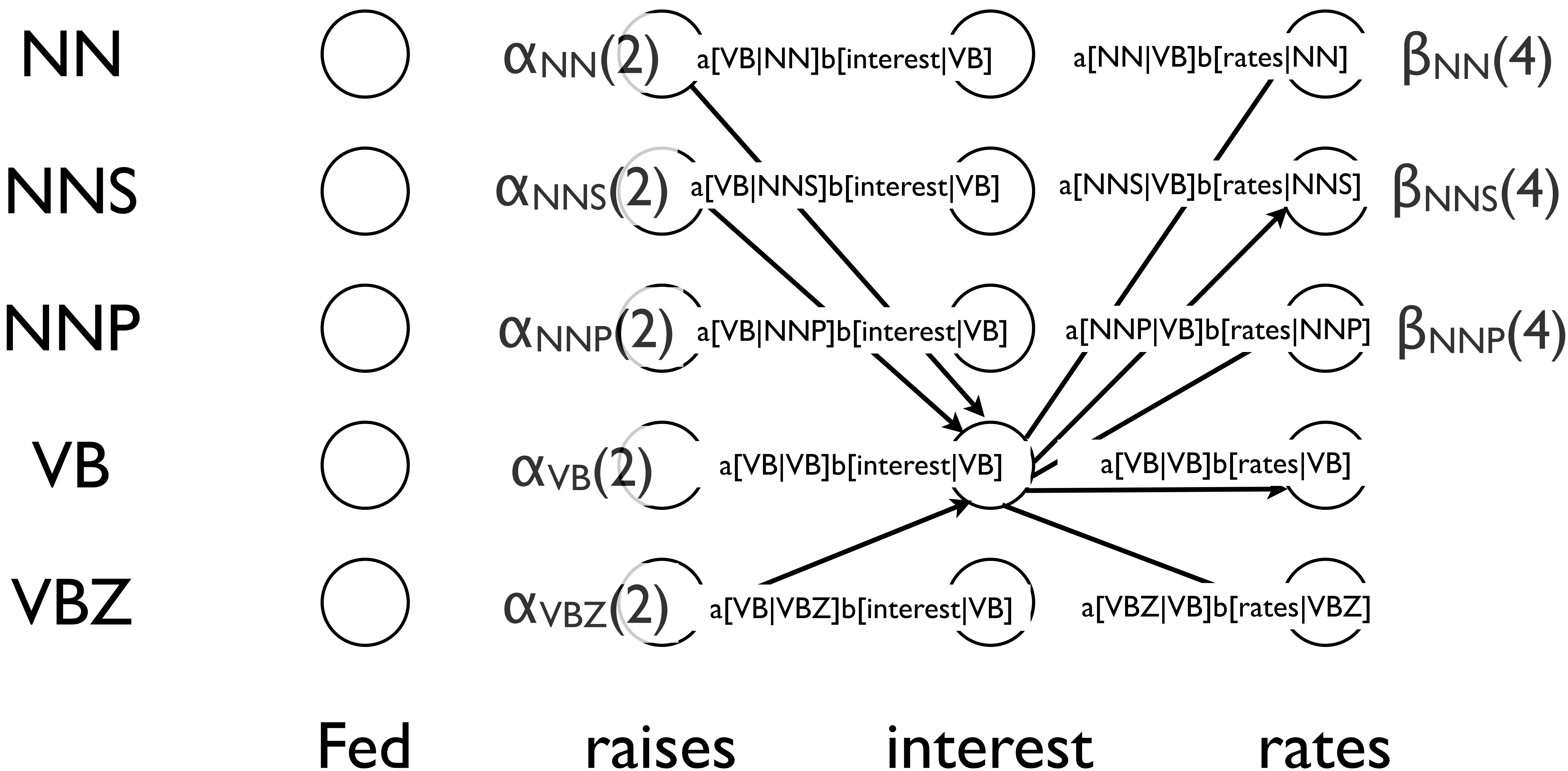
Forward-Backward Algorithm



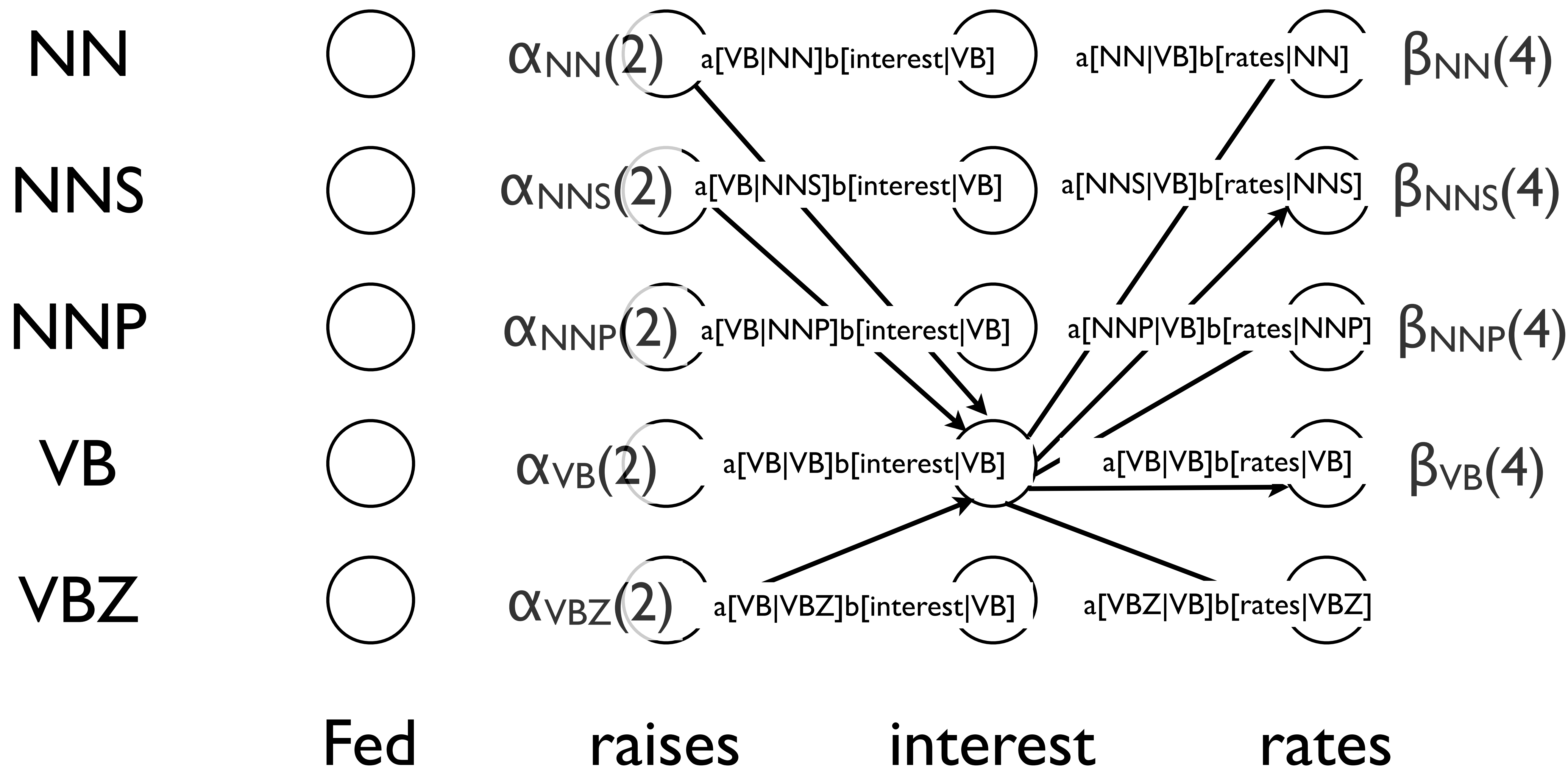
Forward-Backward Algorithm



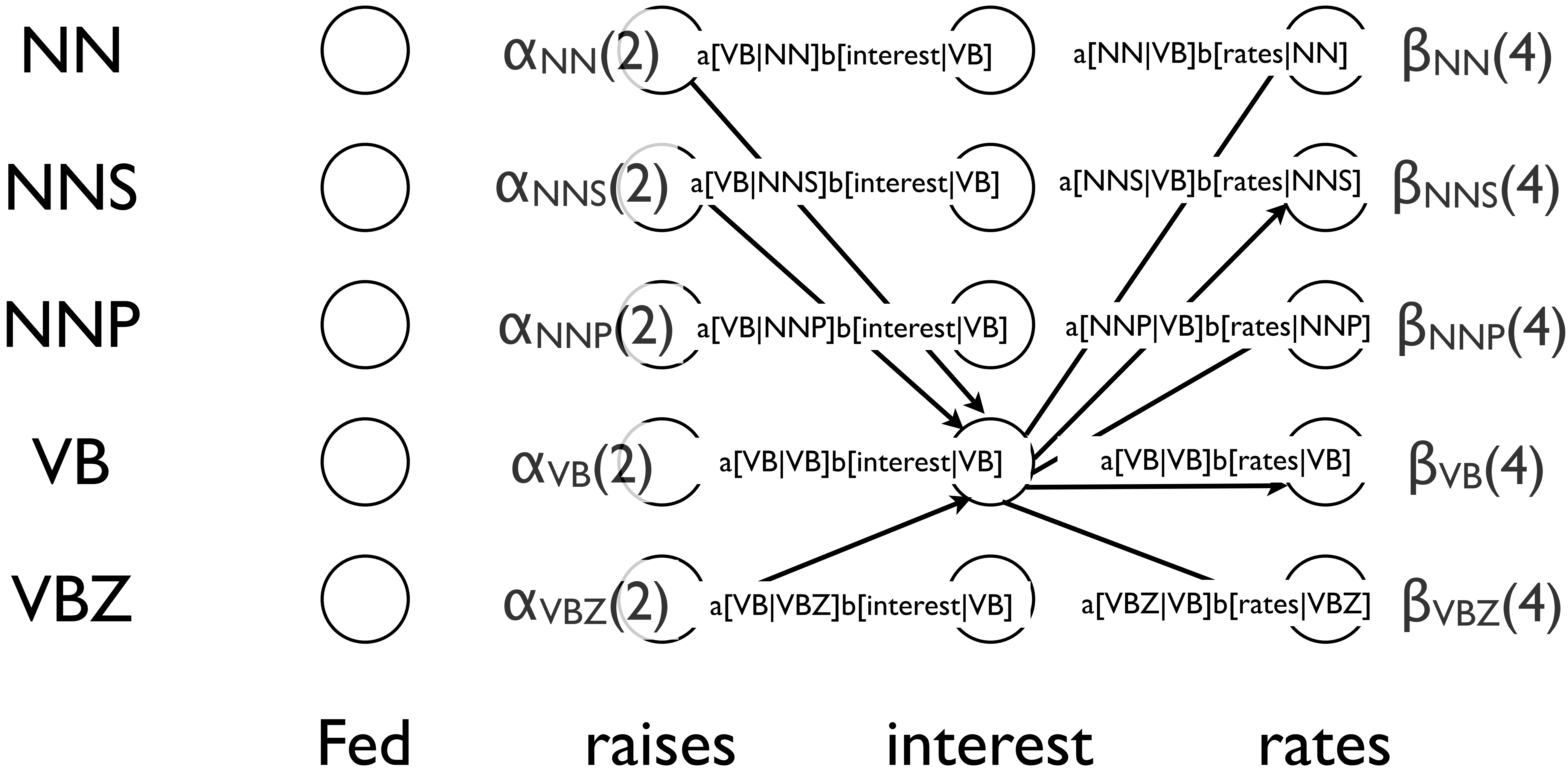
Forward-Backward Algorithm



Forward-Backward Algorithm



Forward-Backward Algorithm



Forward-Backward Algorithm

$$P(o_1 \cdots o_{t-1}, x_t = j \mid \mu) = \alpha_j(t)$$

$$P(o_t \cdots o_T \mid x_t = j, \mu) = \beta_j(t)$$

$$P(o_1 \cdots o_T, x_t = j \mid \mu) = \alpha_j(t)\beta_j(t)$$

$$P(x_t = j \mid O, \mu) = \frac{P(x_t = j, O \mid \mu)}{P(O \mid \mu)} = \frac{\alpha_j(t)\beta_j(t)}{\alpha_\#(T)}$$

$$\begin{aligned} P(x_t = i, x_{t+1} = j \mid O, \mu) &= \frac{P(x_t = i, x_{t+1} = j, O \mid \mu)}{P(O \mid \mu)} \\ &= \frac{\alpha_i(t) a[j \mid i] b[o_t \mid j] \beta_j(t+1)}{\alpha_\#(T)} \end{aligned}$$

Expectation Maximization (EM)

- Iterative algorithm to maximize likelihood of observed data in the absence of hidden data (e.g., tags)
- Choose an initial model μ
- **Expectation step**: find the expected value of hidden variables given current μ
- **Maximization step**: choose new μ to maximize probability of hidden and observed data
- Guaranteed to increase likelihood
- Not guaranteed to find global maximum

Supervised vs. Unsupervised

Supervised	Unsupervised
Annotated training text	Plain text
Simple count/normalize	EM
Fixed tag set	Set during training
Training reads data once	Training needs multiple passes

Logarithms for Precision

$$P(Y) = p(y_1)p(y_2)\cdots p(y_T)$$

$$\log P(Y) = \log p(y_1) + \log p(y_2) \cdots + \log p(y_T)$$

Increased dynamic range of [0,1] to [-∞,0]

Semirings

	Set	$\oplus$	$\otimes$	0	1
Prob	R^+	+	x	0	1
Max	R^+	max	x	0	1
Log	$R \cup \{\pm\infty\}$	log+	+	$-\infty$	0
"Tropical"	$R \cup \{\pm\infty\}$	max	+	$-\infty$	0
Shortest path	$R \cup \{\pm\infty\}$	min	+	∞	0
Boolean	{F,T}	$\vee$	$\wedge$	F	T
String	$\Sigma^* \cup \{\infty\}$	longest common prefix	concat	∞	ε

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Inference rule

$$\forall A, B \in T; W \in V; 0 \leq i, j \leq n$$

$$\begin{aligned} path(B, j) \impliedby\ & path(A, i) \wedge word(W, i, j) \\ & \wedge\ emit(B, W) \wedge trans(A, B) \end{aligned}$$

In Prolog

```
path(B,J) :-
  path(A,I), word(W,I,J), emit(B,W), trans(A,B).
path("Start",0).
word("the",0,1).
word("cool",1,2).
…
emit("DT","the").
…
```

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Inference rule

$$\forall B, j : path(B, j) = \bigvee_{A,W,i} path(A, i) \wedge word(W, i, j) \wedge emit(B, W) \wedge trans(A, B)$$

In Prolog

```
path(B,J) :-
  path(A,I), word(W,I,J), emit(B,W), trans(A,B).
path("Start",0).
word("the",0,1).
word("cool",1,2).
…
emit("DT","the").
…
```

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Inference rule

$$\forall B, j : path(B, j) = \bigvee_{A,W,i} path(A, i) \wedge word(W, i, j) \wedge emit(B, W) \wedge trans(A, B)$$

Shortest path

$$\forall B, j : path(B, j) = \min_{A,W,i} path(A, i) + word(W, i, j) + emit(B, W) + trans(A, B)$$

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Shortest path

$$\forall B, j : path(B, j) = \min_{A,W,i} path(A, i) + word(W, i, j) + emit(B, W) + trans(A, B)$$

Viterbi algorithm

$$\forall B, j : path(B, j) = \max_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Viterbi algorithm

$$\forall B, j : path(B, j) = \max_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Viterbi w/log probabilities

$$\forall B, j : path(B, j) = \max_{A,W,i} path(A, i) + word(W, i, j) + emit(B, W) + trans(A, B)$$

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Viterbi algorithm

$$\forall B, j : path(B, j) = \max_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Forward algorithm

$$\forall B, j : path(B, j) = \sum_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Forward algorithm

$$\forall B, j : path(B, j) = \sum_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Let θ = subset of axioms whose weights we wish to optimize

$$goal = \sum_{B} path(B, n)$$

Chain rule

$$\frac{\partial goal}{\partial \theta} = \sum_{B} \frac{\partial goal}{\partial path(B, n)} \frac{\partial path(B, n)}{\partial \theta}$$

Search as Deduction

Axioms $path(\texttt{Start}, 0), word(\texttt{the}, 0, 1), emit(\texttt{DT}, \texttt{the}), \ldots$

Forward algorithm

$$\forall B, j : path(B, j) = \sum_{A,W,i} path(A, i) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Chain rule

$$\frac{\partial goal}{\partial path(A, i)} = \sum_{B,j} \frac{\partial goal}{\partial path(B, j)} \frac{\partial path(B, j)}{\partial path(A, i)}$$

$$\beta_A(i) = \sum_{B,W,j} \beta_B(j) \cdot word(W, i, j) \cdot emit(B, W) \cdot trans(A, B)$$

Reading

- Barzilay & Lee. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. *HLT-NAACL*, 2004.
  - **http://aclweb.org/anthology//N/N04/N04-1015.pdf**
- Ritter, Cherry & Dolan. Unsupervised Modeling of Twitter Conversations. *HLT-NAACL*, 2010.
  - **http://aclweb.org/anthology//N/N10/N10-1020.pdf**
- **Background:** Jurafsky & Martin, ch. 5 and 6.1–6.5