Review Slides

Introduction to Natural Language Processing
Computer Science 585—Fall 2009
University of Massachusetts Amherst

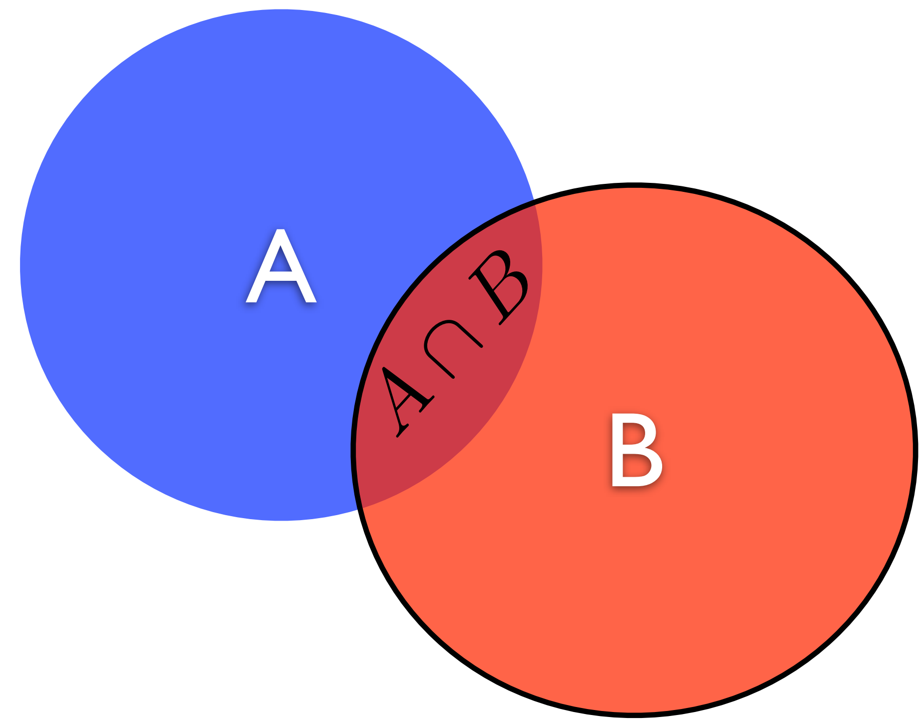
David Smith

Final Exam

- Wednesday, Dec. 16, 10:30, CS 142
- At least 2/3 from course's second half
- Focus on modeling techniques, such as:
  - Log-linear models
  - Sequence labeling, e.g. for information extraction
  - Formal semantics, simple λ-expressions
  - Word clustering
  - Simple machine translation algorithms: IBM Model-1, ITG

Conditional Probability

$$P(A \mid B) = \frac{P(A,B)}{P(B)}$$



$$P(A,B) = P(B)P(A \mid B) = P(A)P(B \mid A)$$

$$\begin{aligned} P(A_1, A_2, \ldots, A_n) &= P(A_1)P(A_2 \mid A_1)P(A_3 \mid A_1, A_2) \\ &\quad \cdots P(A_n \mid A_1, \ldots, A_{n-1}) \end{aligned}$$

Chain rule

Independence

$$
\begin{array}{rcl}
P(A, B) & = & P(A)P(B) \\
& \Leftrightarrow & \\
P(A \mid B) = P(A) & \wedge & P(B \mid A) = P(B)
\end{array}
$$

In coding terms, knowing *B* doesn't help in decoding *A*, and vice versa.

Another View of Markov Models

$$
\begin{aligned}
p(w_1, w_2, \ldots, w_n) \quad &= \quad p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_1, w_2) \\
&\quad\quad p(w_4 \mid w_1, w_2, w_3) \cdots p(w_n \mid p_1, \ldots, p_{n-1})
\end{aligned}
$$

Another View of Markov Models

$$
\begin{aligned}
p(w_1, w_2, \ldots, w_n) \quad &= \quad p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_1, w_2) \\
&\quad\quad p(w_4 \mid w_1, w_2, w_3) \cdots p(w_n \mid p_1, \ldots, p_{n-1})
\end{aligned}
$$

Markov independence assumption

$$p(w_i \mid w_1, \ldots w_{i-1}) \approx p(w_i \mid w_{i-1})$$

Another View of Markov Models

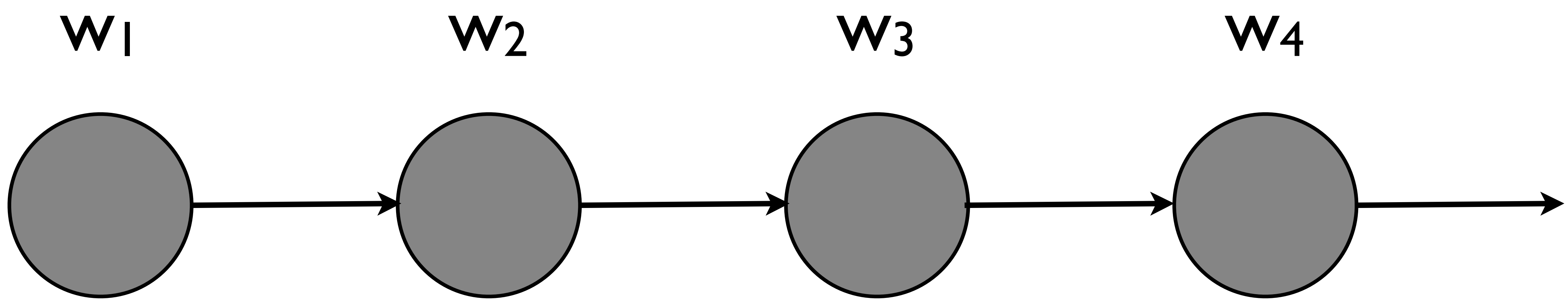
$$\begin{aligned}p(w_1, w_2, \ldots, w_n) &= p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_1, w_2)\\ &\quad p(w_4 \mid w_1, w_2, w_3) \cdots p(w_n \mid p_1, \ldots, p_{n-1})\end{aligned}$$

Markov independence assumption

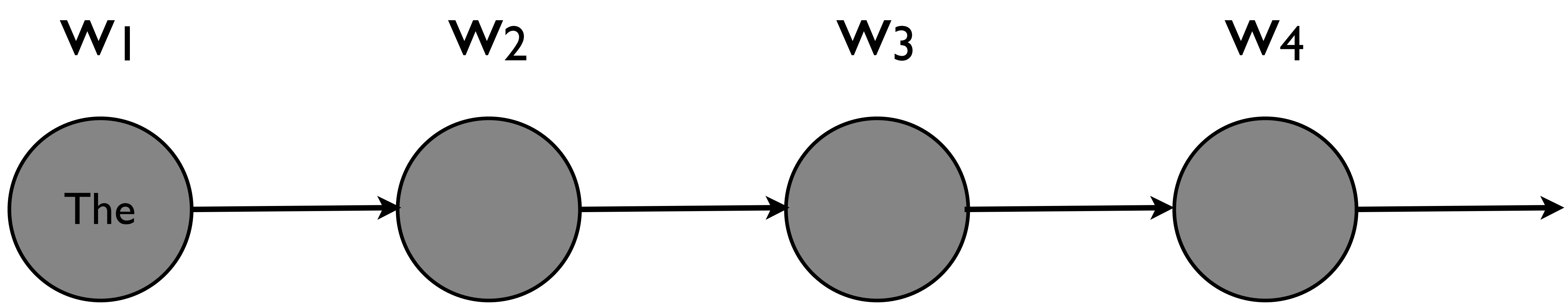
$$p(w_i \mid w_1, \ldots w_{i-1}) \approx p(w_i \mid w_{i-1})$$

$$\begin{aligned}p(w_1, w_2, \ldots, w_n) &\approx p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_2)\\ &\quad p(w_4 \mid w_3) \cdots p(w_n \mid p_{n-1})\end{aligned}$$

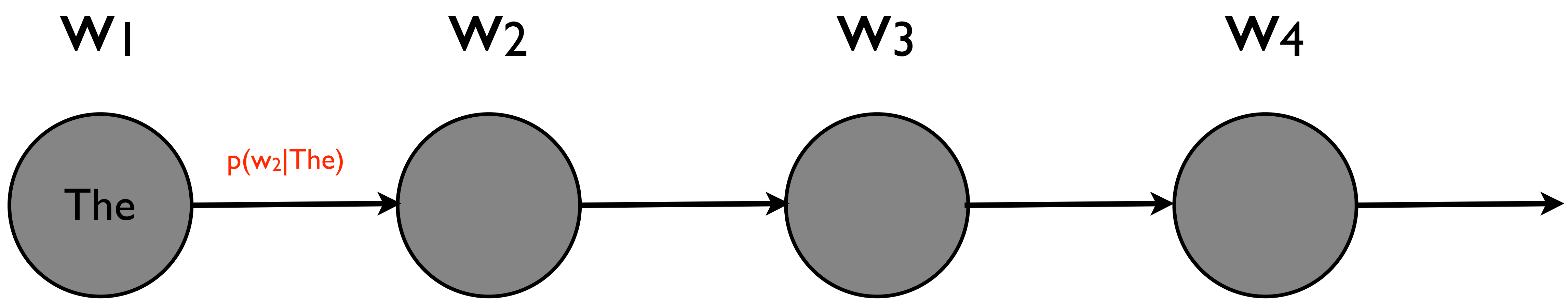
Yet Another View



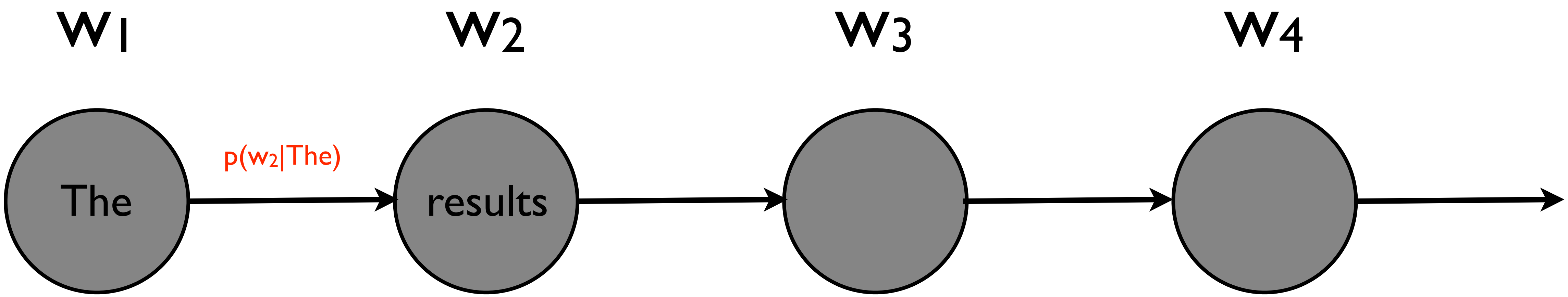
Yet Another View



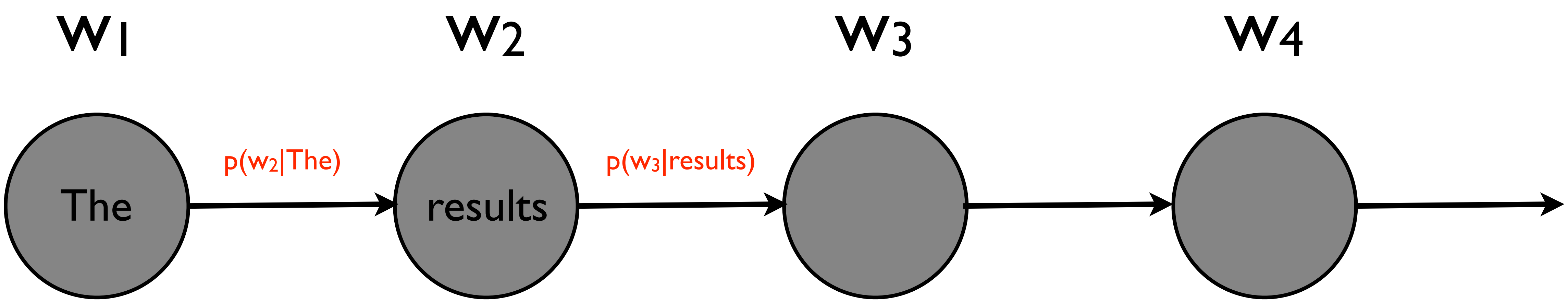
Yet Another View



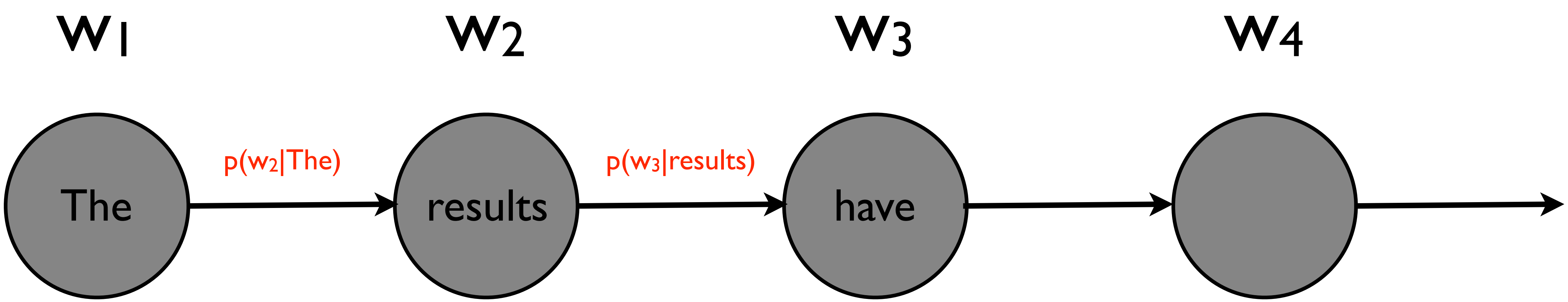
Yet Another View



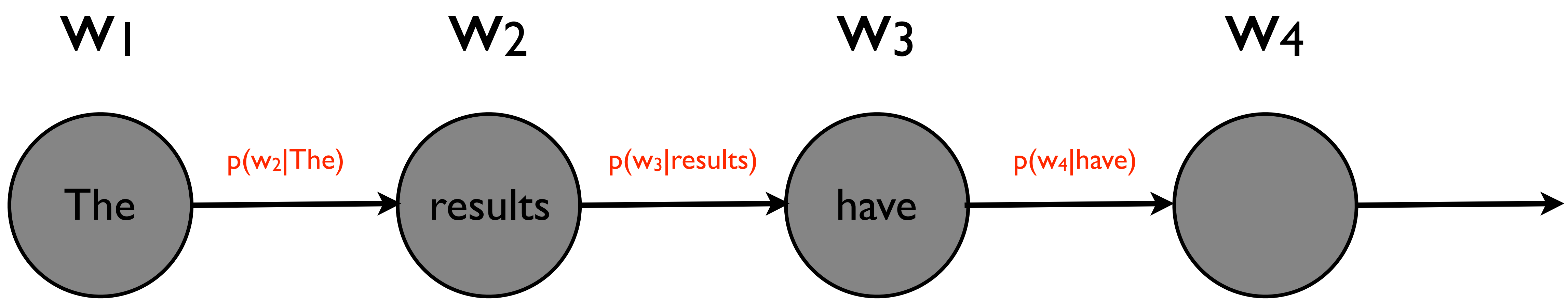
Yet Another View



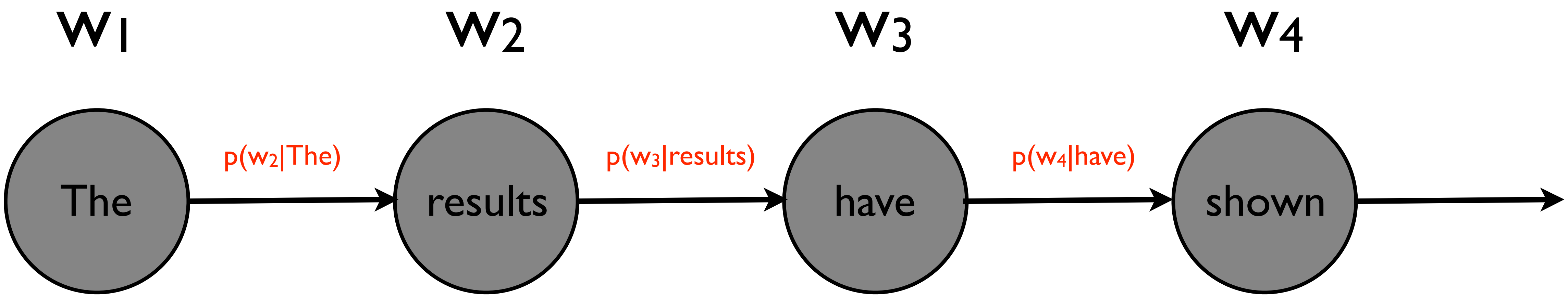
Yet Another View



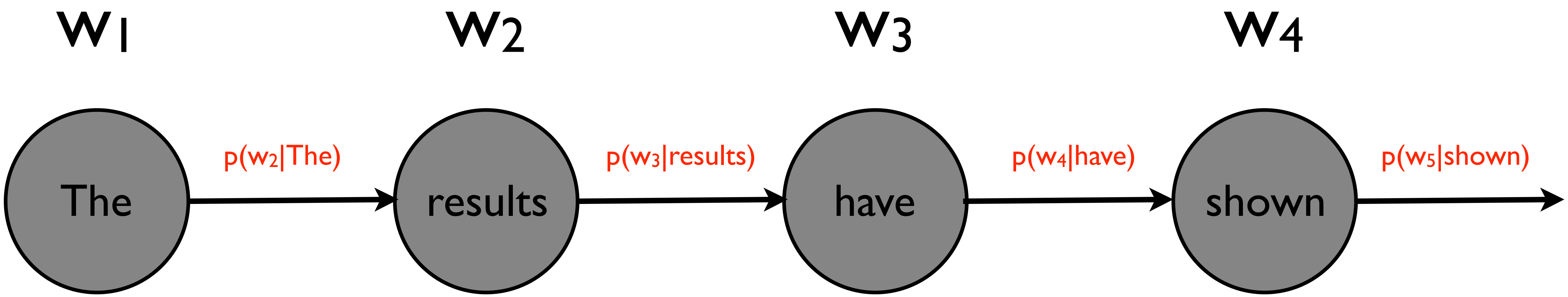
Yet Another View



Yet Another View

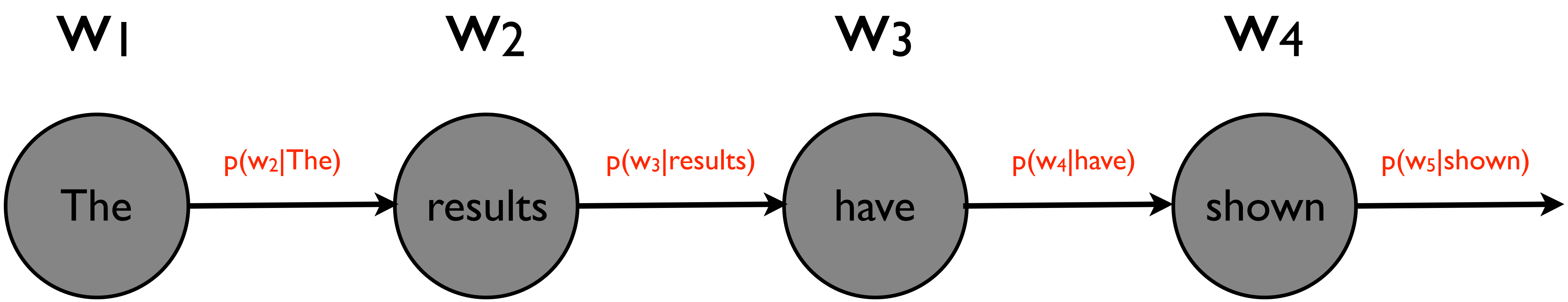


Yet Another View



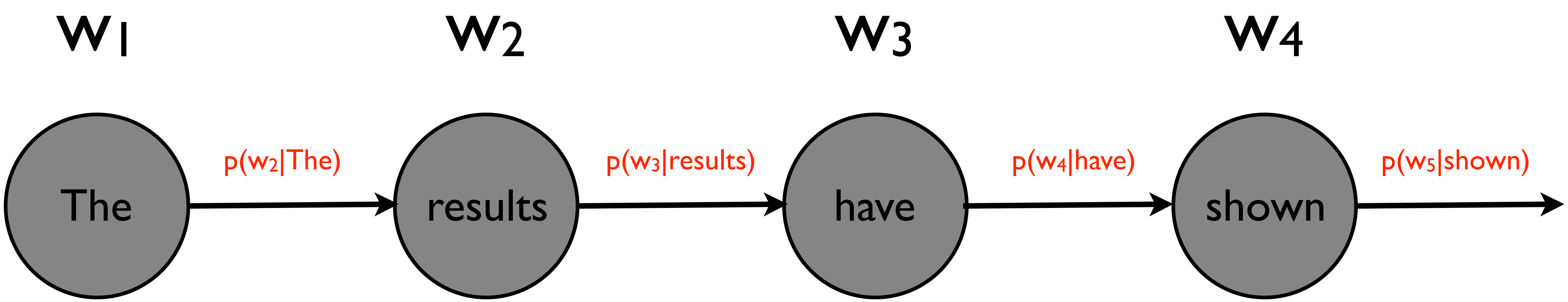
Yet Another View

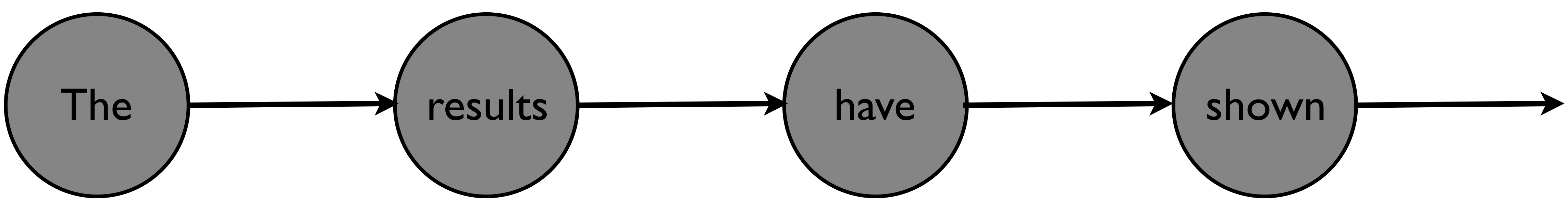
Directed graphical models: *lack* of edge means conditional independence



Yet Another View

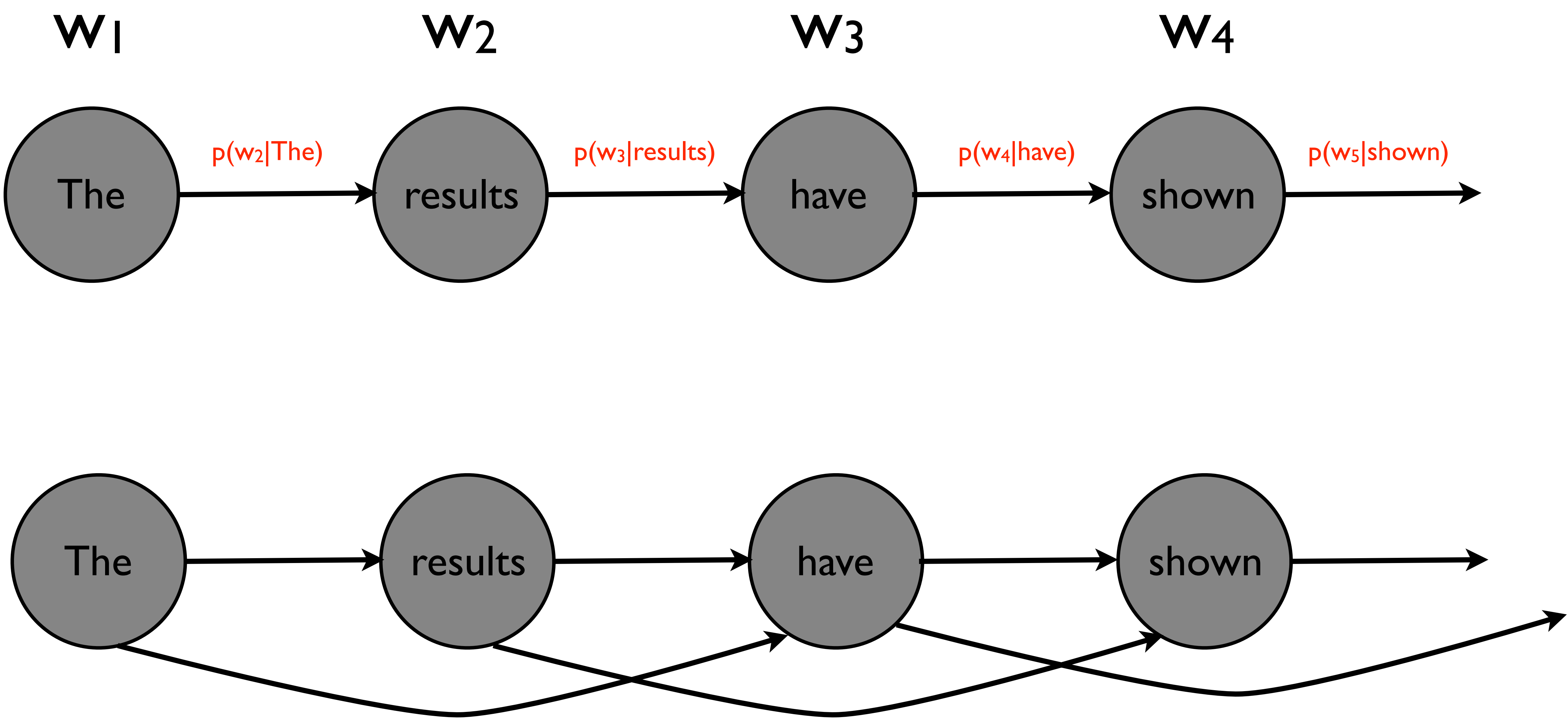
Directed graphical models: *lack* of edge means conditional independence





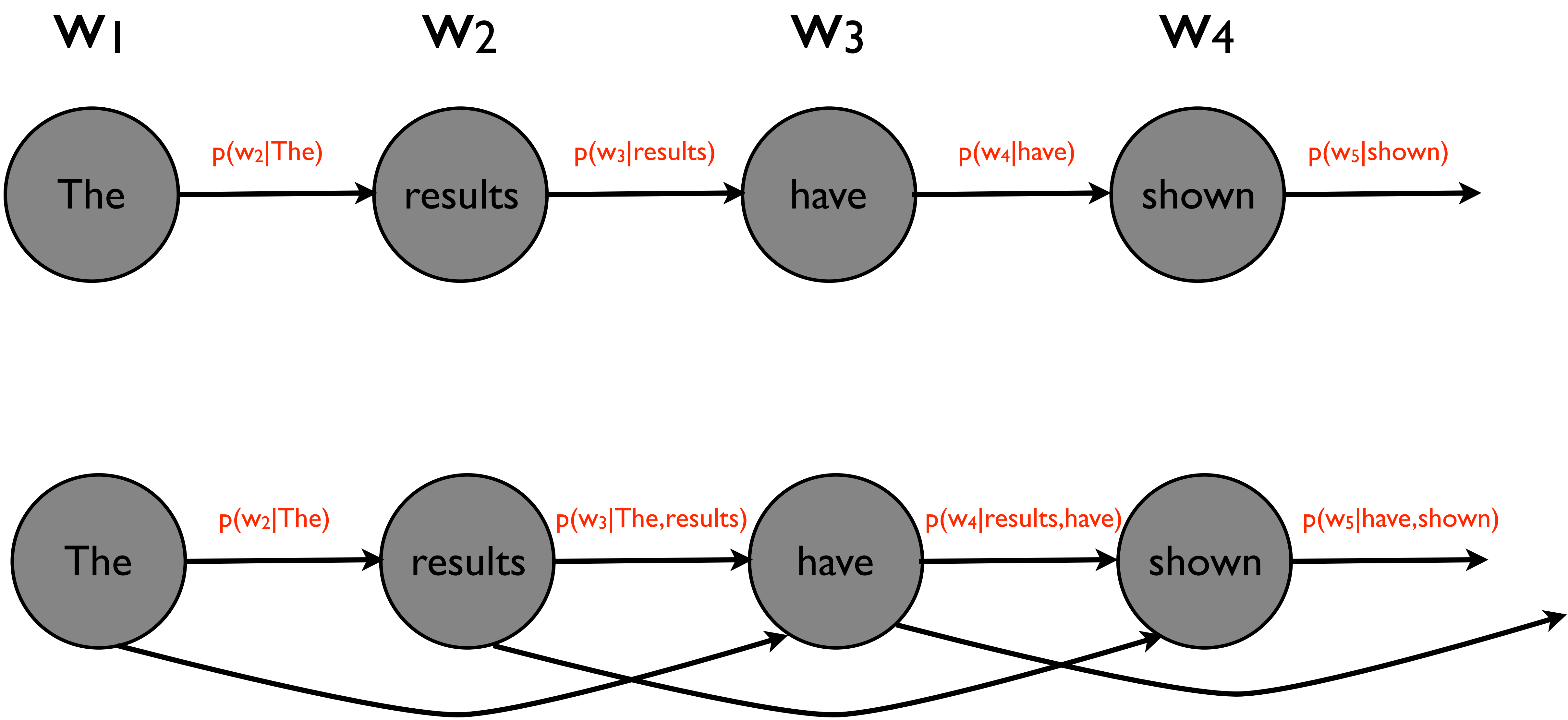
Yet Another View

Directed graphical models: *lack* of edge means conditional independence

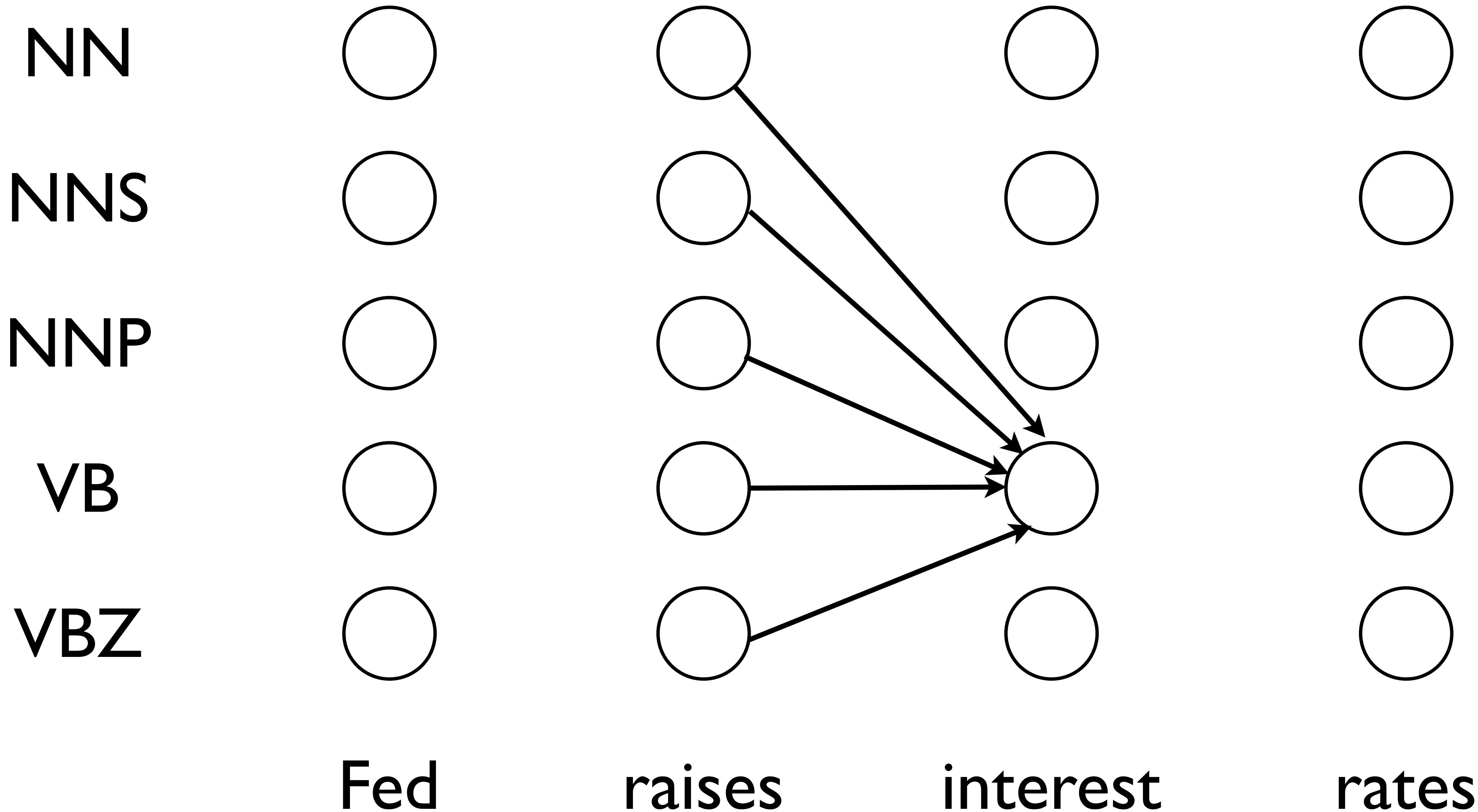


Yet Another View

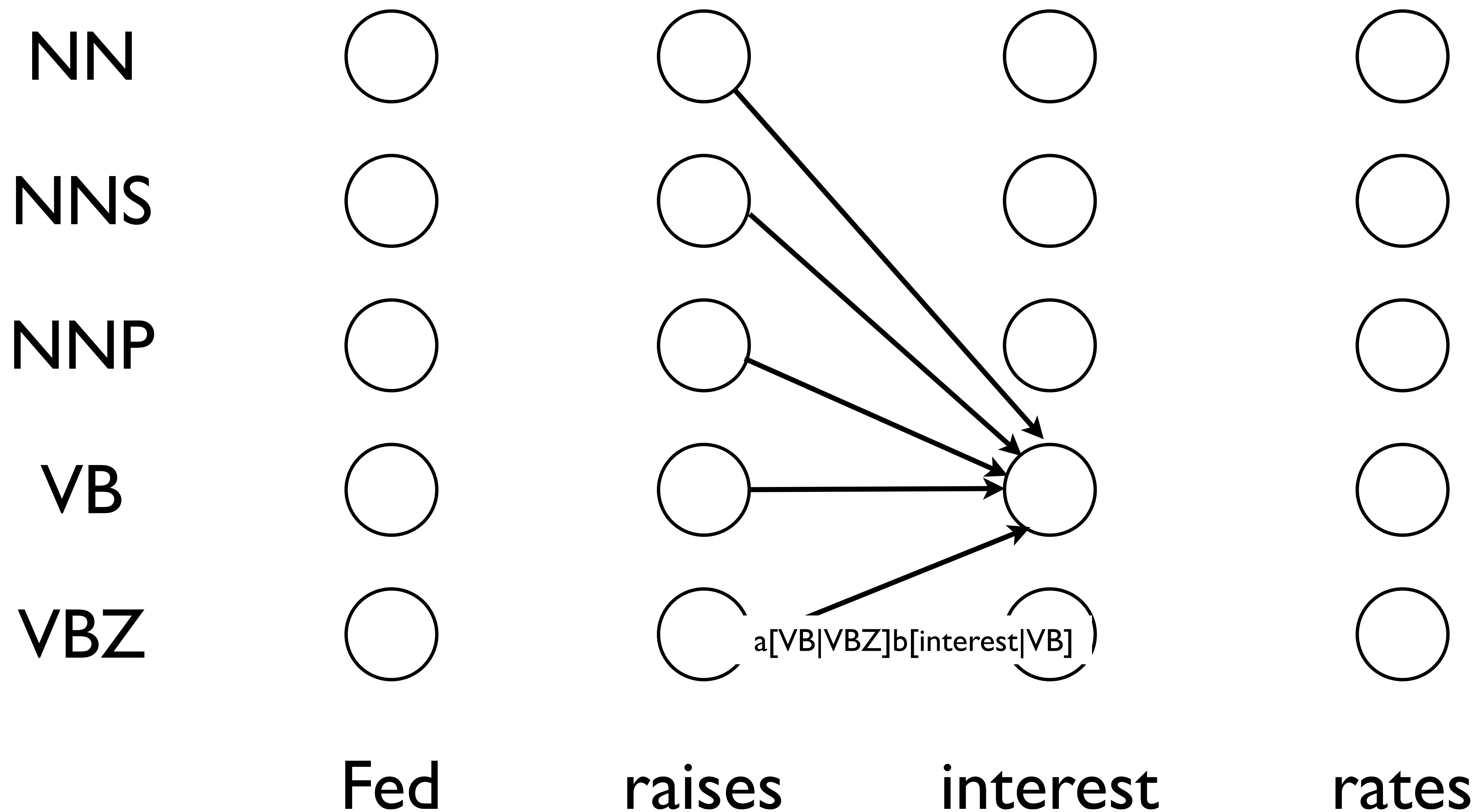
Directed graphical models: *lack* of edge means conditional independence



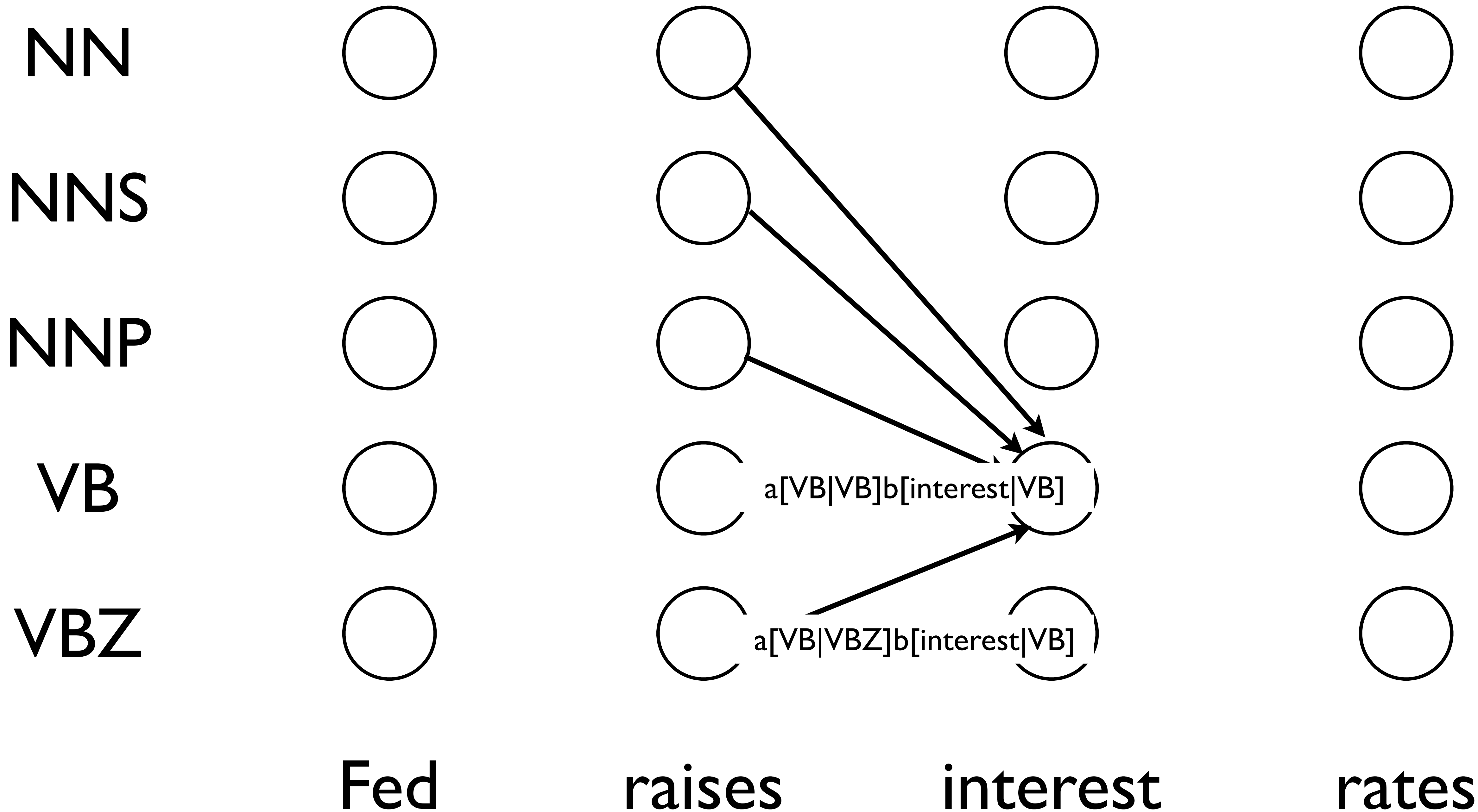
Forward Algorithm (LM)



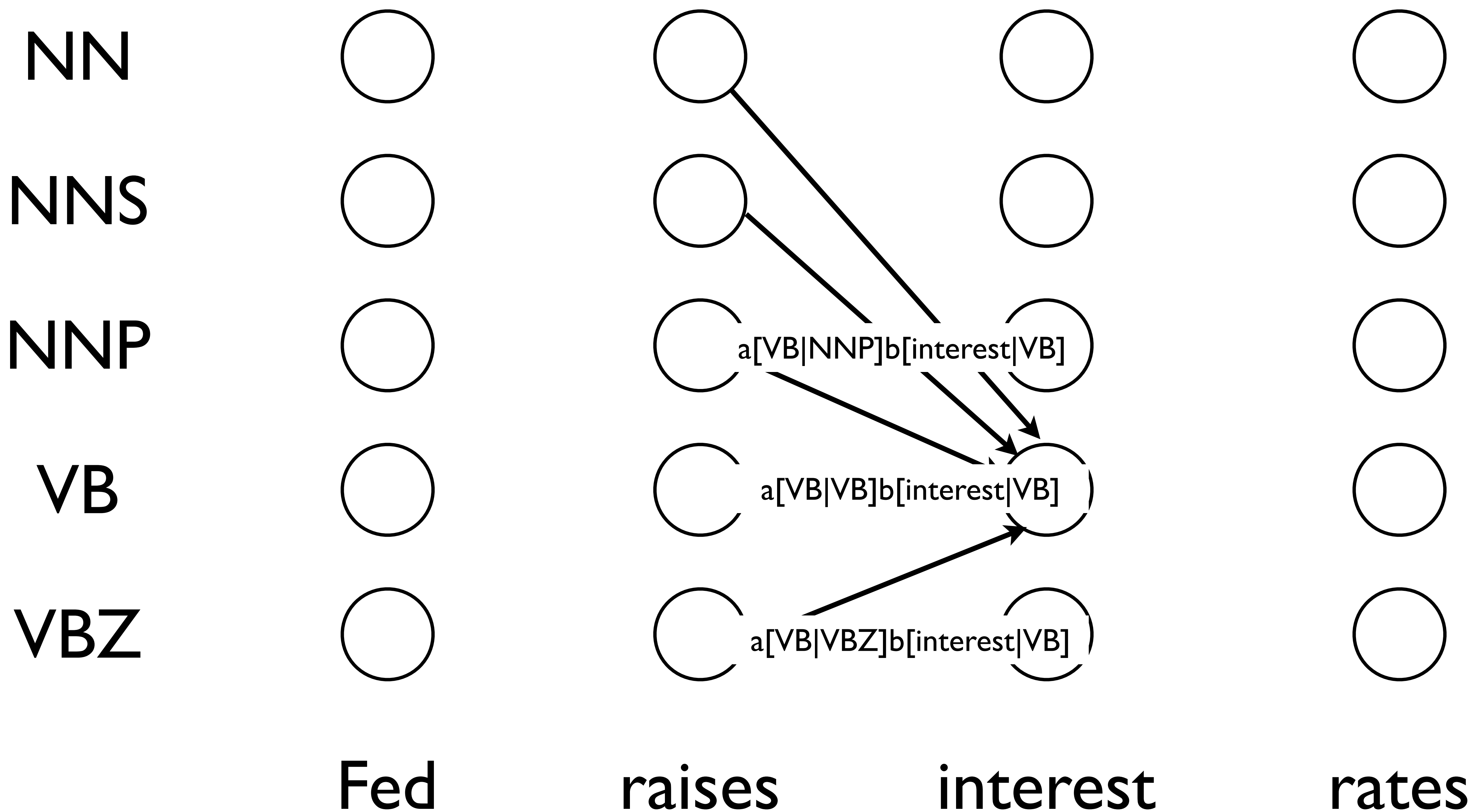
Forward Algorithm (LM)



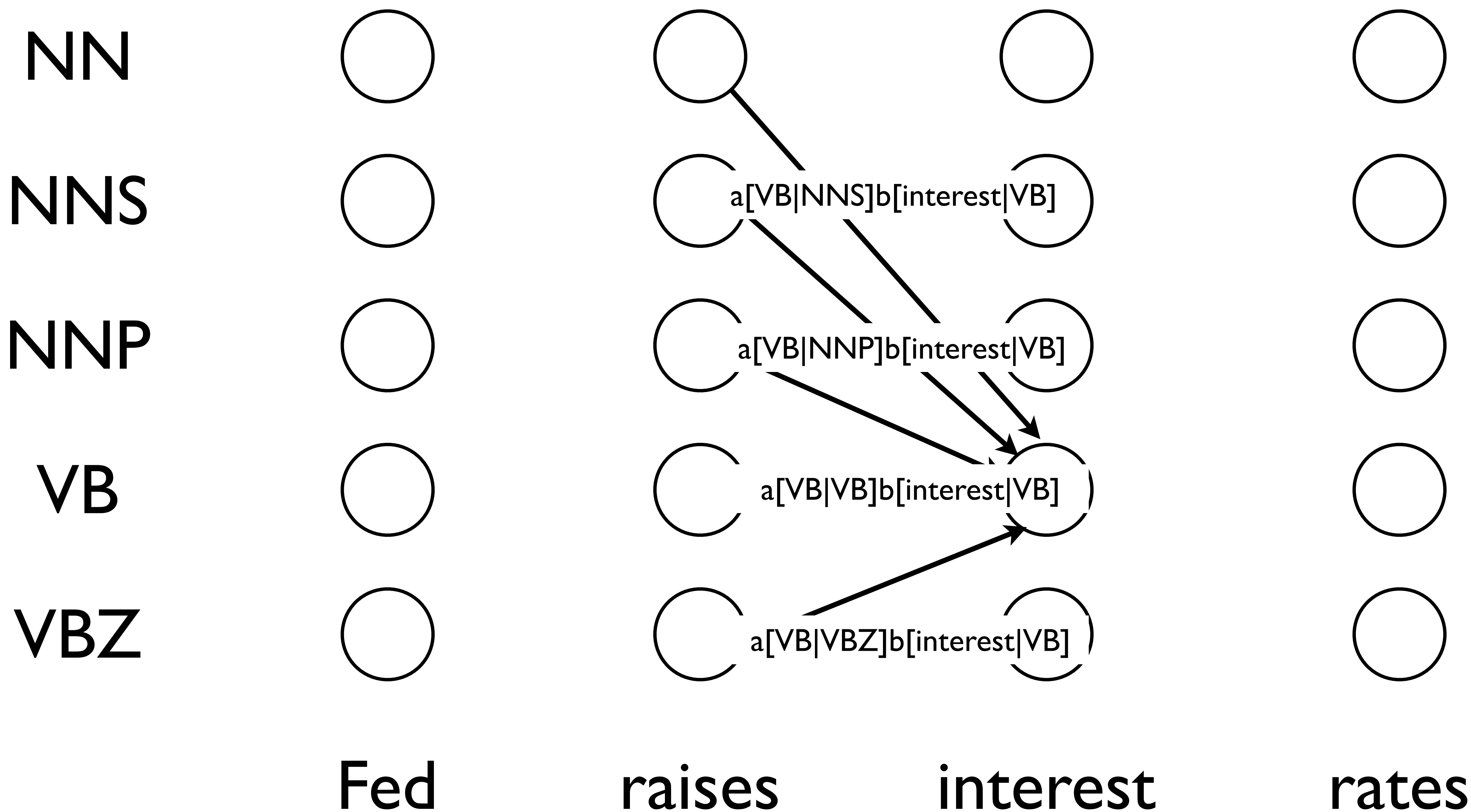
Forward Algorithm (LM)



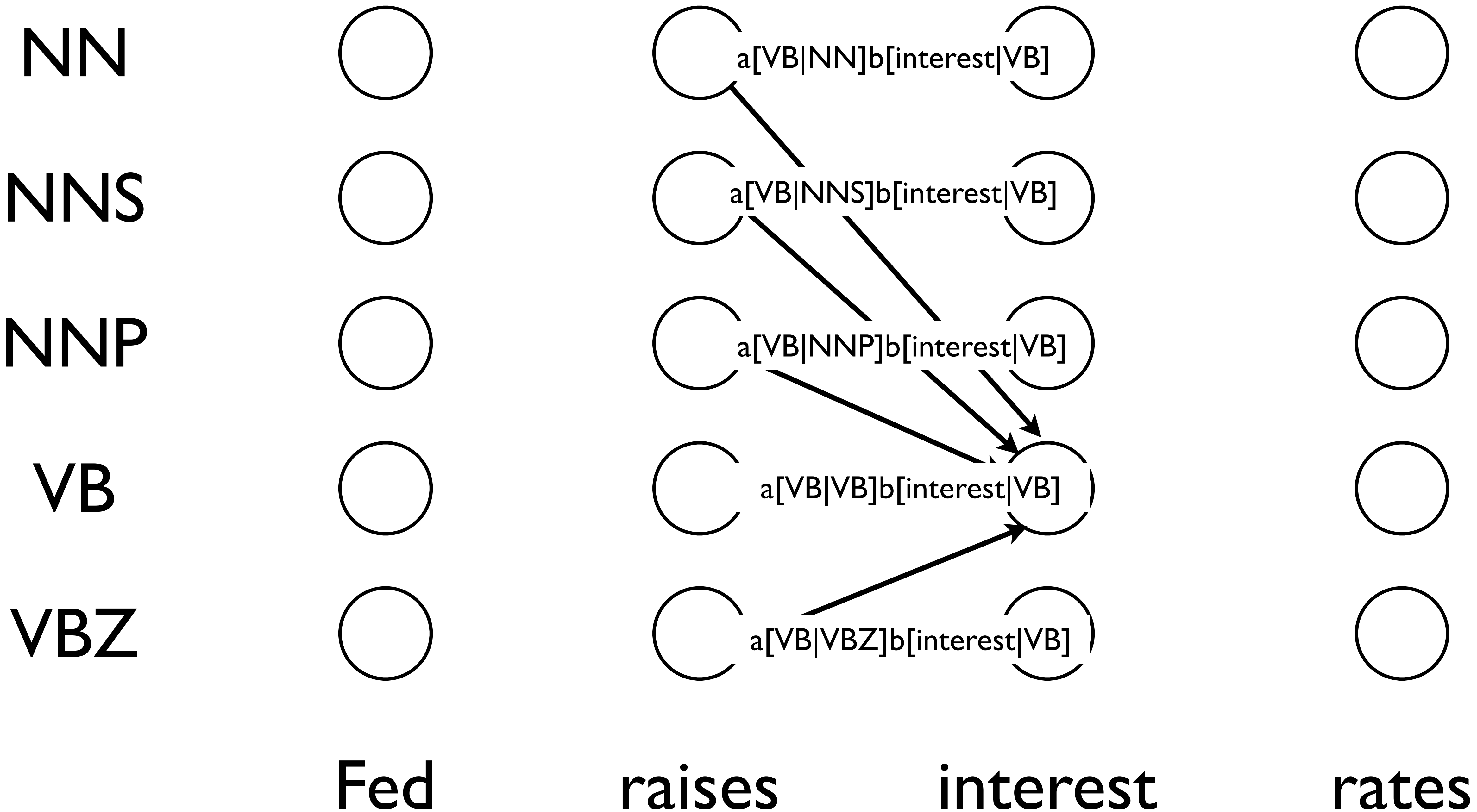
Forward Algorithm (LM)



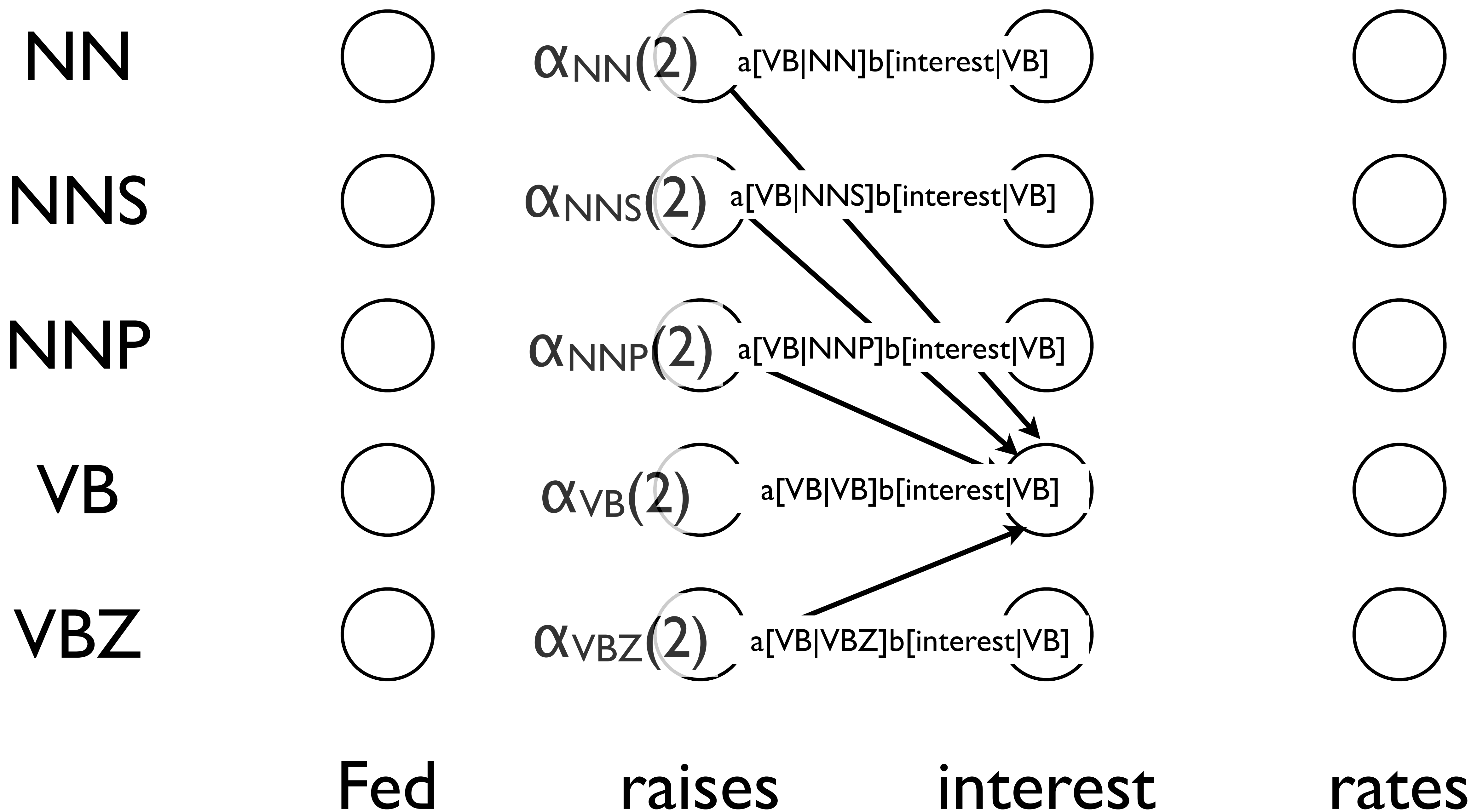
Forward Algorithm (LM)



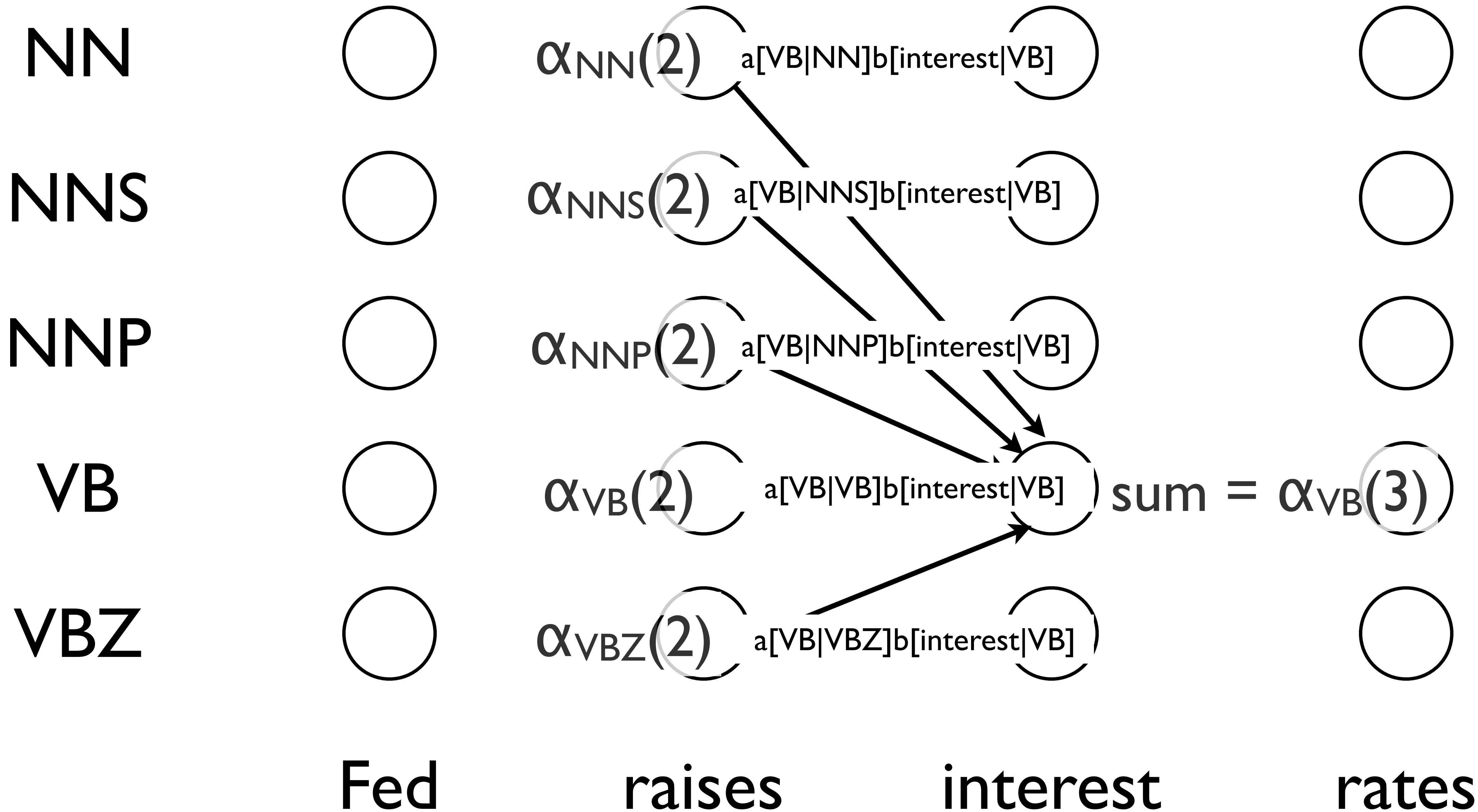
Forward Algorithm (LM)



Forward Algorithm (LM)



Forward Algorithm (LM)



Setting up a Classifier

Setting up a Classifier

- What we want:

  $p(☺ \mid w_1, w_2, ..., w_n) > p(☹ \mid w_1, w_2, ..., w_n)$?

Setting up a Classifier

- What we want:

  $p(☺ \mid w_1, w_2, ..., w_n) > p(☹ \mid w_1, w_2, ..., w_n)$?

- What we know how to build:

Setting up a Classifier

- What we want:

  $p(☺ \mid w_1, w_2, \ldots, w_n) > p(☹ \mid w_1, w_2, \ldots, w_n)$?

- What we know how to build:
  - A language model for each class

Setting up a Classifier

- What we want:

  $p(☺ \mid w_1, w_2, ..., w_n) > p(☹ \mid w_1, w_2, ..., w_n)$?

- What we know how to build:
  - A language model for each class
    - $p(w_1, w_2, ..., w_n \mid ☺)$

Setting up a Classifier

- What we want:

  $p(☺ \mid w_1, w_2, ..., w_n) > p(☹ \mid w_1, w_2, ..., w_n)$?

- What we know how to build:
  - A language model for each class
    - $p(w_1, w_2, ..., w_n \mid ☺)$
    - $p(w_1, w_2, ..., w_n \mid ☹)$

Bayes' Theorem

By the definition of conditional probability:

$$P(A, B) = P(B)P(A \mid B) = P(A)P(B \mid A)$$

we can show:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Seemingly trivial result from 1763; interesting consequences...

REV. T. BAYES

A “Bayesian” Classifier

$$p(R \mid w_1, w_2, \ldots, w_n) = \frac{p(R)p(w_1, w_2, \ldots, w_n \mid R)}{p(w_1, w_2, \ldots, w_n)}$$

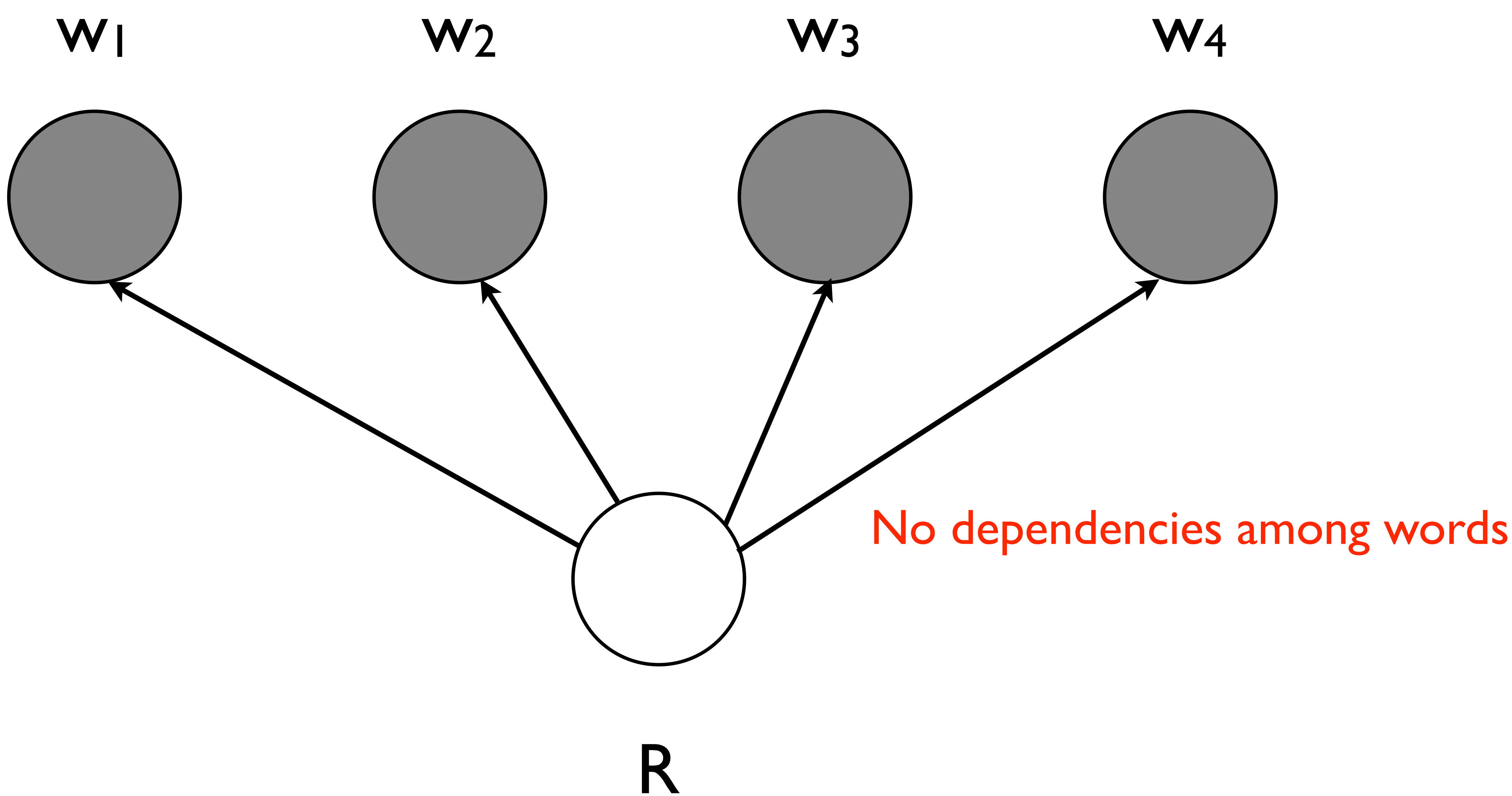
$$\max_{R \in \{\ddot\smile, \ddot\frown\}} p(R \mid w_1, w_2, \ldots, w_n) = \max_{R \in \{\ddot\smile, \ddot\frown\}} p(R)p(w_1, w_2, \ldots, w_n \mid R)$$

Posterior

Prior

Likelihood

Naive Bayes Classifier



NB on Movie Reviews

- Train models for positive, negative
- For each review, find higher posterior
- Which word probability ratios are highest?

```
>>> classifier.show_most_informative_features(5)

classifier.show_most_informative_features(5)
Most Informative Features
   contains(outstanding) = True              pos : neg    =     14.1 : 1.0
         contains(mulan) = True              pos : neg    =      8.3 : 1.0
        contains(seagal) = True              neg : pos    =      7.8 : 1.0
   contains(wonderfully) = True              pos : neg    =      6.6 : 1.0
         contains(damon) = True              pos : neg    =      6.1 : 1.0
```

What's Wrong With NB?

- What happens for word dependencies are strong?
- What happens when some words occur only once?
- What happens when the classifier sees a new word?

Generative vs. Conditional

- What is the most likely label for a given input?
- How likely is a given label for a given input?
- What is the most likely input value?
- How likely is a given input value?
- How likely is a given input value with a given label?
- What is the most likely label for an input that might have one of two values (but we don't know which)?

Generative vs. Conditional

- What is the most likely label for a given input?
- How likely is a given label for a given input?
- What is the most likely input value?
- How likely is a given input value?
- How likely is a given input value with a given label?
- What is the most likely label for an input that might have one of two values (but we don't know which)?

Sequence Labeling

- Inputs: $x = (x_1, \ldots, x_n)$
- Labels: $y = (y_1, \ldots, y_n)$
- Typical goal: Given x, predict y

- Example sequence labeling tasks
  - Part-of-speech tagging
  - Named-entity-recognition (NER)
    - Label people, places, organizations

NER Example:

Red Sox and Their Fans Let Loose

Elise Amendola/Associated Press

Fans of the slugger David Ortiz in Boston's Copley Square.

By PETE THAMEL
Published: October 31, 2007

BOSTON, Oct. 30 — Jonathan Papelbon turned Boston's World Series victory parade into a full-scale dance party Tuesday as the Red Sox put an exclamation point on the 2007 season.

E-MAIL
PRINT
REPRINTS
SAVE

First Solution: Maximum Entropy Classifier

- Conditional model p(y|x).
  - Do not waste effort modeling p(x), since x is given at test time anyway.
  - Allows more complicated input features, since we do not need to model dependencies between them.
- Feature functions f(x,y):
  - $f_1(x,y)$ = { word is Boston & y=Location }
  - $f_2(x,y)$ = { first letter capitalized & y=Name }
  - $f_3(x,y)$ = { x is an HTML link & y=Location}

First Solution: MaxEnt Classifier

- How should we choose a classifier?

- Principle of maximum entropy
  - We want a classifier that:
    - Matches feature constraints from training data.
    - Predictions maximize entropy.

- There is a unique, exponential family distribution that meets these criteria.

First Solution: MaxEnt Classifier

- Problem with using a maximum entropy classifier for sequence labeling:
- It makes decisions at each position independently!

Second Solution: HMM

$$P(\mathbf{y},\mathbf{x}) = \prod_t P(y_t \mid y_{t-1}) P(x \mid y_t)$$

- Defines a generative process.
- Can be viewed as a weighted finite state machine.

Second Solution: HMM

- HMM problems: (ON BOARD)
  - Probability of an input sequence.
  - Most likely label sequence given an input sequence.
  - Learning with known label sequences.
  - Learning with unknown label sequences?

Second Solution: HMM

- How can represent we multiple features in an HMM?
  - Treat them as conditionally independent given the class label?
    - The example features we talked about are not independent.
  - Try to model a more complex generative process of the input features?
    - We may lose tractability (i.e. lose a dynamic programming for exact inference).

Second Solution: HMM

- Let's use a conditional model instead.

Third Solution: MEMM

- Use a series of maximum entropy classifiers that know the previous label.
- Define a Viterbi algorithm for inference.

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_t P(y_t \mid y_{t-1}, \mathbf{x})$$

Third Solution: MEMM

- Combines the advantages of maximum entropy and HMM!
- But there is a problem…

Problem with MEMMs: Label Bias

- In some state space configurations, MEMMs essentially completely ignore the inputs.

- Example (ON BOARD).

- This is not a problem for HMMs, because the input sequence is generated by the model.

Fourth Solution: Conditional Random Field

- Conditionally-trained, undirected graphical model.
- For a standard linear-chain structure:

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z} \prod_t \Psi_t(y_t, y_{t-1}, \mathbf{x})$$

Bigram model: potentials consider pairs of labels

$$\Psi_t(y_t, y_{t-1}, \mathbf{x}) = \exp\left[\sum_k \lambda_k f_k(y_t, y_{t-1}, \mathbf{x})\right]$$

Dot-product of weights and features

$$Z = \sum_{\mathbf{y}'} \prod_t \Psi(y'_t, y'_{t-1}, \mathbf{x})$$

Normalize over all possible outputs using forward alg.

Fourth Solution: CRF

- Have the advantages of MEMMs, but avoid the label bias problem.

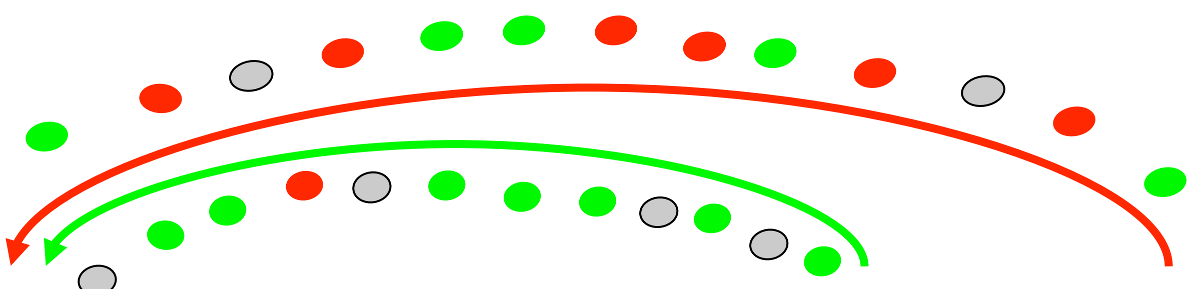
- CRFs are globally normalized, whereas MEMMs are locally normalized.

- Widely used and applied. CRFs give state-the-art results in many domains.

Example Applications

- CRFs have been applied to:
  - Part-of-speech tagging
  - Named-entity-recognition
  - Table extraction
  - Gene prediction
  - Chinese word segmentation
  - Extracting information from research papers.
  - Many more…

Edge-Factored Parsers (McDonald et al. 2005)



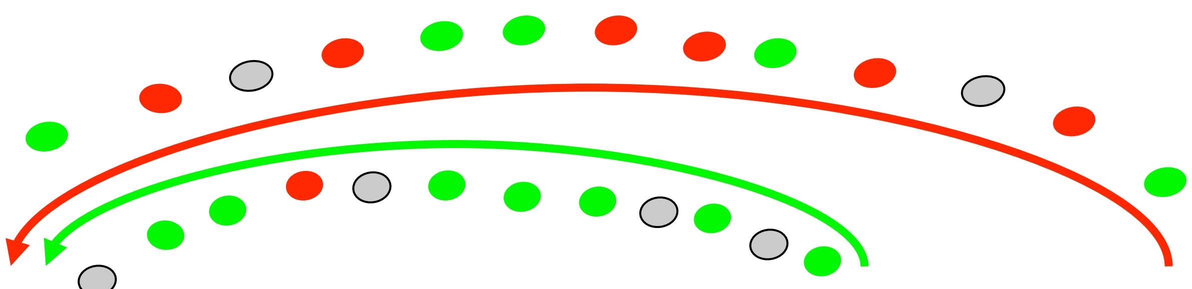
Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

"It was a bright cold day in April and the clocks were striking thirteen"

31

Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?



Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

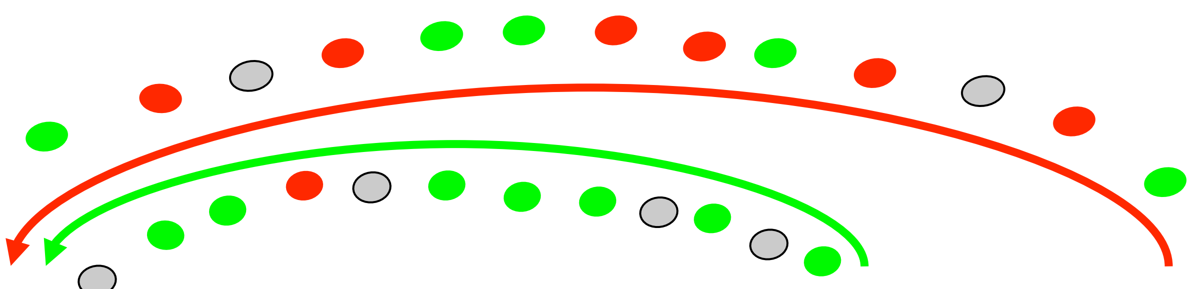
"It was a bright cold day in April and the clocks were striking thirteen"

31

Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?

our current weight vector



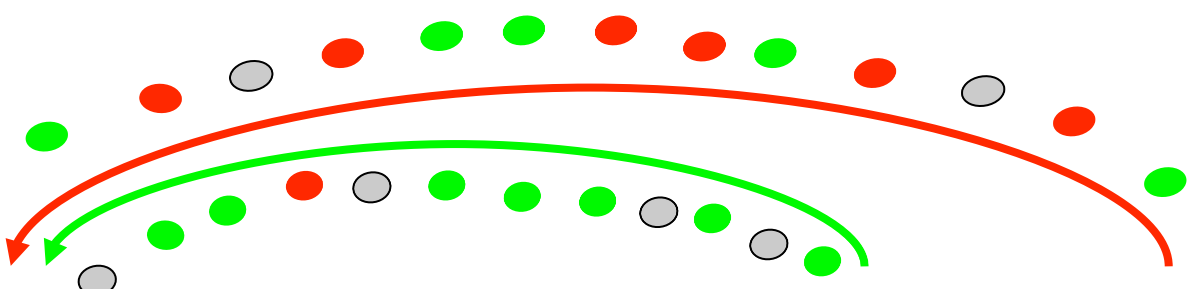
Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

"It was a bright cold day in April and the clocks were striking thirteen"

Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
- Score of an edge e = θ · **features**(e)

our current weight vector



Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

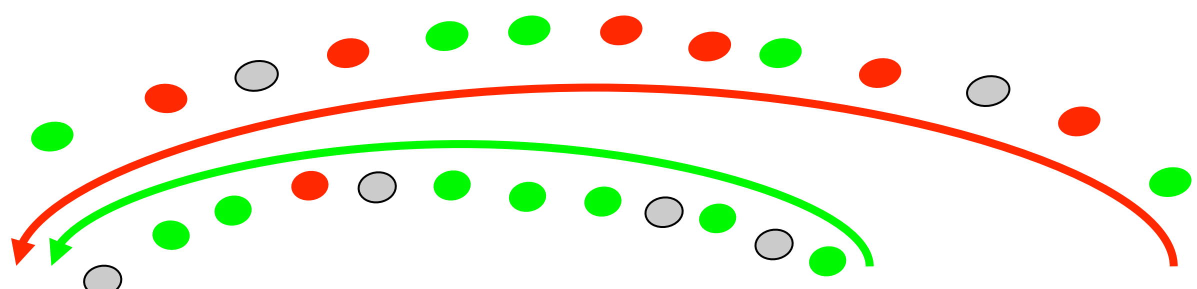
“It was a bright cold day in April and the clocks were striking thirteen”

31

Edge-Factored Parsers (McDonald et al. 2005)

- Which edge is better?
- Score of an edge e = θ · **features**(e)
- Standard algos ➔ valid parse with max <u>total</u> score

our current weight vector



Byl	jasný	studený	dubnový	den	a	hodiny	odbíjely	třináctou
V	A	A	A	N	J	N	V	C
byl	jasn	stud	dubn	den	a	hodi	odbí	třin

“It was a bright cold day in April and the clocks were striking thirteen”

31

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data

$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$

2. Initialize all parameters to zero
3. Classify training data with current parameters; calculate expectations $E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$
4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$
5. Take a step in the direction of the gradient
6. Repeat from 3 until convergence

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data

$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$

2. Initialize all parameters to zero

3. Classify training data with current parameters; calculate expectations $E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$

4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$

5. Take a step in the direction of the gradient

6. Repeat from 3 until convergen

Where have we seen expected counts before?

Recipe for Conditional Training of p(y | x)

1. Gather constraints/features from training data

$$\alpha_{iy} = \tilde{E}[f_{iy}] = \sum_{x_j, y_j \in D} f_{iy}(x_j, y_j)$$

2. Initialize all parameters to zero
3. Classify training data with current parameters; calculate expectations $E_\Theta[f_{iy}] = \sum_{x_j \in D} \sum_{y'} p_\Theta(y'|x_j) f_{iy}(x_j, y')$
4. Gradient is $\tilde{E}[f_{iy}] - E_\Theta[f_{iy}]$
5. Take a step in the direction of the gradient
6. Repeat from 3 until convergence

EM!

Where have we seen expected counts before?

Gradient-Based Training

- $\lambda := \lambda + \text{rate} * \text{Gradient}(F)$
- After all training examples? (batch)
- After every example? (on-line)
- Use second derivative for faster learning?
- A big field: numerical optimization

Overfitting

- If we have too many features, we can choose weights to model the training data perfectly
- If we have a feature that only appears in spam training, not ham training, it will get weight ∞ to maximize p(spam | feature) at 1.
- These behaviors
  - Overfit the training data
  - Will probably do poorly on test data

Solutions to Overfitting

- Throw out rare features.
  - Require every feature to occur > 4 times, and > 0 times with ling, and > 0 times with spam.
- Only keep, e.g., 1000 features.
  - Add one at a time, always greedily picking the one that most improves performance on held-out data.
- Smooth the observed feature counts.
- Smooth the weights by using a prior.
  - max $p(\lambda|data)$ = max $p(\lambda, data) = p(\lambda)p(data|\lambda)$
  - decree $p(\lambda)$ to be high when most weights close to 0

Smoothing with Priors

- What if we had a prior expectation that parameter values wouldn't be very large?
- We could then balance evidence suggesting large (or infinite) parameters against our prior expectation.
- The evidence would never totally defeat the prior, and parameters would be smoothed (and kept finite)
- We can do this explicitly by changing the optimization objective to maximum posterior likelihood:

  $\log P(y, \lambda \mid x) = \log P(\lambda) + \log P(y \mid x, \lambda)$

  Posterior Prior Likelihood

(First Order) Logic
Some Preliminaries

(First Order) Logic
Some Preliminaries

Three major kinds of objects

(First Order) Logic
Some Preliminaries

Three major kinds of objects

1. Booleans
   - Roughly, the semantic values of sentences

(First Order) Logic
Some Preliminaries

Three major kinds of objects

1. Booleans
   - Roughly, the semantic values of sentences
2. Entities
   - Values of NPs, e.g., objects like this slide
   - Maybe also other types of entities, like times

(First Order) Logic
Some Preliminaries

Three major kinds of objects

1. Booleans
   - Roughly, the semantic values of sentences
2. Entities
   - Values of NPs, e.g., objects like this slide
   - Maybe also other types of entities, like times
3. Functions of various types
   - Functions from booleans to booleans (and, or, not)
   - A function from entity to boolean is called a "predicate" – e.g., frog(x), green(x)
   - Functions might return other functions!

(First Order) Logic
Some Preliminaries

Three major kinds of objects

1. Booleans
   - Roughly, the semantic values of sentences
2. Entities
   - Values of NPs, e.g., objects like this slide
   - Maybe also other types of entities, like times
3. Functions of various types
   - Functions from booleans to booleans (and, or, not)
   - A function from entity to boolean is called a "predicate" – e.g., frog(x), green(x)
   - Functions might return other functions!
   - Function might take other functions as arguments!

First-order Representations

First-order Representations

- `Gilly swallowed a goldfish`
  - First attempt: swallowed(Gilly, goldfish)

First-order Representations

- `Gilly swallowed a goldfish`
  - First attempt: swallowed(Gilly, goldfish)
- Better: ∃g goldfish(g) AND swallowed(Gilly, g)

First-order Representations

- `Gilly swallowed a goldfish`
  - First attempt: swallowed(Gilly, goldfish)
- Better: ∃g goldfish(g) AND swallowed(Gilly, g)
- Or using one of our quantifier predicates:
  - exists(λg goldfish(g), λg swallowed(Gilly,g))
  - Equivalently: exists(goldfish, swallowed(Gilly))
    - "In the set of goldfish there exists one swallowed by Gilly"

First-order Representations

- `Gilly swallowed a goldfish`
  - First attempt: swallowed(Gilly, goldfish)
- Better: ∃g goldfish(g) AND swallowed(Gilly, g)
- Or using one of our quantifier predicates:
  - exists(λg goldfish(g), λg swallowed(Gilly,g))
  - Equivalently: exists(goldfish, swallowed(Gilly))
    - "In the set of goldfish there exists one swallowed by Gilly"
- Here goldfish is a predicate on entities
  - This is the same semantic type as red
  - But note: `goldfish` is noun and `red` is adjective

Compositional Semantics

Compositional Semantics

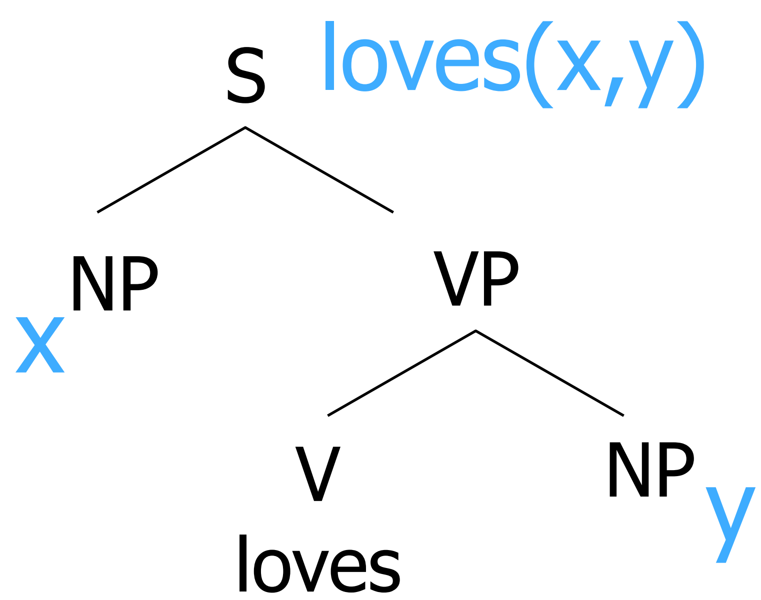
- Add a “sem” feature to each context-free rule
  - S $\rightarrow$ NP loves NP
  - S[sem=loves(x,y)] $\rightarrow$ NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs

Compositional Semantics

- Add a "sem" feature to each context-free rule
  - S $\rightarrow$ NP loves NP
  - S[sem=loves(x,y)] $\rightarrow$ NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:

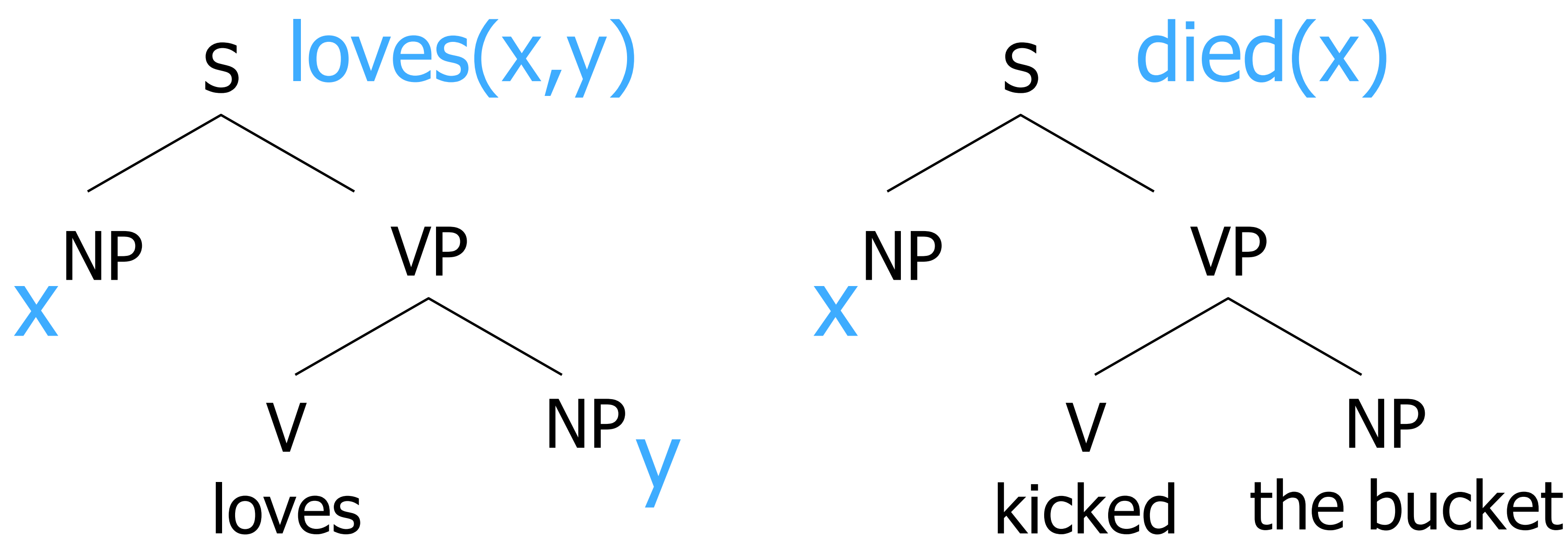
Compositional Semantics

- Add a “sem” feature to each context-free rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:



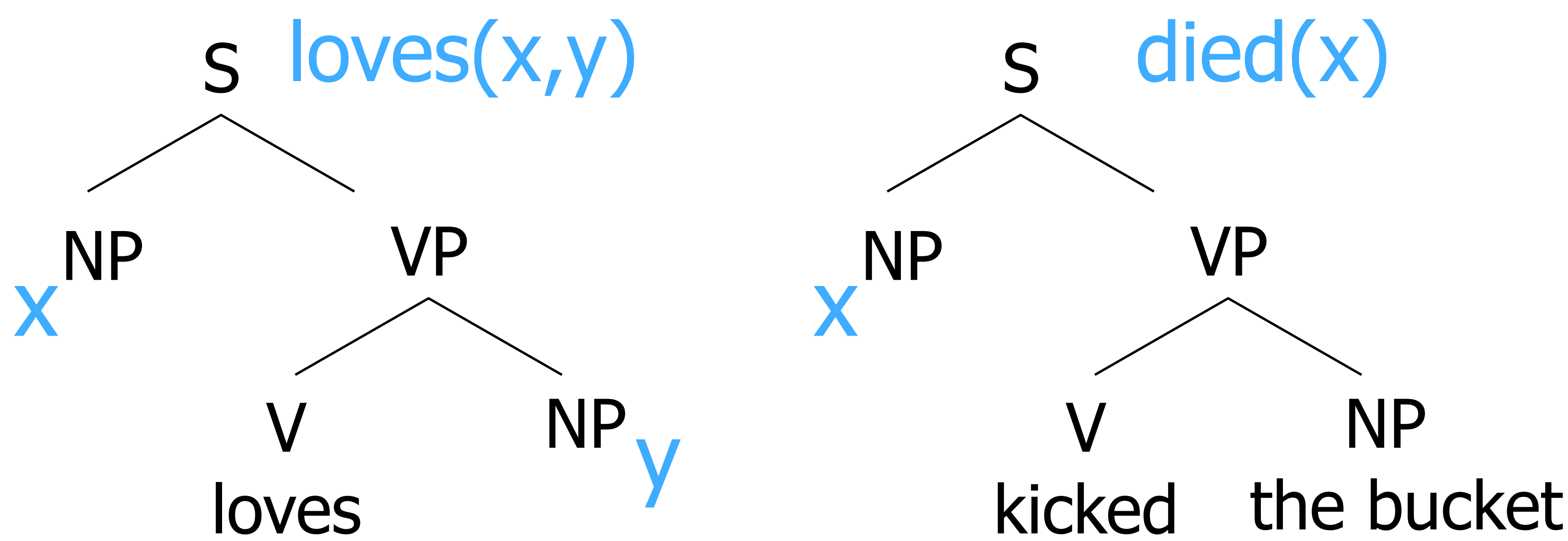
Compositional Semantics

- Add a "sem" feature to each context-free rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:



Compositional Semantics

- Add a “sem” feature to each context-free rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Meaning of S depends on meaning of NPs
- TAG version:



- Template filling: S[sem=showflights(x,y)] → I want a flight from NP[sem=x] to NP[sem=y]

"Non-constituents" in CCG – Right Node Raising

$$
\begin{array}{cccccc}
\textit{Google} & \textit{sells} & \textit{but} & \textit{Microsoft} & \textit{buys} & \textit{shares} \\
\hline
NP & (S\backslash NP)/NP & conj & NP & (S\backslash NP)/NP & NP \\
\xrightarrow{\quad >\mathbf{T}} & & & \xrightarrow{\quad >\mathbf{T}} & & \\
S/(S\backslash NP) & & & S/(S\backslash NP) & & \\
\multicolumn{2}{c}{\xrightarrow{\qquad\qquad >\mathbf{B}}} & & \multicolumn{2}{c}{\xrightarrow{\qquad\qquad >\mathbf{B}}} & \\
\multicolumn{2}{c}{S/NP} & & \multicolumn{2}{c}{S/NP} & \\
\multicolumn{5}{c}{\xrightarrow{\qquad\qquad\qquad\qquad <\Phi>}} & \\
\multicolumn{5}{c}{S/NP} & \\
\multicolumn{6}{c}{\xrightarrow{\qquad\qquad\qquad\qquad\qquad >}} \\
\multicolumn{6}{c}{S}
\end{array}
$$

CCG Semantics

- Categories encode argument sequences
- Parallel syntactic combinator operations and lambda calculus semantic operations

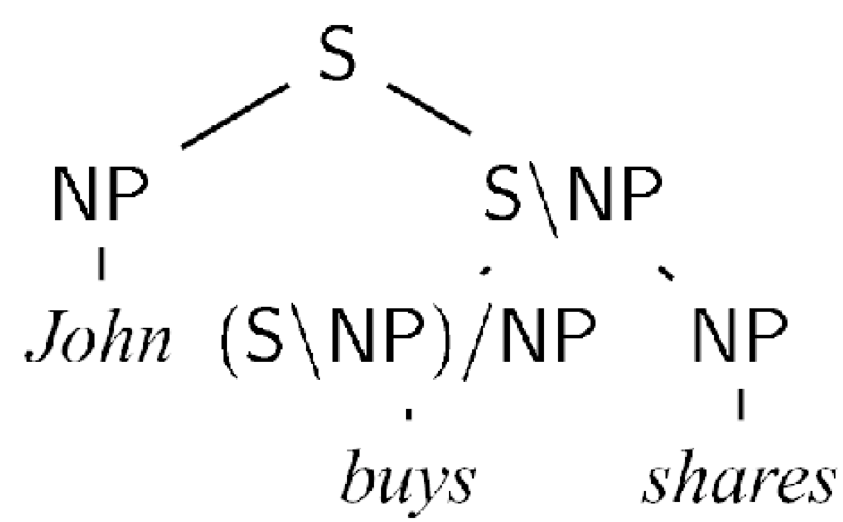
$John \vdash \mathsf{NP} : john'$

$shares \vdash \mathsf{NP} : shares'$

$buys \vdash (\mathsf{S}\backslash\mathsf{NP})/\mathsf{NP} : \lambda x.\lambda y.buys'xy$

$sleeps \vdash \mathsf{S}\backslash\mathsf{NP} : \lambda x.sleeps'x$

$well \vdash (\mathsf{S}\backslash\mathsf{NP})\backslash(\mathsf{S}\backslash\mathsf{NP}) : \lambda f.\lambda x.well'(fx)$



Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  - e.g., k is size of vocabulary
  - the 17^{th} coordinate of w represents **strength** of w's association with vocabulary word 17

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  - e.g., k is size of vocabulary
  - the 17^{th} coordinate of w represents **strength** of w's association with vocabulary word 17

= party

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  - e.g., k is size of vocabulary
  - the 17^{th} coordinate of w represents **strength** of w's association with vocabulary word 17

= party

(0, 0, 3, 1, 0, 7, . . . 1, 0)

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  - e.g., k is size of vocabulary
  - the 17th coordinate of w represents **strength** of w's association with vocabulary word 17

= party

aardvark abacus abandoned abbot abduct above zygote zymurgy

(0, 0, 3, 1, 0, 7, . . . 1, 0)

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  = **party**
  - e.g., k is size of vocabulary
  - the 17th coordinate of w represents **strength** of w's association with vocabulary word 17

aardvark	abacus	abandoned	abbot	abduct	above		zygote	zymurgy
(0,	0,	3,	1,	0,	7,	. . .	1,	0)

From corpus:

Arlen Specter **abandoned** the Republican **party**.
There were lots of **abbots** and nuns dancing at that **party**.
The **party** **above** the art gallery was, **above** all, a laboratory for synthesizing **zygotes** and beer.

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  = **party**
  - e.g., k is size of vocabulary
  - the 17th coordinate of w represents **strength** of w's association with vocabulary word 17

aardvark abacus abandoned abbot abduct above . . . zygote zymurgy

(0, 0, 3, 1, 0, 7, . . . 1, 0)

count too high (too influential)

From corpus:

Arlen Specter **abandoned** the Republican **party**.
There were lots of **abbots** and nuns dancing at that **party**.
The **party** **above** the art gallery was, **above** all, a laboratory for synthesizing **zygotes** and beer.

Words as Vectors

- Represent each word **type** w by a point in k-dimensional space
  - e.g., k is size of vocabulary
  - the 17^{th} coordinate of w represents **strength** of w's association with vocabulary word 17

= **party**

aardvark	abacus	abandoned	abbot	abduct	above		zygote	zymurgy
(0,	0,	3,	1,	0,	7,	. . .	1,	0)

count too high (too influential)

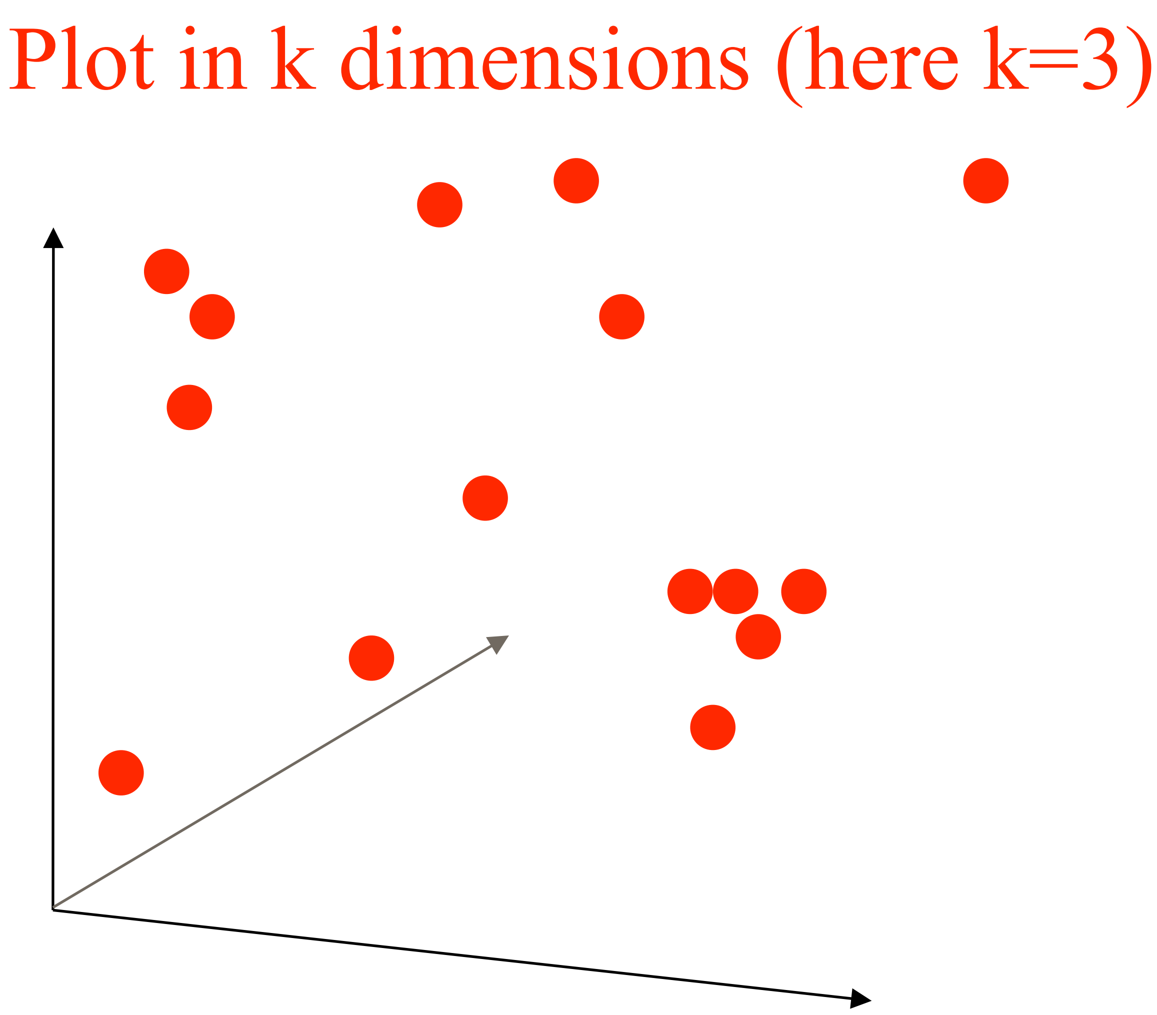
count too low

From corpus:

Arlen Specter **abandoned** the Republican **party**.
There were lots of **abbots** and nuns dancing at that **party**.
The **party** **above** the art gallery was, **above** all, a laboratory for synthesizing **zygotes** and beer.

Learning Classes by Clustering

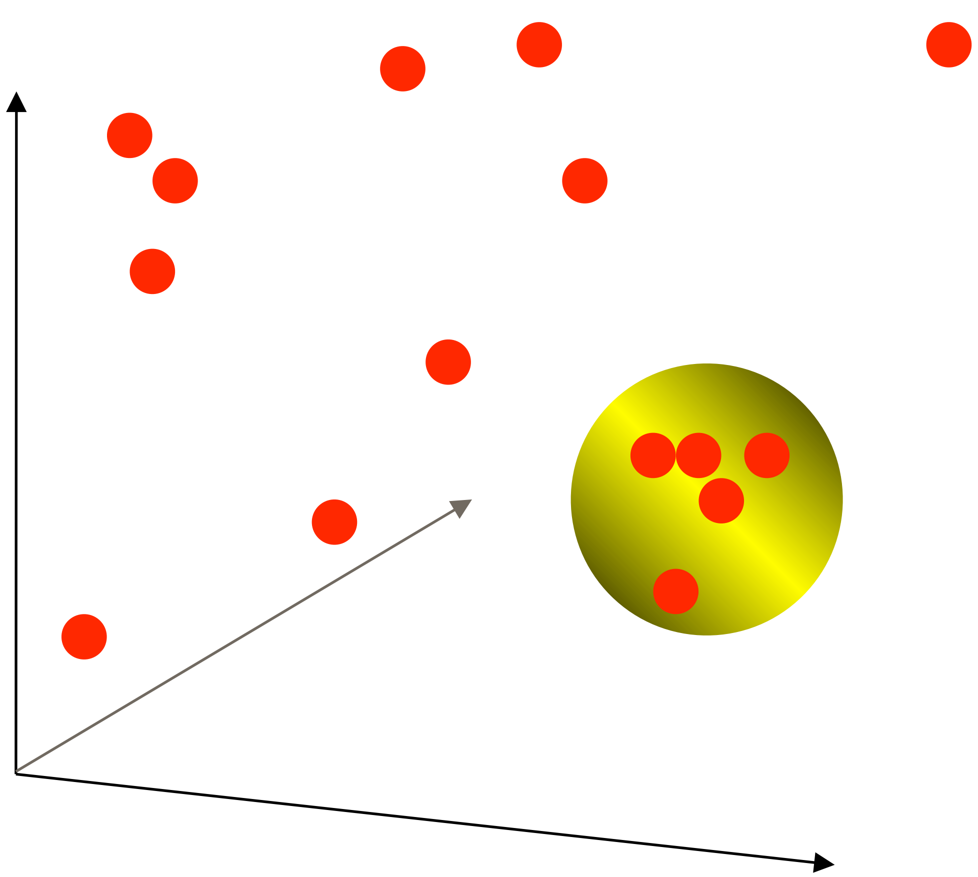
- Plot all word types in k-dimensional space
- Look for **clusters** of close-together types



Learning Classes by Clustering

- Plot all word types in k-dimensional space
- Look for **clusters** of close-together types

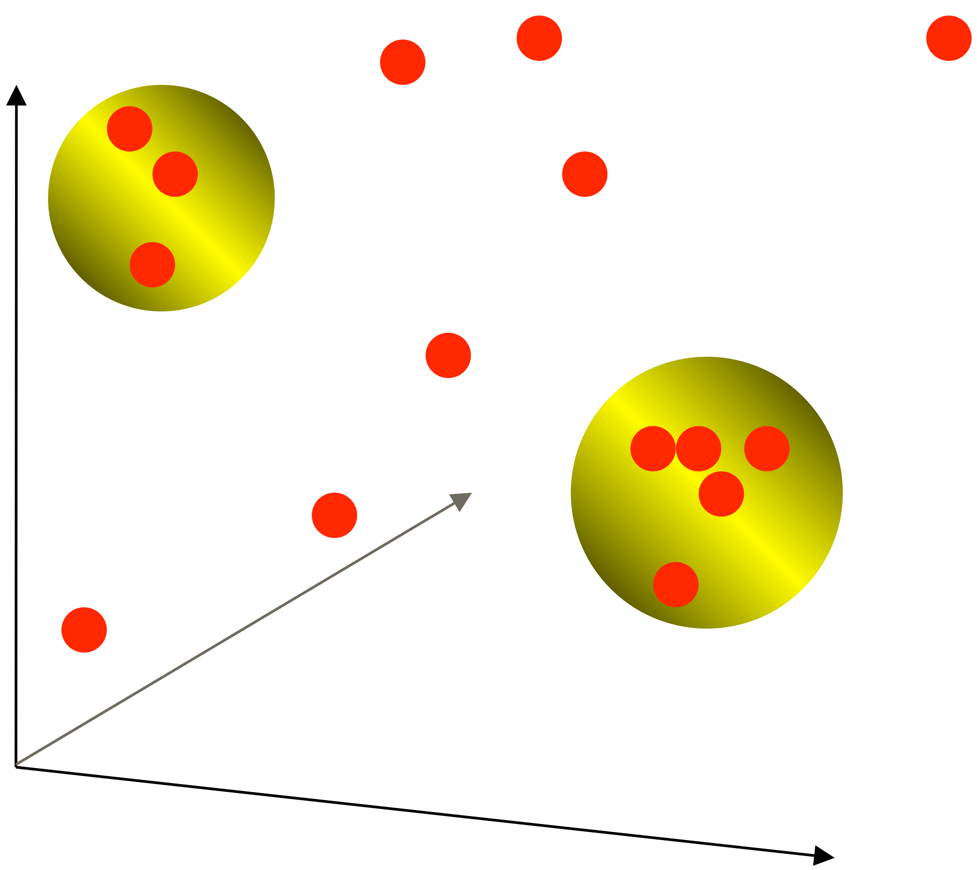
Plot in k dimensions (here k=3)



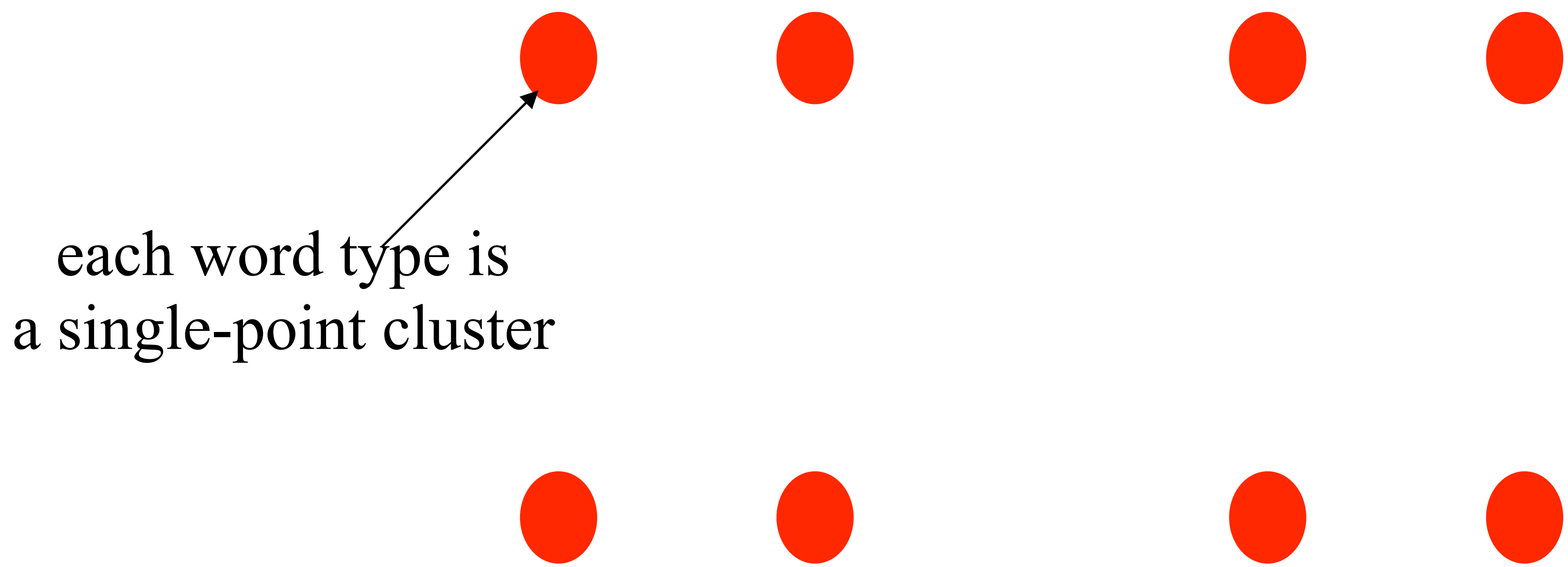
Learning Classes by Clustering

- Plot all word types in k-dimensional space
- Look for **clusters** of close-together types

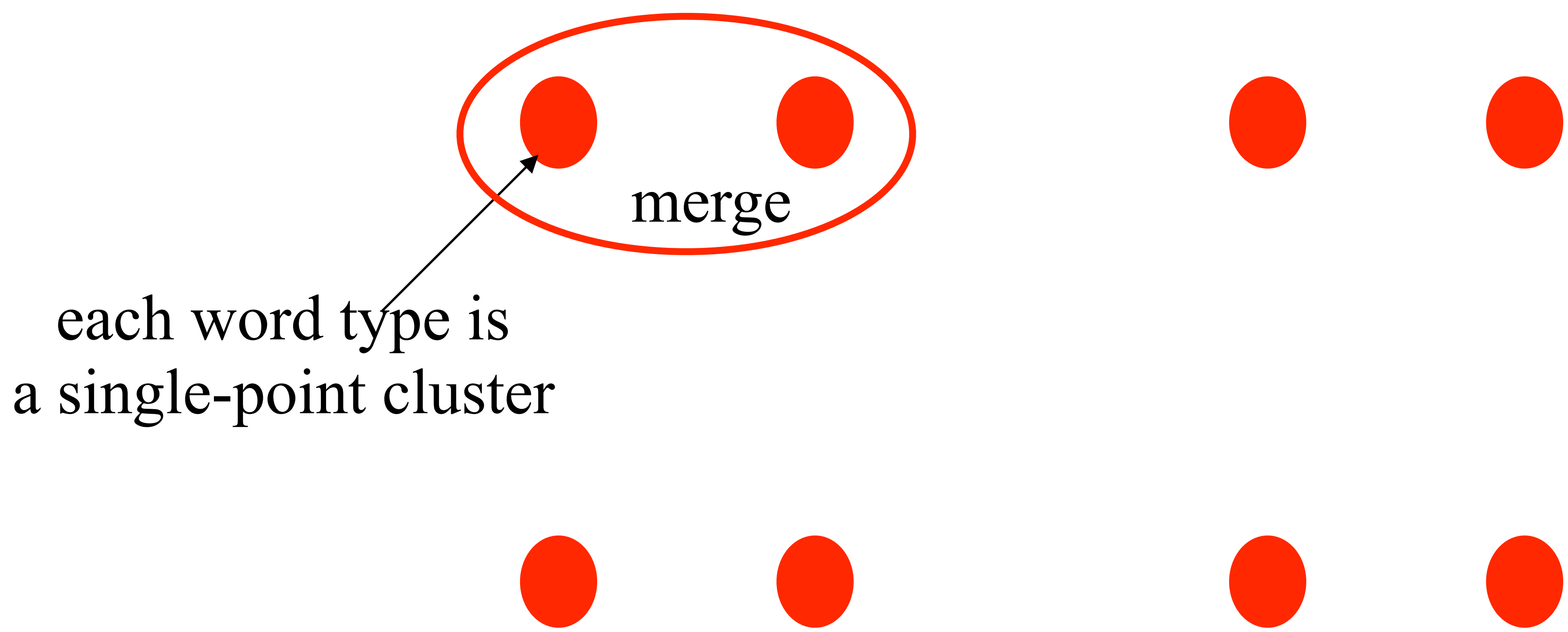
Plot in k dimensions (here k=3)



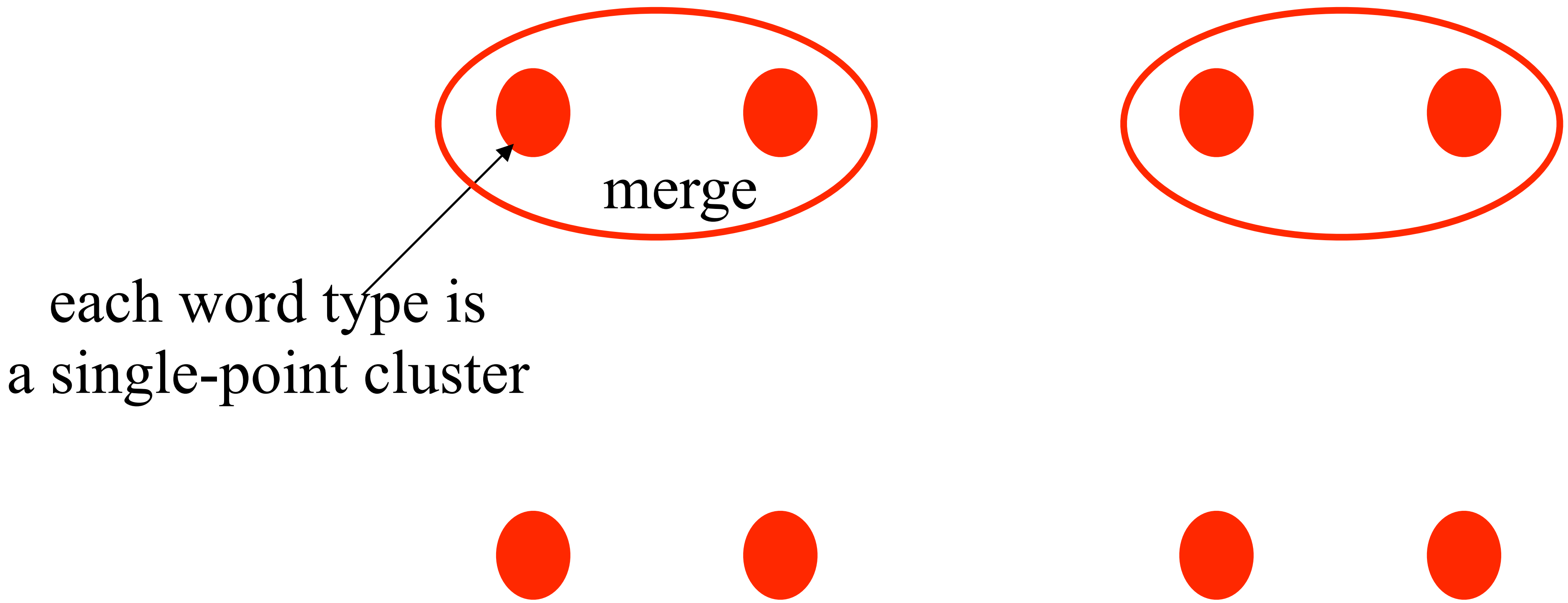
Bottom-Up Clustering – Single-Link



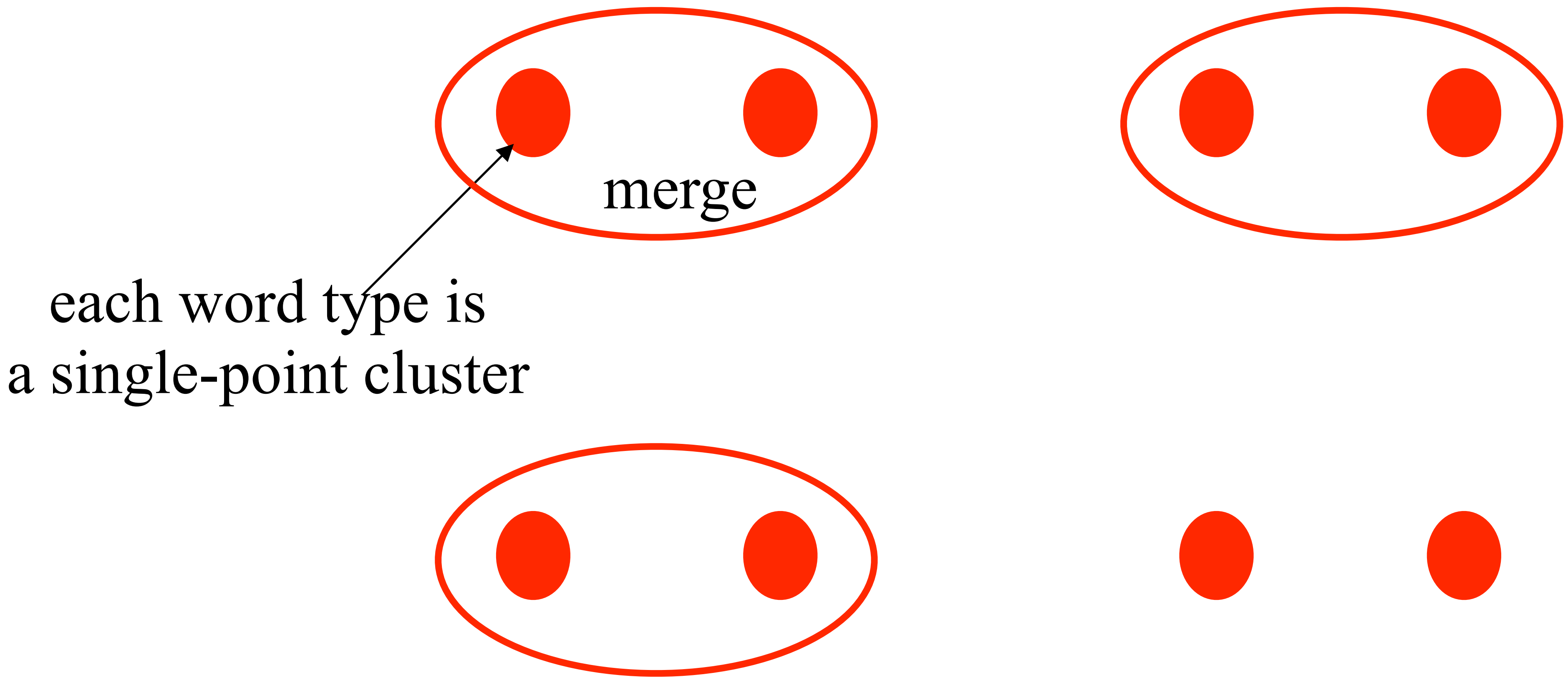
Bottom-Up Clustering – Single-Link



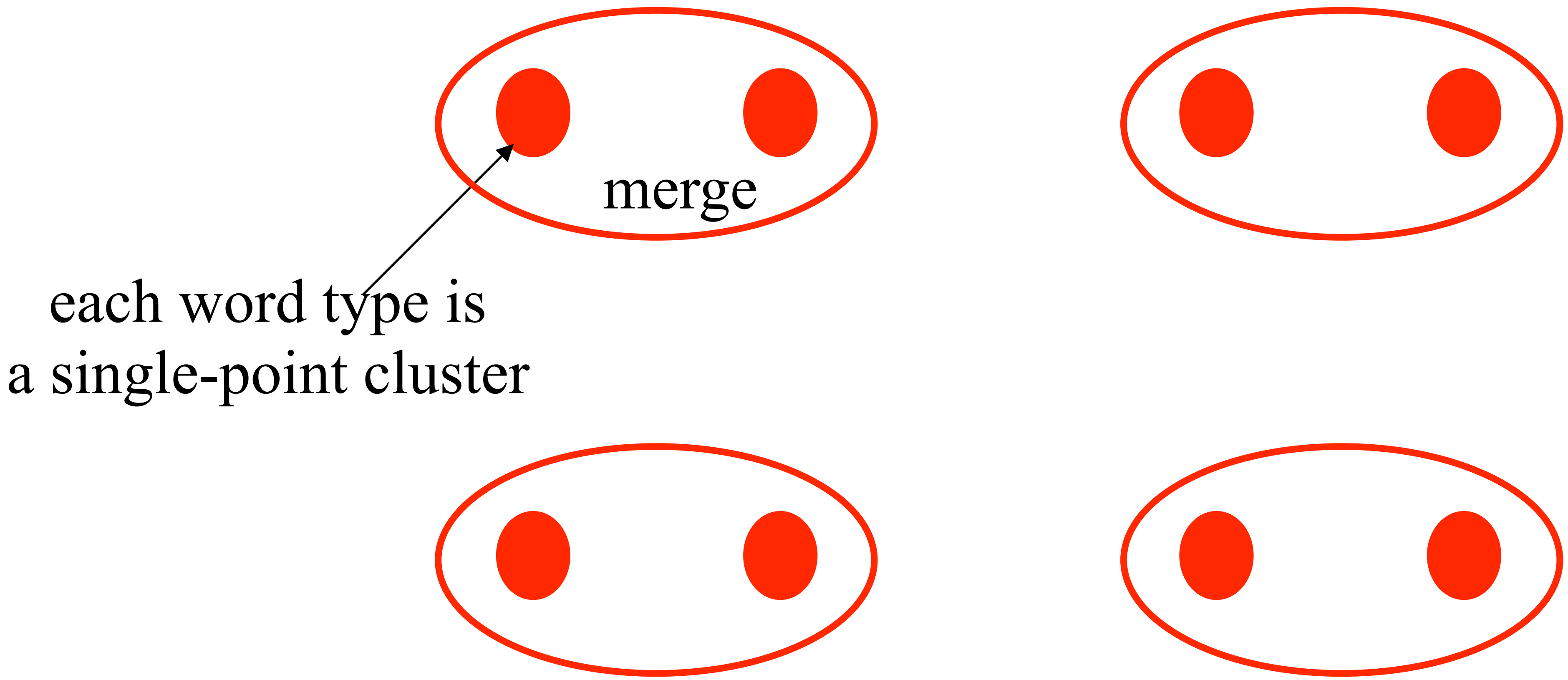
Bottom-Up Clustering – Single-Link



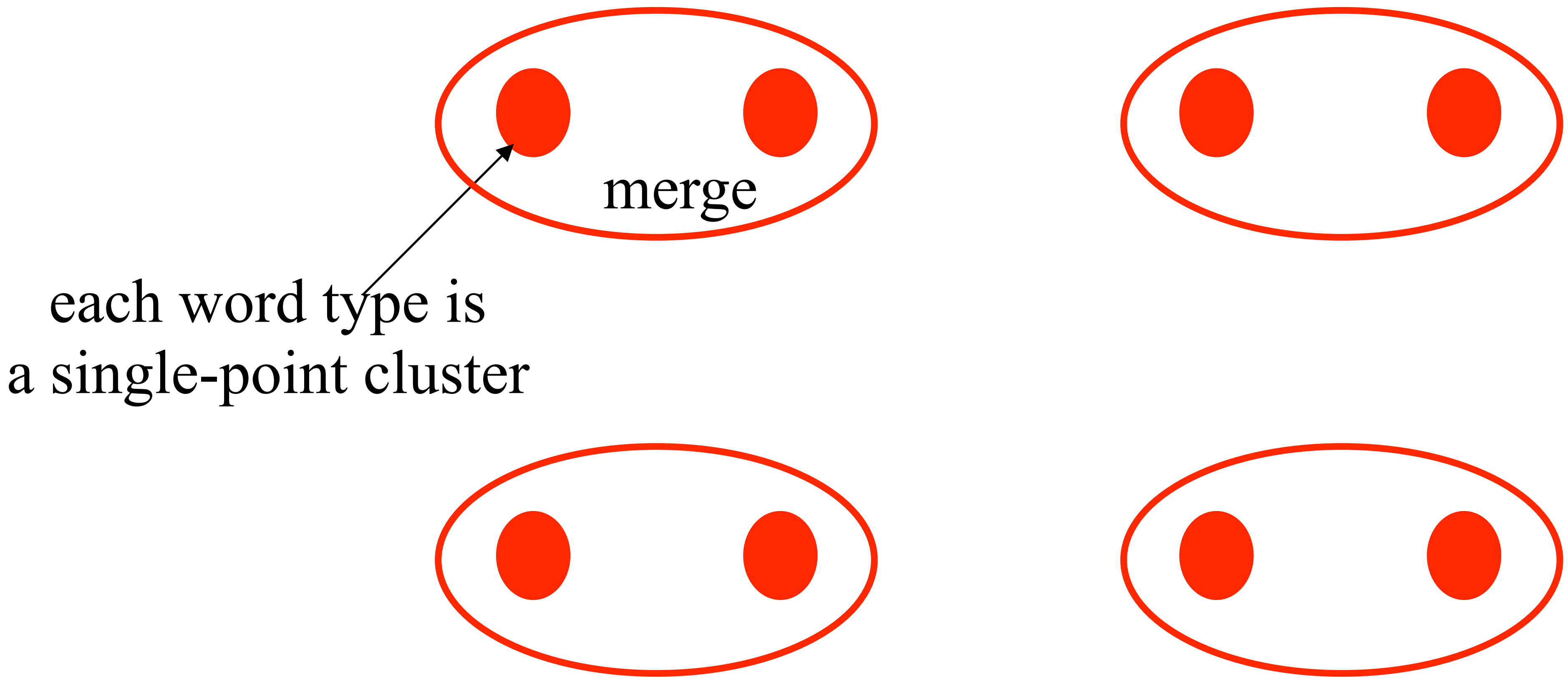
Bottom-Up Clustering – Single-Link



Bottom-Up Clustering – Single-Link



Bottom-Up Clustering – Single-Link



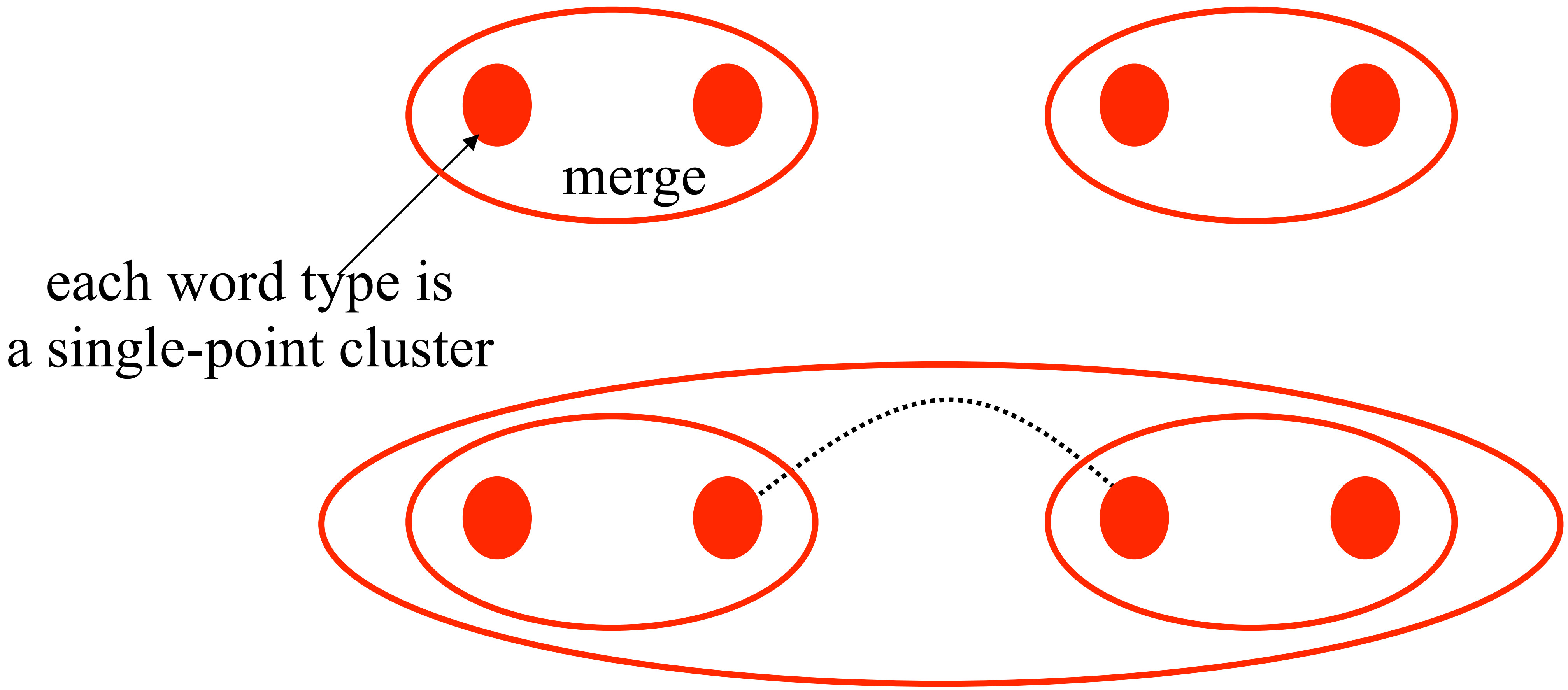
Again, merge closest pair of clusters:

Single-link: clusters are close if any of their points are

dist(A,B) = min dist(a,b) for a∈A, b∈B

example from Manning & Schütze

Bottom-Up Clustering – Single-Link



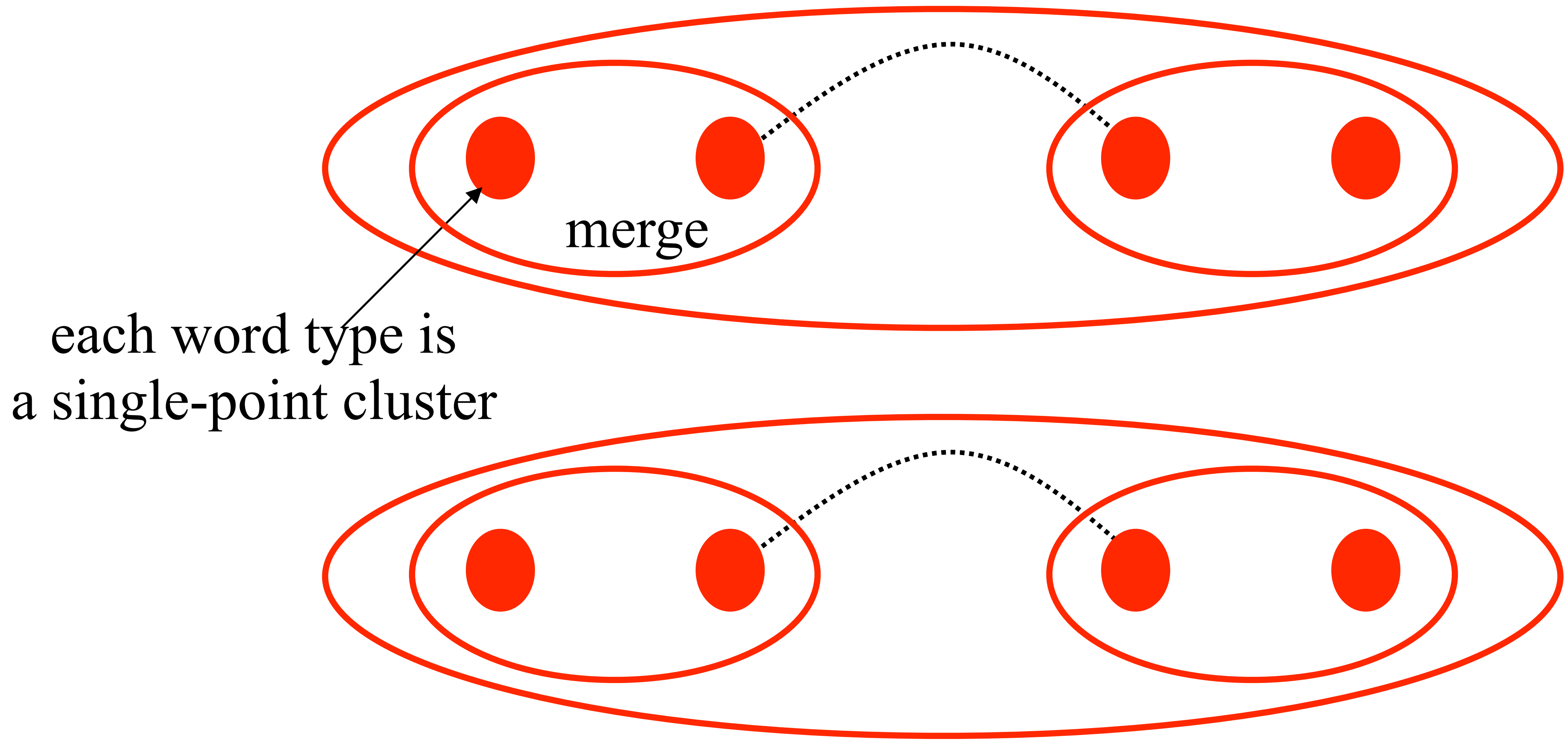
Again, merge closest pair of clusters:

Single-link: clusters are close if any of their points are

dist(A,B) = min dist(a,b) for a∈A, b∈B

example from Manning & Schütze

Bottom-Up Clustering – Single-Link



Again, merge closest pair of clusters:

Single-link: clusters are close if any of their points are

$\text{dist}(A,B) = \min \text{dist}(a,b)$ for $a \in A$, $b \in B$

Bottom-Up Clustering

- Start with one cluster per point
- Repeatedly merge 2 closest clusters
  - Single-link: dist(A,B) = min dist(a,b) for $a \in A$, $b \in B$
  - Complete-link: dist(A,B) = max dist(a,b) for $a \in A$, $b \in B$
    - too slow to update cluster distances after each merge; but ∃ alternatives!

Bottom-Up Clustering

- Start with one cluster per point
- Repeatedly merge 2 closest clusters
  - Single-link: dist(A,B) = min dist(a,b) for a∈A, b∈B
  - Complete-link: dist(A,B) = max dist(a,b) for a∈A, b∈B
    - too slow to update cluster distances after each merge; but ∃ alternatives!
  - Average-link: dist(A,B) = mean dist(a,b) for a∈A, b∈B
  - Centroid-link: dist(A,B) = dist(mean(A),mean(B))

Bottom-Up Clustering

- Start with one cluster per point
- Repeatedly merge 2 closest clusters
  - Single-link: dist(A,B) = min dist(a,b) for $a \in A$, $b \in B$
  - Complete-link: dist(A,B) = max dist(a,b) for $a \in A$, $b \in B$
    - too slow to update cluster distances after each merge; but ∃ alternatives!
  - Average-link: dist(A,B) = mean dist(a,b) for $a \in A$, $b \in B$
  - Centroid-link: dist(A,B) = dist(mean(A),mean(B))
- Stop when clusters are "big enough"
  - e.g., provide adequate support for backoff (on a development corpus)

Bottom-Up Clustering

- Start with one cluster per point
- Repeatedly merge 2 closest clusters
  - Single-link: dist(A,B) = min dist(a,b) for a∈A, b∈B
  - Complete-link: dist(A,B) = max dist(a,b) for a∈A, b∈B
    - too slow to update cluster distances after each merge; but ∃ alternatives!
  - Average-link: dist(A,B) = mean dist(a,b) for a∈A, b∈B
  - Centroid-link: dist(A,B) = dist(mean(A),mean(B))
- Stop when clusters are "big enough"
  - e.g., provide adequate support for backoff (on a development corpus)
- Some flexibility in defining dist(a,b)
  - Might not be Euclidean distance; e.g., use vector angle

EM Clustering (for k clusters)

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called "k-means clustering"
  - Full EM version – called "Gaussian mixtures"

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called “k-means clustering”
  - Full EM version – called “Gaussian mixtures”

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called "k-means clustering"
  - Full EM version – called "Gaussian mixtures"

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called "k-means clustering"
  - Full EM version – called "Gaussian mixtures"

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure
- **Maximization step:** Use that hidden structure (and observations) to reestimate parameters

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called "k-means clustering"
  - Full EM version – called "Gaussian mixtures"

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure
- **Maximization step:** Use that hidden structure (and observations) to reestimate parameters

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called "k-means clustering"
  - Full EM version – called "Gaussian mixtures"

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure
- **Maximization step:** Use that hidden structure (and observations) to reestimate parameters

- Parameters: k points representing cluster centers

EM Clustering (for k clusters)

- EM algorithm
  - Viterbi version – called “k-means clustering”
  - Full EM version – called “Gaussian mixtures”

- **Expectation step:** Use current parameters (and observations) to reconstruct hidden structure
- **Maximization step:** Use that hidden structure (and observations) to reestimate parameters

- Parameters: k points representing cluster centers
- Hidden structure: for each data point (word type), which center generated it?

Lexical translation

- How to translate a word → look up in dictionary

  Haus — *house, building, home, household, shell.*

- *Multiple translations*
  - some more frequent than others
  - for instance: *house*, and *building* most common
  - special cases: *Haus* of a *snail* is its *shell*

- Note: During all the lectures, we will translate from a foreign language into English

Collect statistics

- Look at a *parallel corpus* (German text along with English translation)

Translation of *Haus*	**Count**
house	8,000
building	1,600
home	200
household	150
shell	50

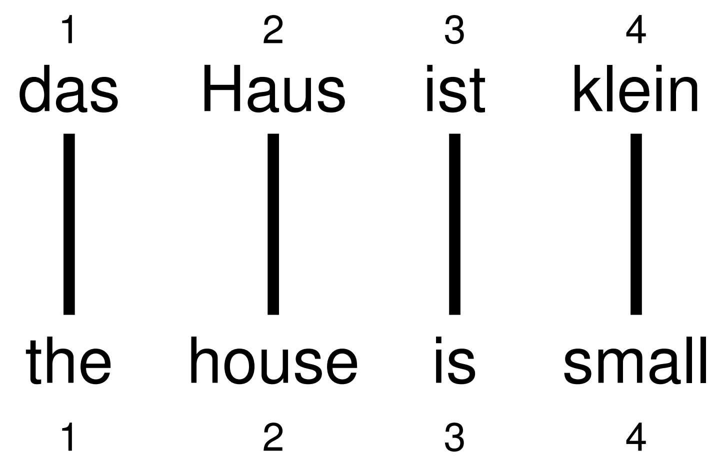
Estimate translation probabilities

- *Maximum likelihood estimation*

$$p_f(e) = \begin{cases} 0.8 & \text{if } e = \textit{house}, \\ 0.16 & \text{if } e = \textit{building}, \\ 0.02 & \text{if } e = \textit{home}, \\ 0.015 & \text{if } e = \textit{household}, \\ 0.005 & \text{if } e = \textit{shell}. \end{cases}$$

Alignment

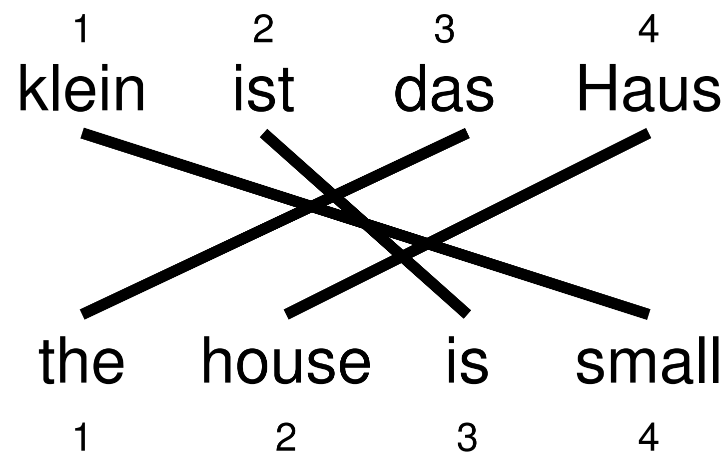
- In a parallel text (or when we translate), we **align** words in one language with the words in the other



- Word *positions* are numbered 1–4

Reordering

- Words may be **reordered** during translation



$$a : \{1 \rightarrow 3, 2 \rightarrow 4, 3 \rightarrow 2, 4 \rightarrow 1\}$$

IBM Model 1

- *Generative model*: break up translation process into smaller steps
  - **IBM Model 1** only uses *lexical translation*

- Translation probability
  - for a foreign sentence $\mathbf{f} = (f_1, ..., f_{l_f})$ of length l_f
  - to an English sentence $\mathbf{e} = (e_1, ..., e_{l_e})$ of length l_e
  - with an alignment of each English word e_j to a foreign word f_i according to the alignment function $a : j \rightarrow i$

$$p(\mathbf{e}, a|\mathbf{f}) = \frac{\epsilon}{(l_f + 1)^{l_e}} \prod_{j=1}^{l_e} t(e_j | f_{a(j)})$$

  - parameter ϵ is a *normalization constant*

Example

das	
e	$t(e\|f)$
the	0.7
that	0.15
which	0.075
who	0.05
this	0.025

Haus	
e	$t(e\|f)$
house	0.8
building	0.16
home	0.02
household	0.015
shell	0.005

ist	
e	$t(e\|f)$
is	0.8
's	0.16
exists	0.02
has	0.015
are	0.005

klein	
e	$t(e\|f)$
small	0.4
little	0.4
short	0.1
minor	0.06
petty	0.04

$$\begin{aligned} p(e,a|f) &= \frac{\epsilon}{4^3} \times t(\text{the}|\text{das}) \times t(\text{house}|\text{Haus}) \times t(\text{is}|\text{ist}) \times t(\text{small}|\text{klein}) \\ &= \frac{\epsilon}{4^3} \times 0.7 \times 0.8 \times 0.8 \times 0.4 \\ &= 0.0028\epsilon \end{aligned}$$

Learning lexical translation models

- We would like to *estimate* the lexical translation probabilities $t(e|f)$ from a parallel corpus

- ... but we do not have the alignments

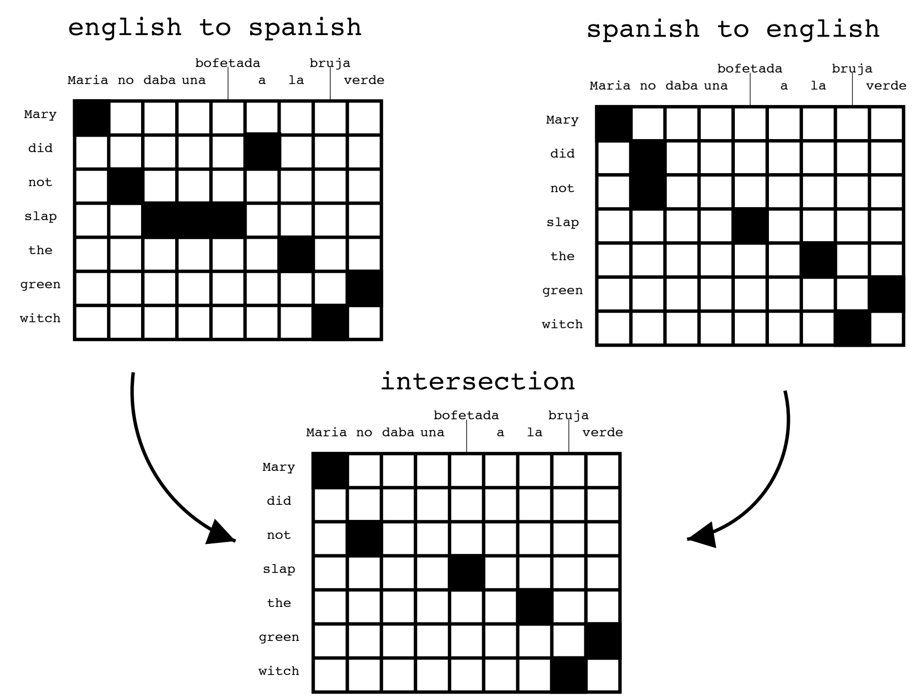
- **Chicken and egg problem**
  - if we had the *alignments*,
    $\rightarrow$ we could estimate the *parameters* of our generative model
  - if we had the *parameters*,
    $\rightarrow$ we could estimate the *alignments*

EM algorithm

- **Incomplete data**
  - if we had *complete data*, would could estimate *model*
  - if we had *model*, we could fill in the *gaps in the data*

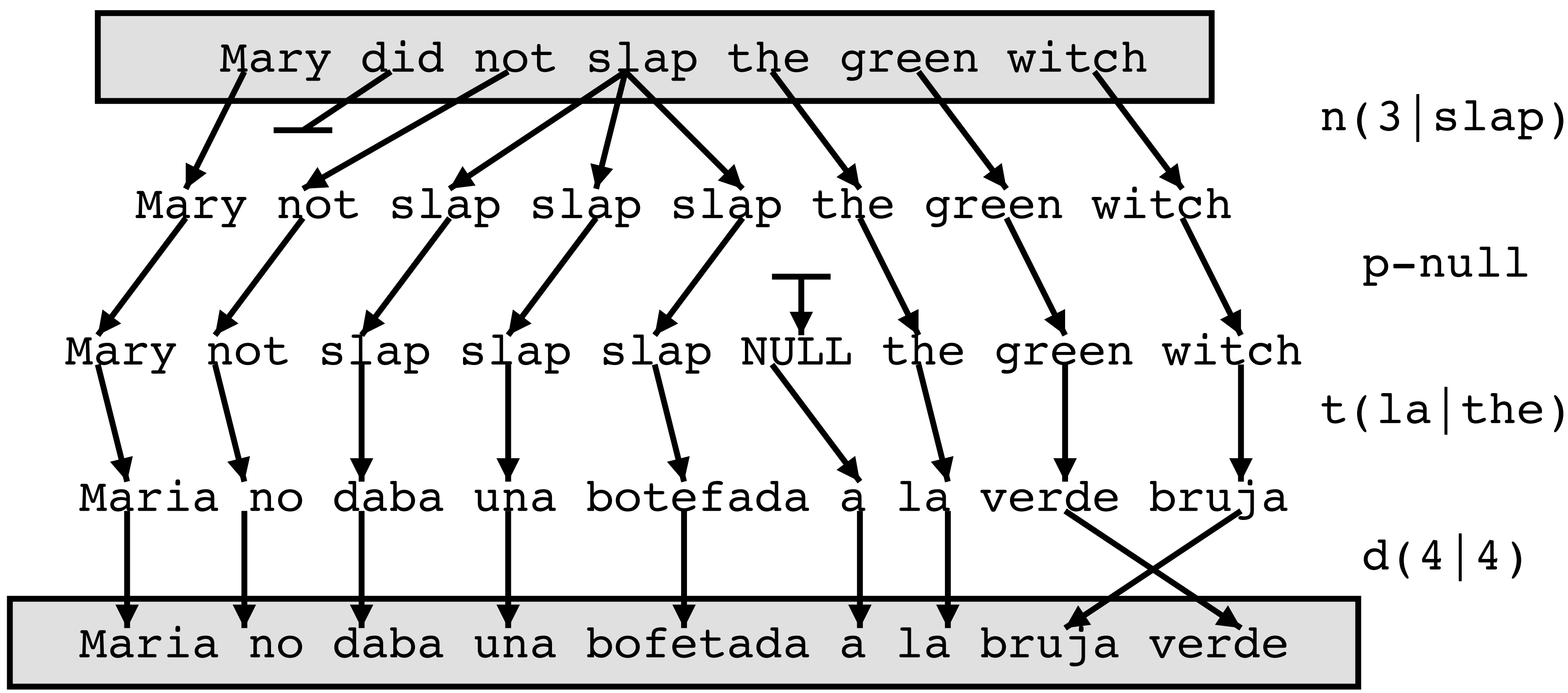
- **Expectation Maximization (EM)** in a nutshell
  - initialize model parameters (e.g. uniform)
  - assign probabilities to the missing data
  - estimate model parameters from completed data
  - iterate

Symmetrizing word alignments



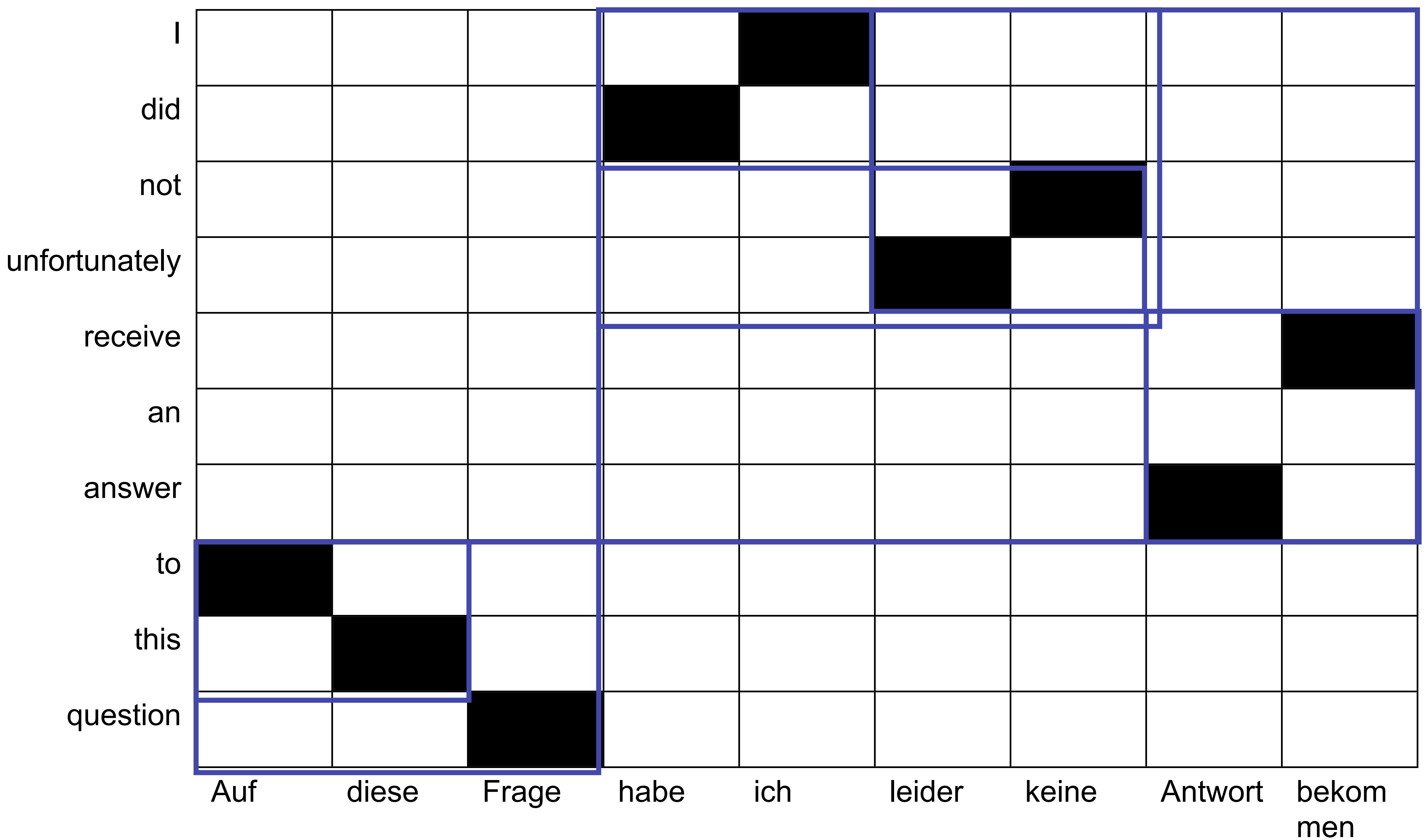
- *Intersection* of GIZA++ bidirectional alignments

IBM Model 4



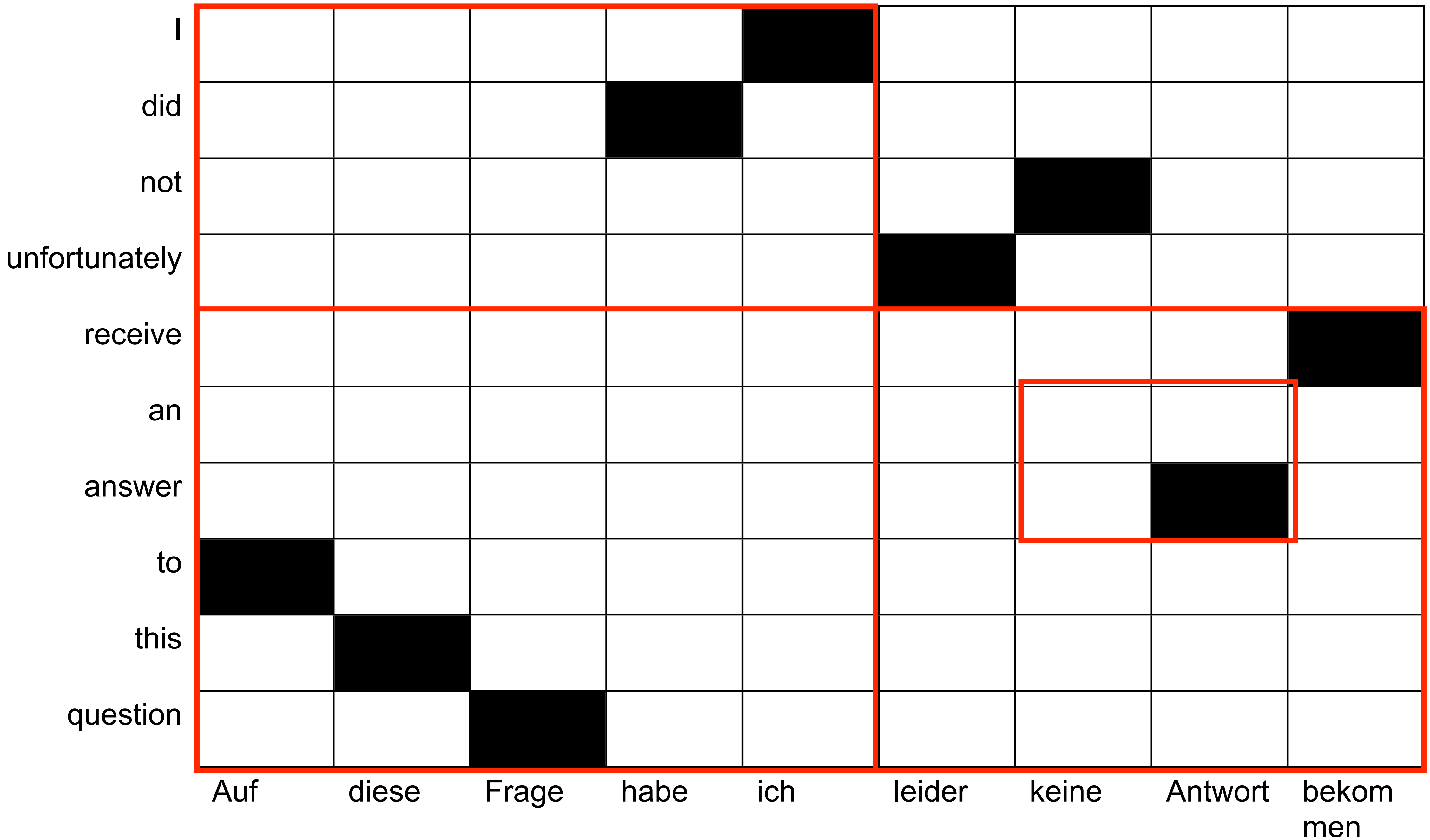
Mary did not slap the green witch
Mary not slap slap slap the green witch
Mary not slap slap slap NULL the green witch
Maria no daba una botefada a la verde bruja
Maria no daba una bofetada a la bruja verde
n(3|slap)
p-null
t(la|the)
d(4|4)

Phrase Models



Some good phrase pairs.

Phrase Models

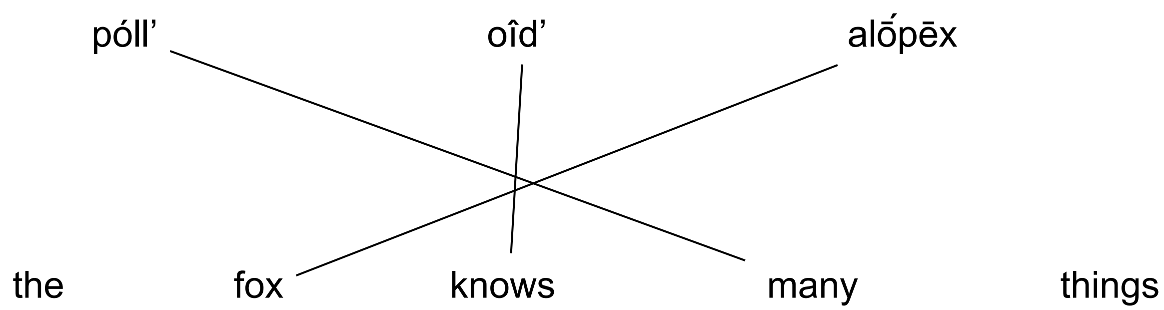


Some bad phrase pairs.

Synchronous Grammars

- Just like monolingual grammars except...
  - Each rule involves pairs (tuples) of nonterminals
  - Tuples of elementary trees for TAG, etc.
- First proposed for source-source translation in compilers
- Can be constituency, dependency, lexicalized, etc.
- Parsing speedups for monolingual grammar don't necessarily work
  - E.g., no split-head trick for lexicalized parsing
- Binarization less straightforward

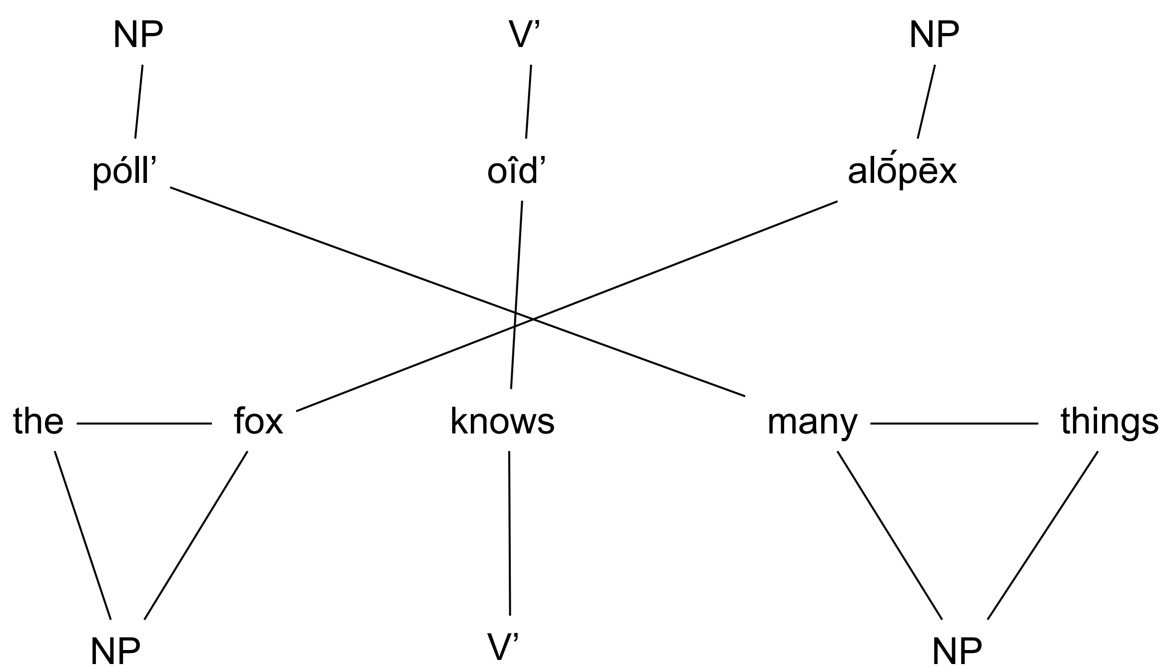
Bilingual Parsing



	póll’	oîd’	alṓpēx
the			
fox			NN/NN
knows		VB/VB	
many	JJ/JJ		
things			

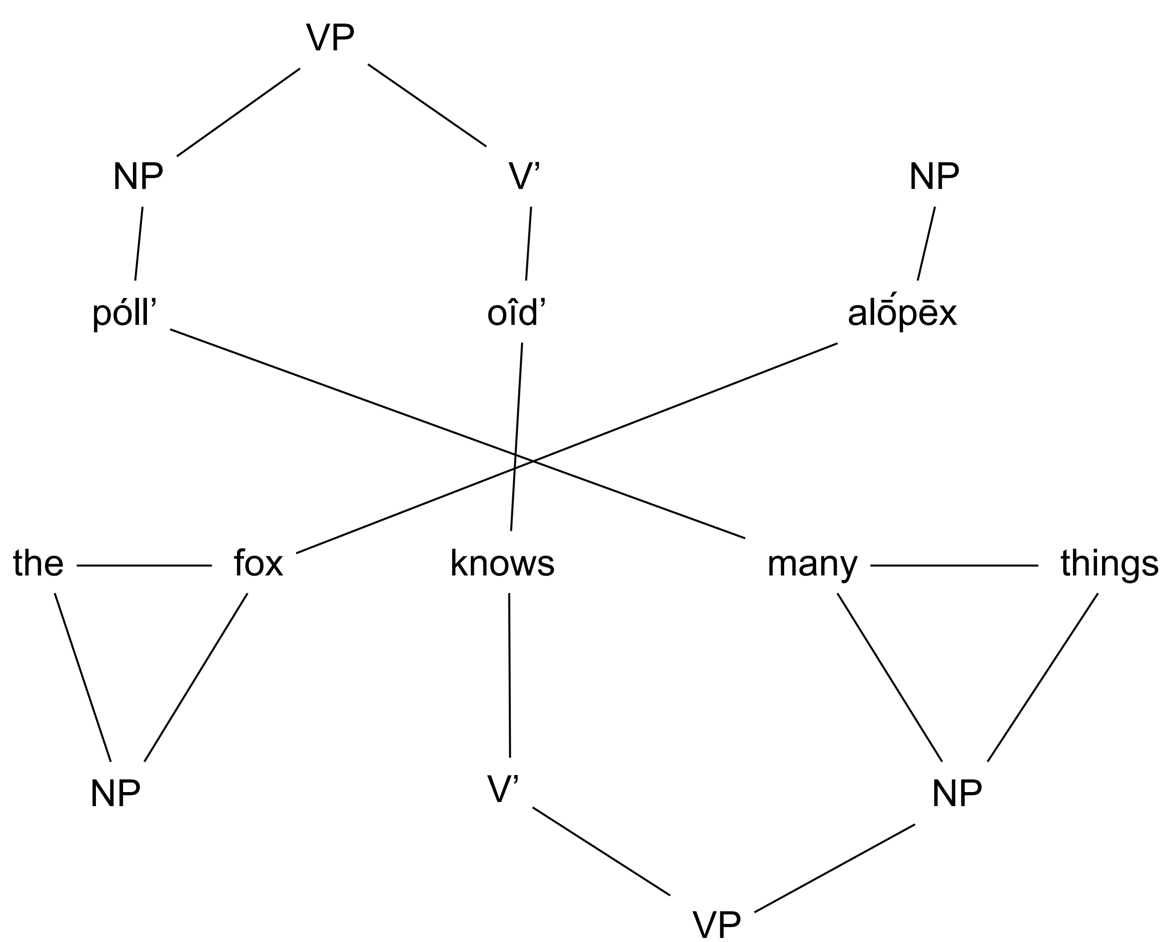
A variant of CKY chart parsing.

Bilingual Parsing



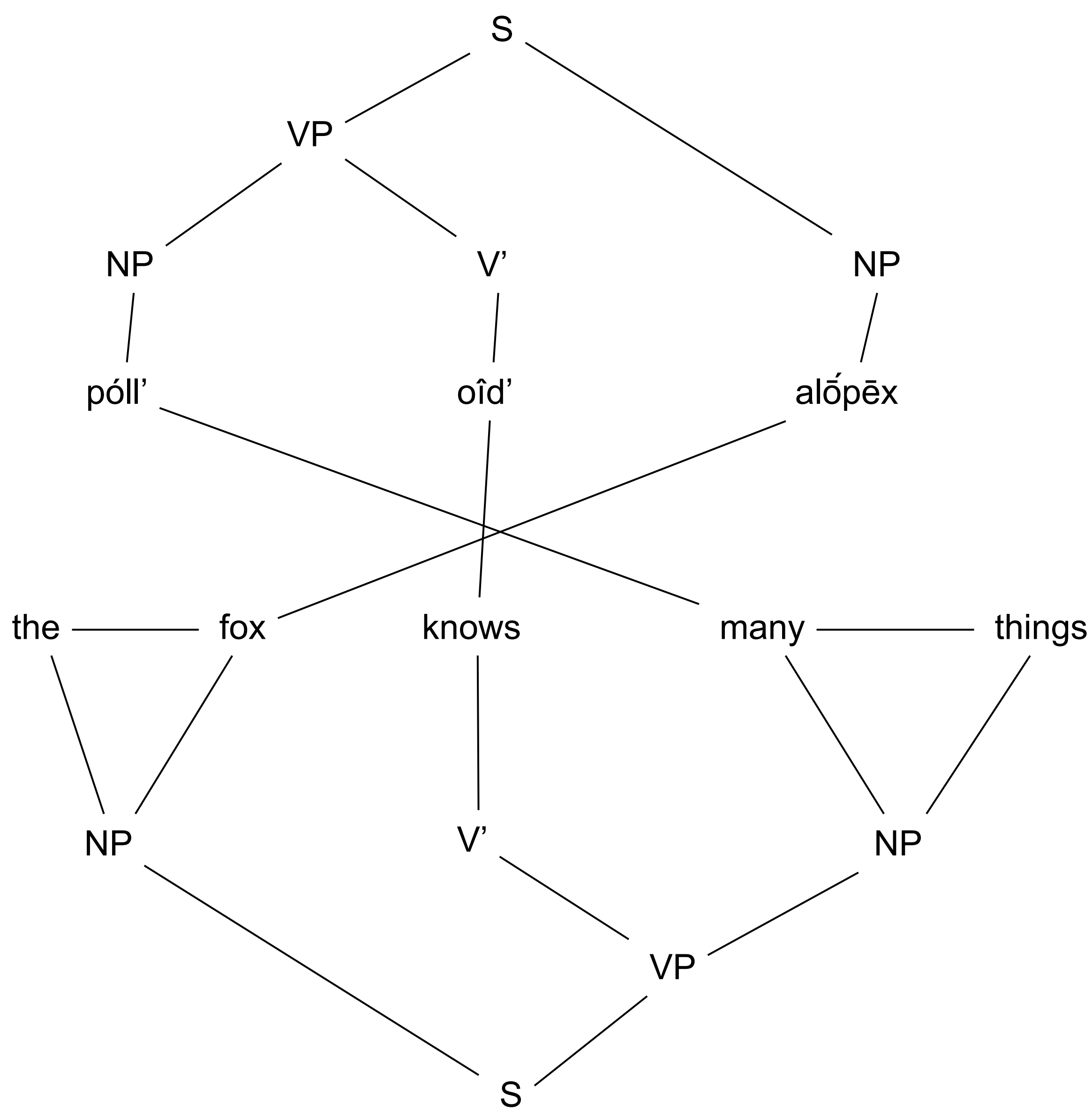
<table>
<tr><th></th><th>póll’</th><th>oîd’</th><th>alṓpēx</th></tr>
<tr><td>the</td><td></td><td></td><td rowspan="2">NP/NP</td></tr>
<tr><td>fox</td><td></td><td></td></tr>
<tr><td>knows</td><td></td><td>VP/VP</td><td></td></tr>
<tr><td>many</td><td rowspan="2">NP/NP</td><td></td><td></td></tr>
<tr><td>things</td><td></td><td></td></tr>
</table>

Bilingual Parsing



<table>
<tr><th></th><th>póll’</th><th>oîd’</th><th>alṓpēx</th></tr>
<tr><td>the</td><td></td><td></td><td rowspan="2">NP/NP</td></tr>
<tr><td>fox</td><td></td><td></td></tr>
<tr><td>knows</td><td colspan="2" rowspan="3">VP/VP</td><td></td></tr>
<tr><td>many</td><td></td></tr>
<tr><td>things</td><td></td></tr>
</table>

Bilingual Parsing



	póll’	oîd’	alṓpēx
the			
fox			
knows		S/S	
many			
things			