Markov Language Models

CS6120: Natural Language Processing
Northeastern University

David Smith

A **language model**
is a function that
assigns a probability to
a string of text.

Finite LMs

The quick brown fox jumped over the lazy dog

Finite LMs

S = {The quick brown fox jumped over the lazy dog}

$$P(s) = 1 \text{ if } s \in S$$
$$P(s) = 0 \text{ otherwise}$$

Defining **languages as sets**
in your theory of computation course

Finie LMs

S = {
  The quick brown fox…,
  When in the course of human events…,
  It was a bright cold day in April and the clocks…
}

Finite LMs

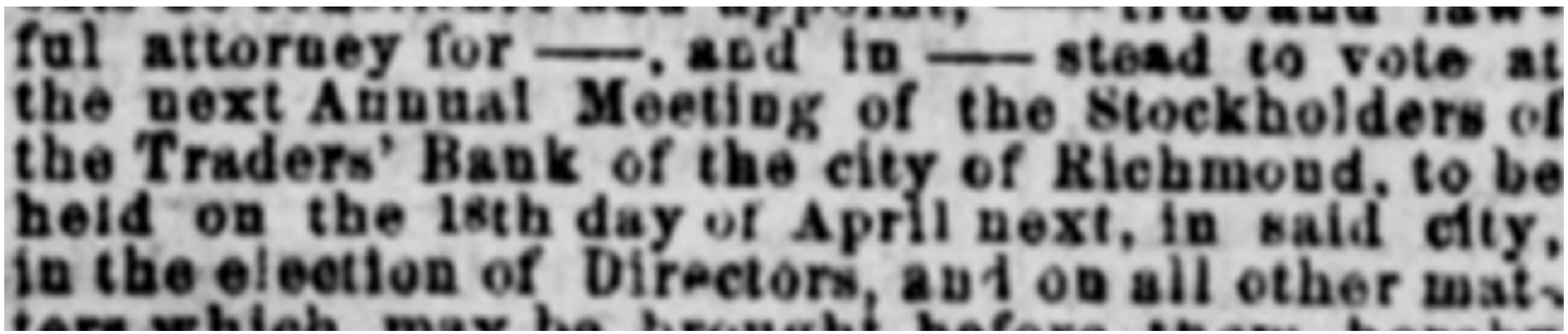
S = {
  The quick brown fox…,
  When in the course of human events…,
  It was a bright cold day in April and the clocks…
}

$$P(s) = \frac{1}{|S|} \text{ if } s \in S$$
$$P(s) = 0 \text{ otherwise}$$

This works for finite sets

Strings as Queries

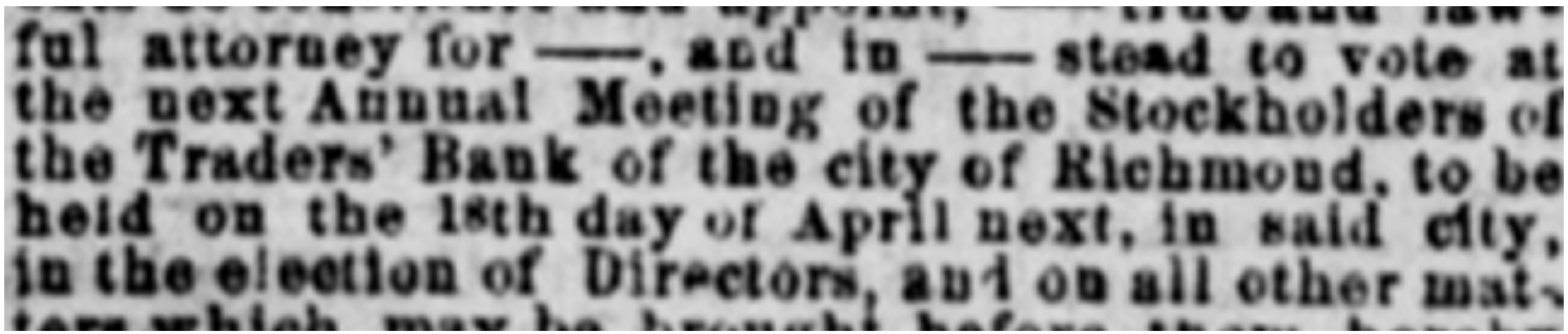
You’re looking at old financial notices:


ful attorney for ——, and in —— stead to vote at
the next Annual Meeting of the Stockholders of
the Traders' Bank of the city of Richmond, to be
held on the 18th day of April next, in said city,
in the election of Directors, and on all other mat-

searching for:

`the Traders’ Bank of the city of Richmond`

Strings as Queries


ful attorney for ——, and in —— stead to vote at
the next Annual Meeting of the Stockholders of
the Traders' Bank of the city of Richmond, to be
held on the 18th day of April next, in said city,
in the election of Directors, and on all other mat-

But these lines get transcribed as:

```
the Trader. Bank of the city of Richmoud, to be
tbe Traders' Bank or the city of Biebmond, to bo
tbe Traders' Bank of the city of Klchmoud, to be,
the Traders' Hank of the city of Richmoud, lo be j
the Trader*' Bsnk of the city of Richmond, to be
the Traders' Hank of the city of Richmond, to he
tha Traders' Bank of the cltv of Richmond to be
```

Exact match won't work! Goodbye, Knuth-Morris-Pratt, etc.

Generalized Queries

Notice confusion of c/e/o, b/h, B/H/K/R:

```
the Trader. Bank of the city of Richmoud, to be
tbe Traders' Bank or the city of Biebmond, to bo
tbe Traders' Bank of the city of Klchmoud, to be,
the Traders' Hank of the city of Richmoud, lo be j
the Trader*' Bsnk of the city of Richmond, to be
the Traders' Hank of the city of Richmond, to he
tha Traders' Bank of the cltv of Richmond to be
```

Instead of searching for:

```
the Traders' Bank of the city of Richmond
```

Try this:

```
t[bh][ceo] Trad[ceo]rs' [BHKR]ank o[fr]
th[ceo] [ceo]ity [ceo][fr] [BHKR]i[ceo]
[bh]m[ceo]nd
```

Generalized Queries

Try this:

`t[bh][ceo] Trad[ceo]rs’ [BHKR]ank o[fr] th[ceo] [ceo]ity [ceo][fr] [BHKR]i[ceo] [bh]m[ceo]nd`

Which would match two of them:

```
the Trader. Bank of the city of Richmoud, to be
tbe Traders' Bank or the city of Biebmond, to bo
tbe Traders' Bank of the city of Klchmoud, to be,
the Traders' Hank of the city of Richmoud, lo be j
the Trader*' Bsnk of the city of Richmond, to be
the Traders' Hank of the city of Richmond, to he
tha Traders' Bank of the cltv of Richmond to be
```

Regular Languages

S = {
  ha,
  haha,
  hahaha,
  hahahaha,
  …
}

Regular Languages

S = {
ha,
haha,
hahaha,
hahahaha,
…
}

Regular expression `(ha)+`

Regular Languages

S = {
ha,
haha,
hahaha,
hahahaha,
…
}

Regular expression `(ha)+`

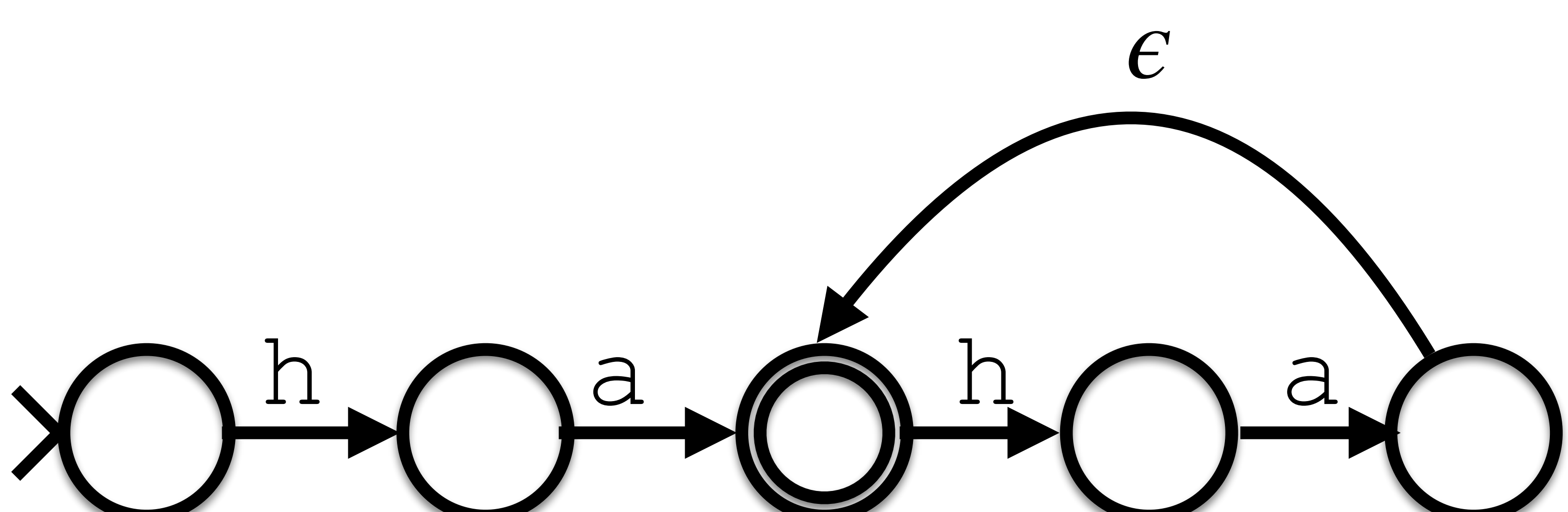
Syntactic sugar for `ha(ha)*`

Regular Languages

S = {
ha,
haha,
hahaha,
hahahaha,
…
}

Regular expression `(ha)+`

Syntactic sugar for `ha(ha)*`



Regular Languages

- Closed under:
  - Concatenation, e.g., `the`
  - Union, `(this)|(that)`, `[aeiou]`
    - Many regexes have syntactic sugar for unions like \w, \s, \d, \p{Greek}, etc.
  - Kleene star, e.g., `(ha)*`, `(ha)+`
  - Intersection, reversal, complement, and other operations not implemented in most regular expressions

Regular Languages

But this regular language

```
t[bh][ceo] Trad[ceo]rs' [BHKR]ank o[fr]
th[ceo] [ceo]ity [ceo][fr] [BHKR]i[ceo]
[bh]m[ceo]nd
```

weights each of the

2*3*3*4*2*3*3*3*2*4*3*2*3=559,872

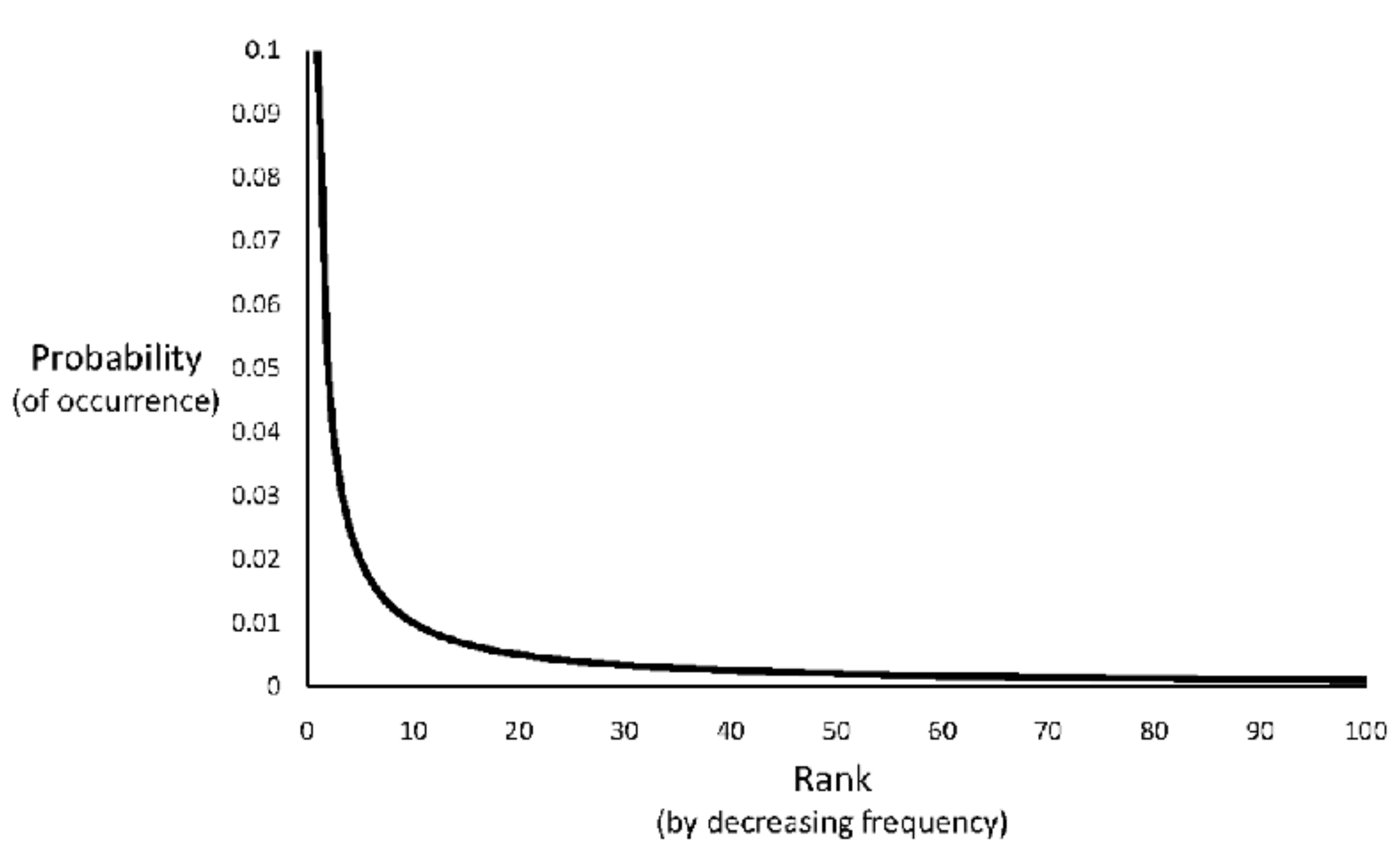
strings in the language equally.

Surely some strings are more likely!

Zipf's Law

- Distribution of word frequencies is very *skewed*
  - a few words occur very often, many words hardly ever occur
  - e.g., two most common words ("the", "of") make up about 10% of all word occurrences in text documents
- Zipf's law (more generally, a "power law"):
  - observation that rank (*r*) of a word times its frequency (*f*) is approximately a constant (*k*)
  - assuming words are ranked in order of decreasing frequency
  - i.e., $r \cdot f \approx k$ or $r \cdot P_r \approx c$, where P_r is relative frequency of word occurrence and $c \approx 0.1$ for English

Zipf's Law



0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Probability
(of occurrence)
0
10
20
30
40
50
60
70
80
90
100
Rank
(by decreasing frequency)

AP89 Example

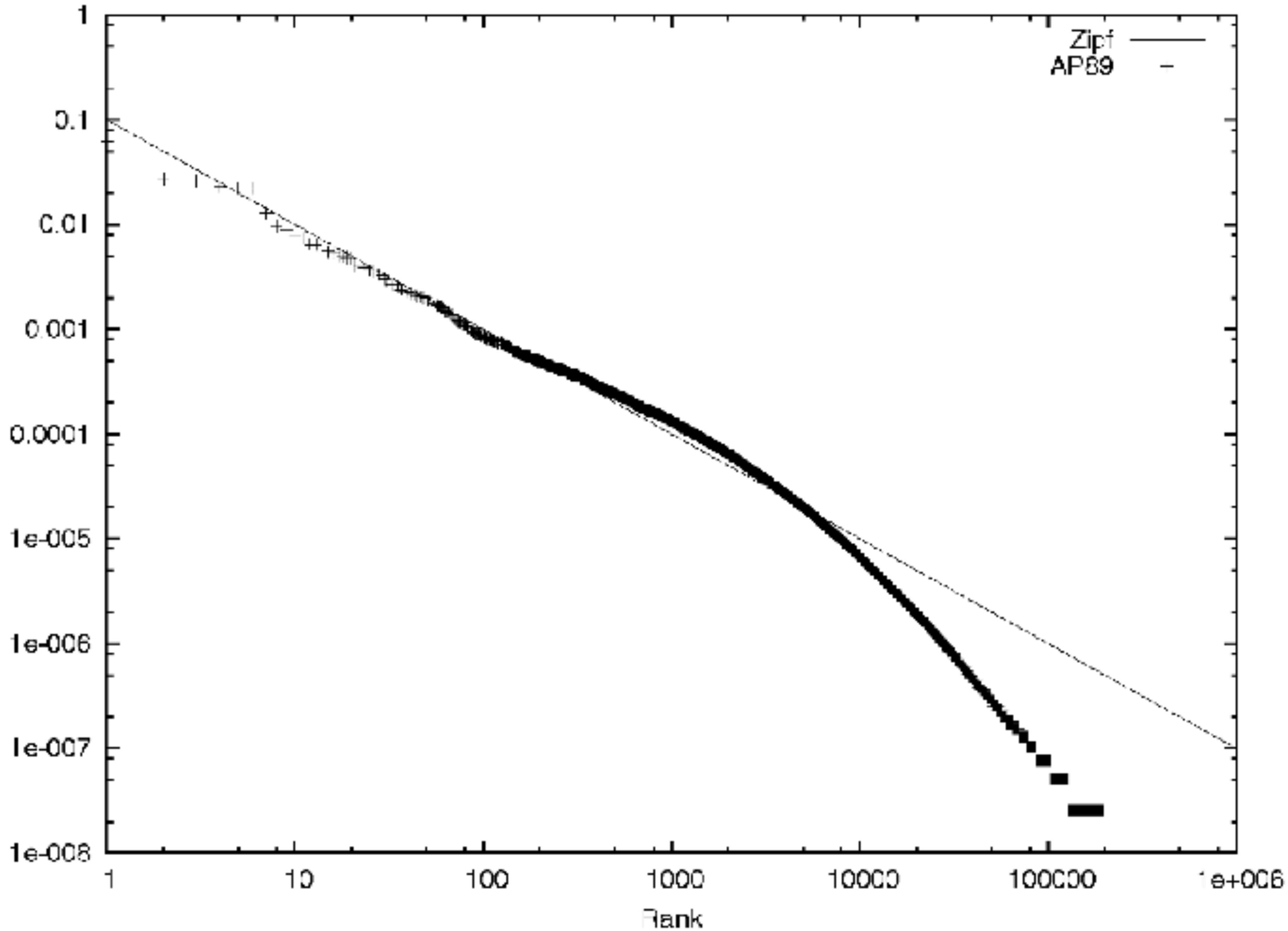
Total documents	84,678
Total word occurrences	39,749,179
Vocabulary size	198,763
Words occurring > 1000 times	4,169
Words occurring once	70,064

Word	*Freq.*	*r*	*Pr(%)*	*r.Pr*
assistant	5,095	1,021	.013	0.13
sewers	100	17,110	$2.56 \times 10{-}4$	0.04
toothbrush	10	51,555	$2.56 \times 10{-}5$	0.01
hazmat	1	166,945	$2.56 \times 10{-}6$	0.04

Top 50 Words in AP89

Word	Freq.	r	$P_r(\%)$	$r.P_r$	Word	Freq	r	$P_r(\%)$	$r.P_r$
the	2,420,778	1	6.49	0.065	has	136,007	26	0.37	0.095
of	1,045,733	2	2.80	0.056	are	130,322	27	0.35	0.094
to	968,882	3	2.60	0.078	not	127,493	28	0.34	0.096
a	892,429	4	2.39	0.096	who	116,364	29	0.31	0.090
and	865,644	5	2.32	0.120	they	111,024	30	0.30	0.089
in	847,825	6	2.27	0.140	its	111,021	31	0.30	0.092
said	504,593	7	1.35	0.095	had	103,943	32	0.28	0.089
for	363,865	8	0.98	0.078	will	102,949	33	0.28	0.091
that	347,072	9	0.93	0.084	would	99,503	34	0.27	0.091
was	293,027	10	0.79	0.079	about	92,983	35	0.25	0.087
on	291,947	11	0.78	0.086	i	92,005	36	0.25	0.089
he	250,919	12	0.67	0.081	been	88,786	37	0.24	0.088
is	245,843	13	0.65	0.086	this	87,286	38	0.23	0.089
with	223,846	14	0.60	0.084	their	84,638	39	0.23	0.089
at	210,064	15	0.56	0.085	new	83,449	40	0.22	0.090
by	209,586	16	0.56	0.090	or	81,796	41	0.22	0.090
it	195,621	17	0.52	0.089	which	80,385	42	0.22	0.091
from	189,451	18	0.51	0.091	we	80,245	43	0.22	0.093
as	181,714	19	0.49	0.093	more	76,388	44	0.21	0.090
be	157,300	20	0.42	0.084	after	75,165	45	0.20	0.091
were	153,913	21	0.41	0.087	us	72,045	46	0.19	0.089
an	152,576	22	0.41	0.090	percent	71,956	47	0.19	0.091
have	149,749	23	0.40	0.092	up	71,082	48	0.19	0.092
his	142,285	24	0.38	0.092	one	70,266	49	0.19	0.092
but	140,880	25	0.38	0.094	people	68,988	50	0.19	0.093

Zipf's Law for AP89



log–log plot: note deviations at high and low frequencies

Zipf's Law

- What is the proportion of words with a given frequency?
  - Word that occurs n times has rank $r_n = k/n$
  - Number of words with frequency n is
    - $r_n - r_{n+1} = k/n - k/(n+1) = k/n(n+1)$
  - Proportion found by dividing by total number of words = highest rank = k
  - So, proportion with frequency n is $1/n(n+1)$

Zipf Example

Number of Occurrences (n)	*Predicted Proportion (1/n(n+1))*	*Actual Proportion*	*Actual Number of Words*
1	.500	.402	204,357
2	.167	.132	67,082
3	.083	.069	35,083
4	.050	.046	23,271
5	.033	.032	16,332
6	.024	.024	12,421
7	.018	.019	9,766
8	.014	.016	8,200
9	.011	.014	6,907
10	.009	.012	5,893

- Proportions of words occurring n times in 336,310 TREC documents
- Vocabulary size is 508,209

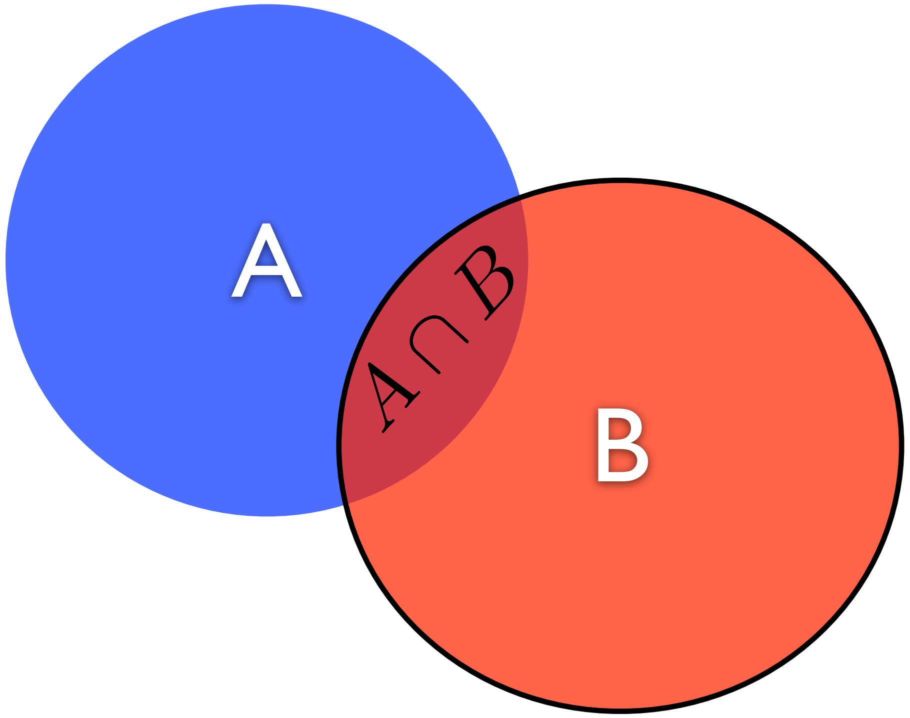
Probability

Axioms of Probability

- Define event space $\bigcup_i \mathcal{F}_i = \Omega$
- Probability function, s.t. $P : \mathcal{F} \rightarrow [0, 1]$
  - Disjoint events sum $A \cap B = \emptyset \Leftrightarrow P(A \cup B) = P(A) + P(B)$
  - All events sum to one $P(\Omega) = 1$
- Show that: $P(\bar{A}) = 1 - P(A)$

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.

Markov Models

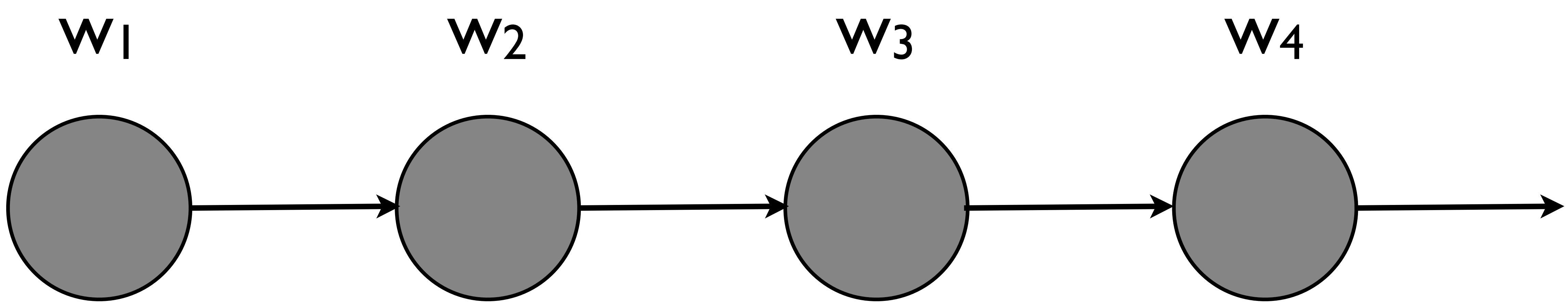
$$p(w_1, w_2, \ldots, w_n) = p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_1, w_2) \\ \cdot p(w_4 \mid w_1, w_2, w_3) \cdots p(w_n \mid w_1, \ldots, w_{n-1})$$

Markov independence assumption

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

$$p(w_1, w_2, \ldots, w_n) \approx p(w_1)p(w_2 \mid w_1)p(w_3 \mid w_2) \\ \cdot p(w_4 \mid w_3) \cdots p(w_n \mid w_{n-1})$$

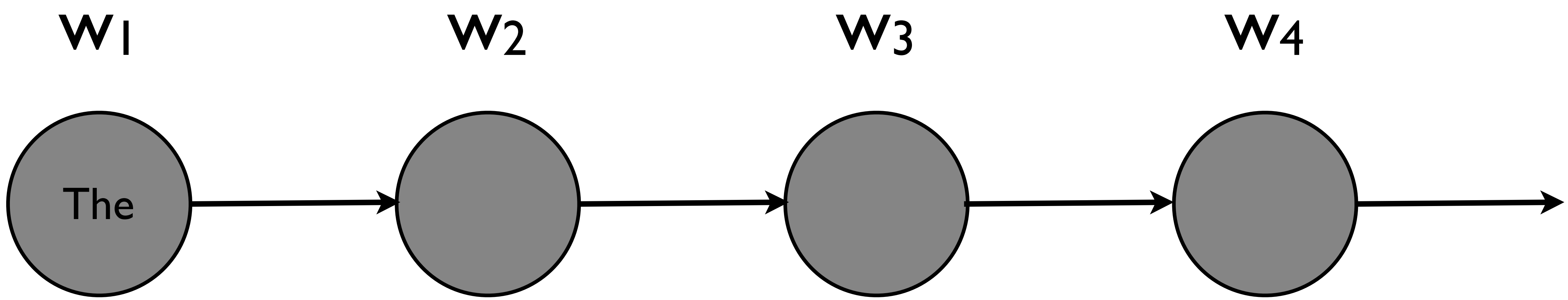
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

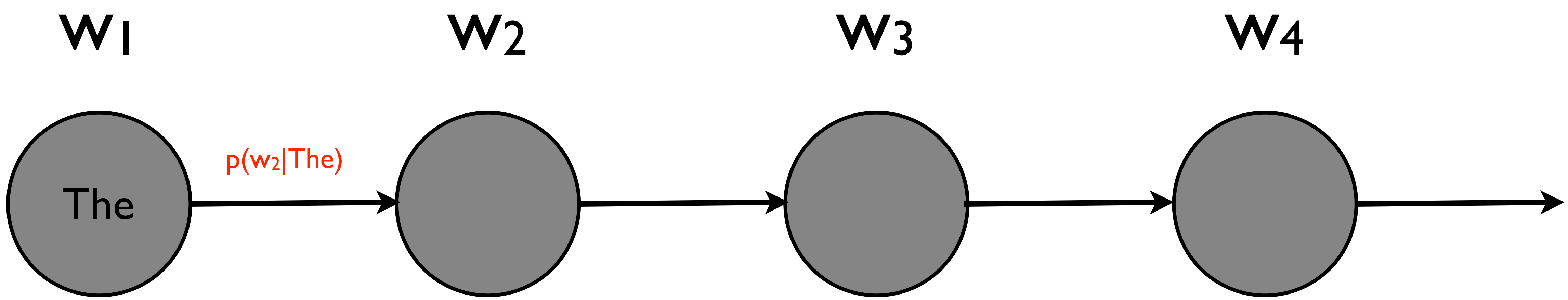
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

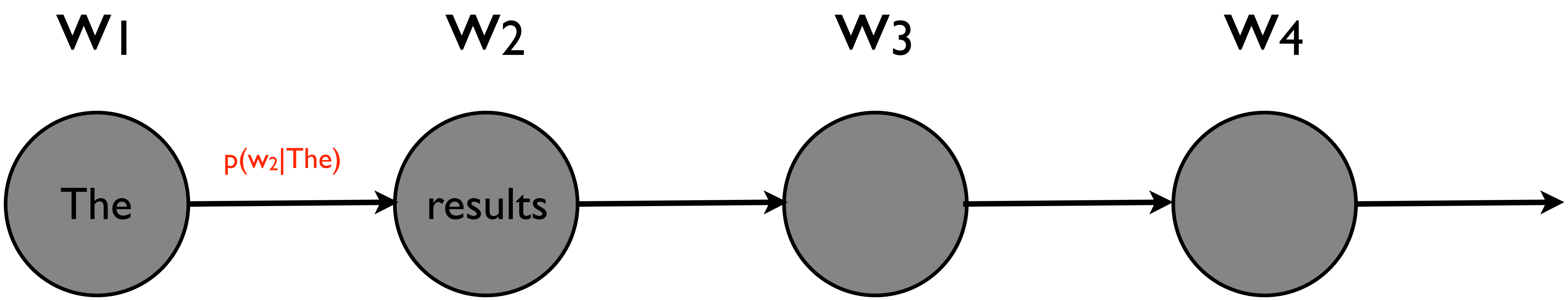
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

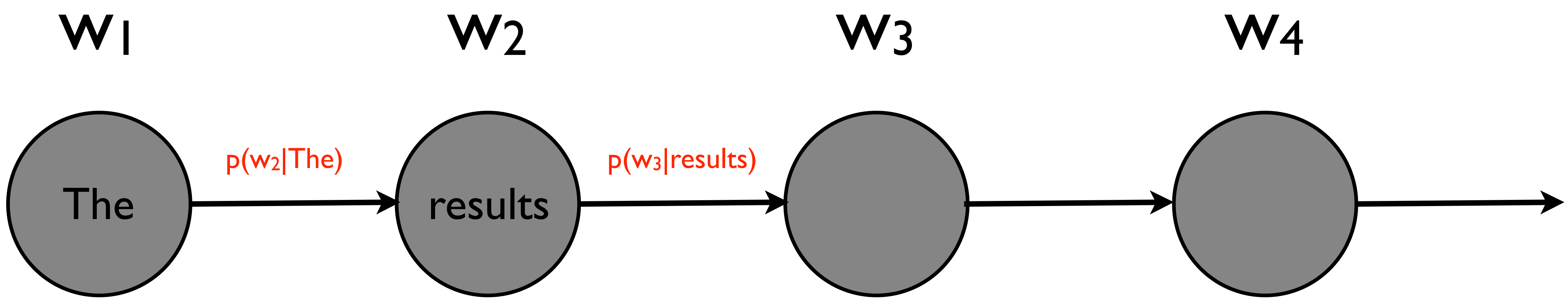
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

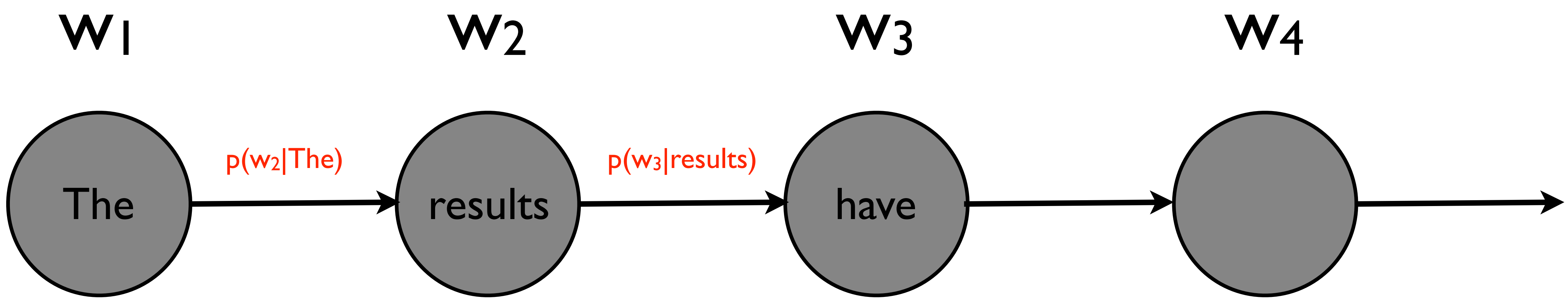
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

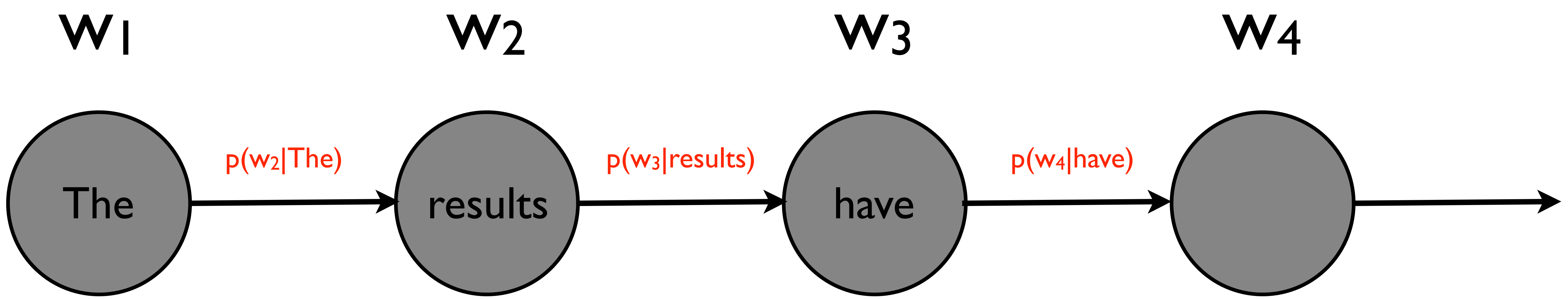
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

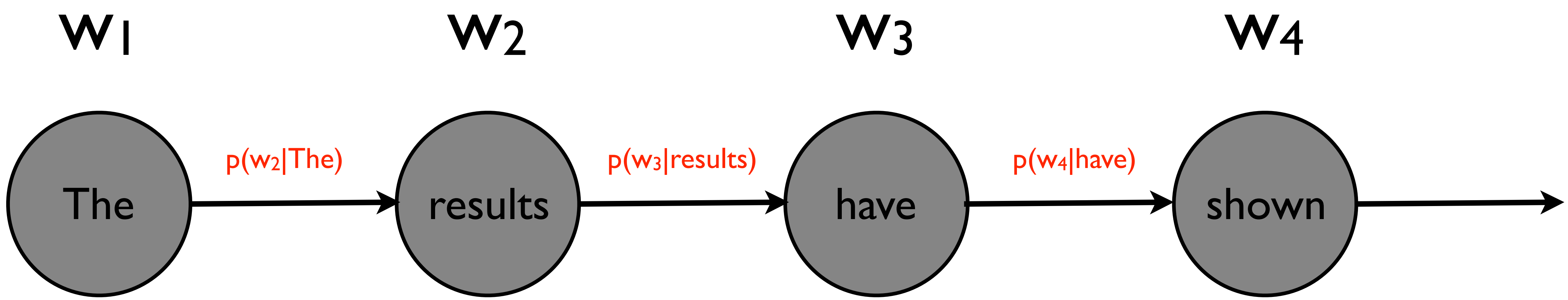
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

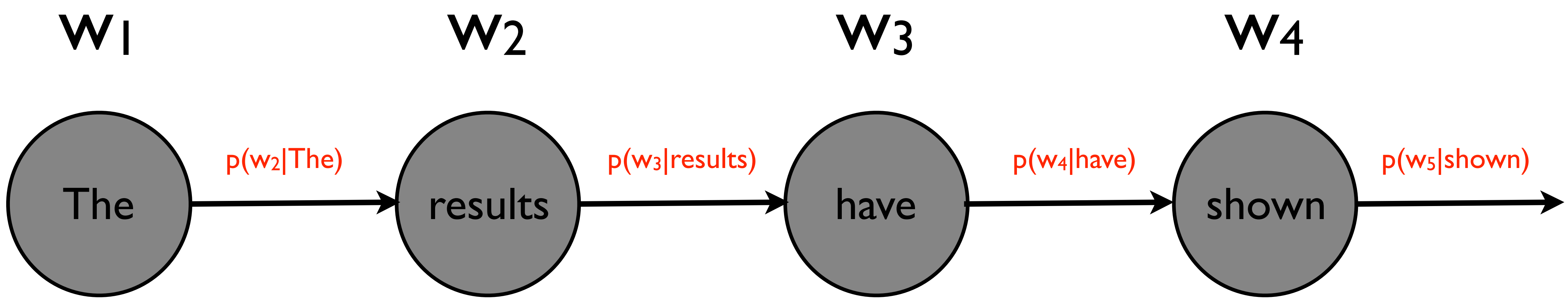
Another View



Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

Another View

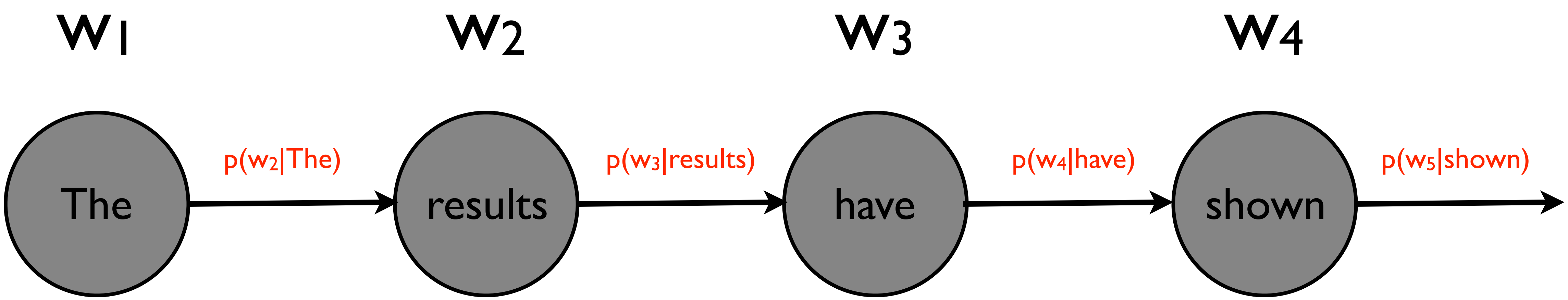


Bigram model as (dynamic) Bayes net

Trigram model as (dynamic) Bayes net

Another View

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

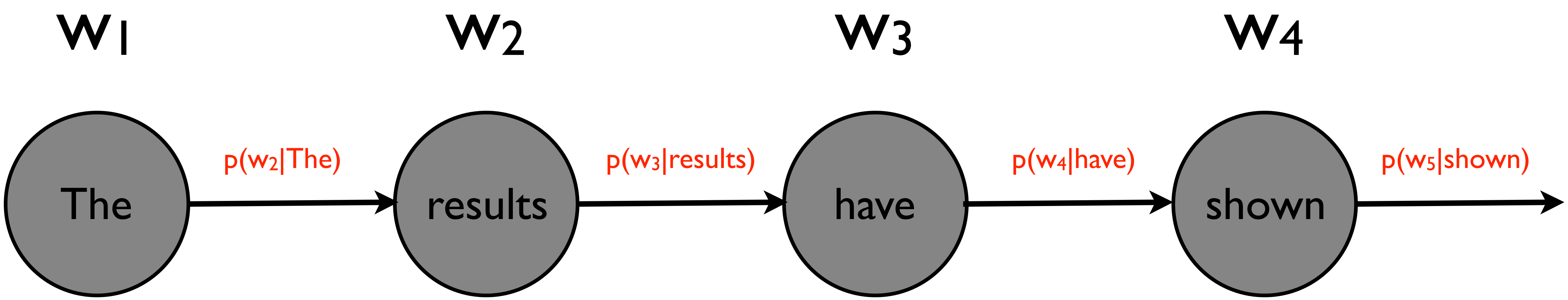


Bigram model as (dynamic) Bayes net

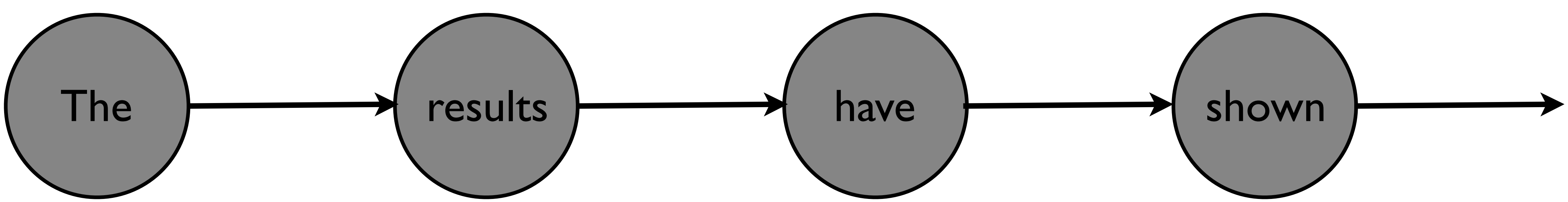
Trigram model as (dynamic) Bayes net

Another View

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



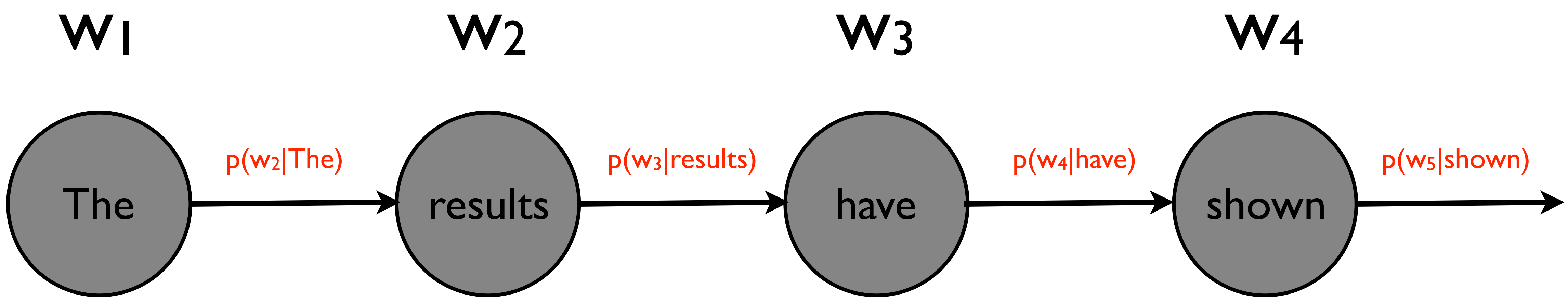
Bigram model as (dynamic) Bayes net



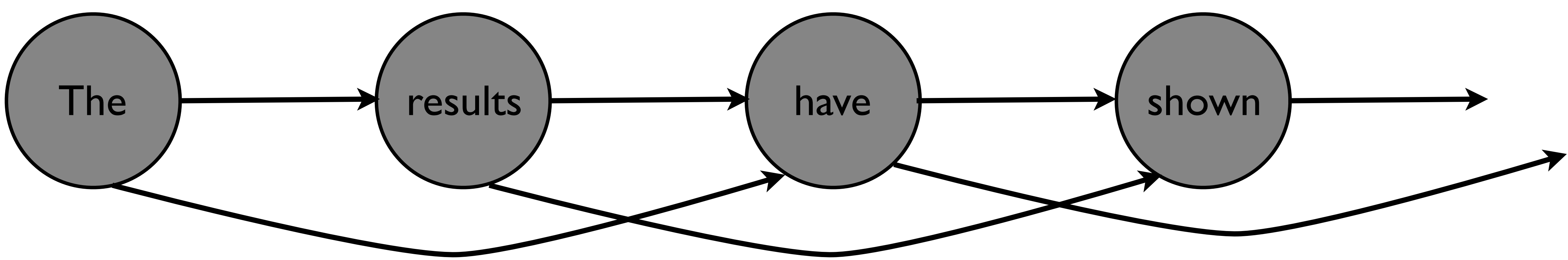
Trigram model as (dynamic) Bayes net

Another View

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



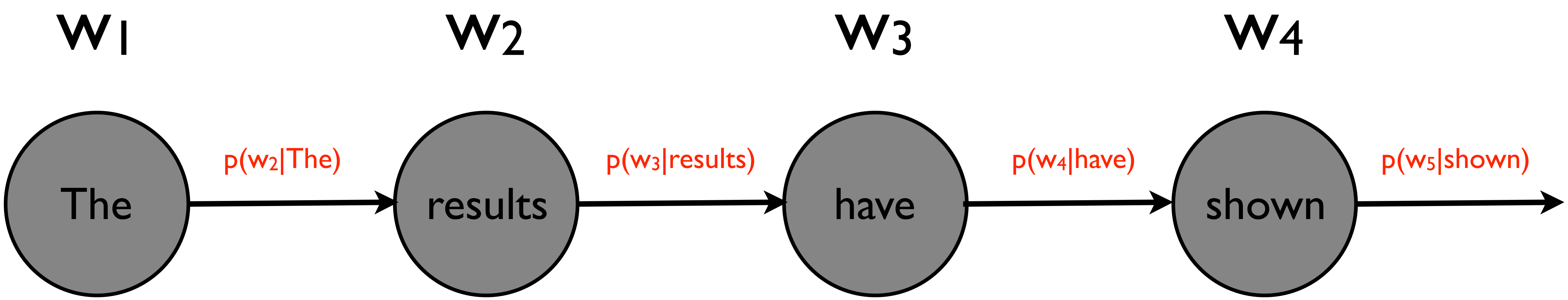
Bigram model as (dynamic) Bayes net



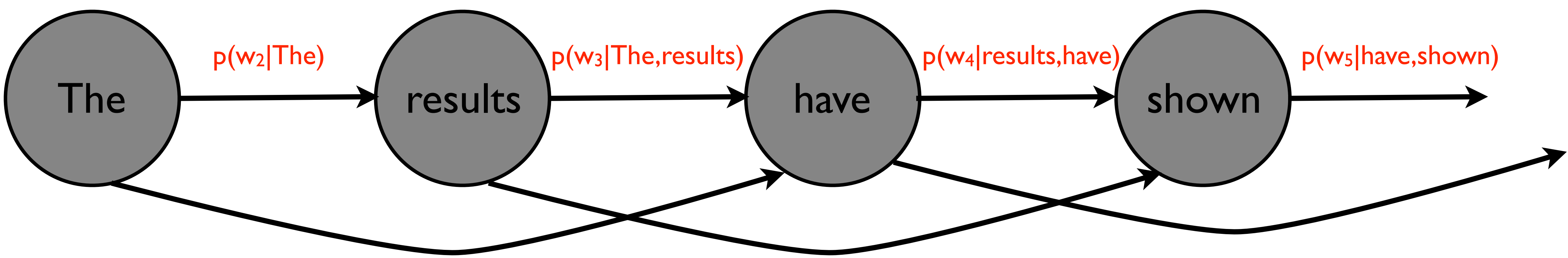
Trigram model as (dynamic) Bayes net

Another View

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

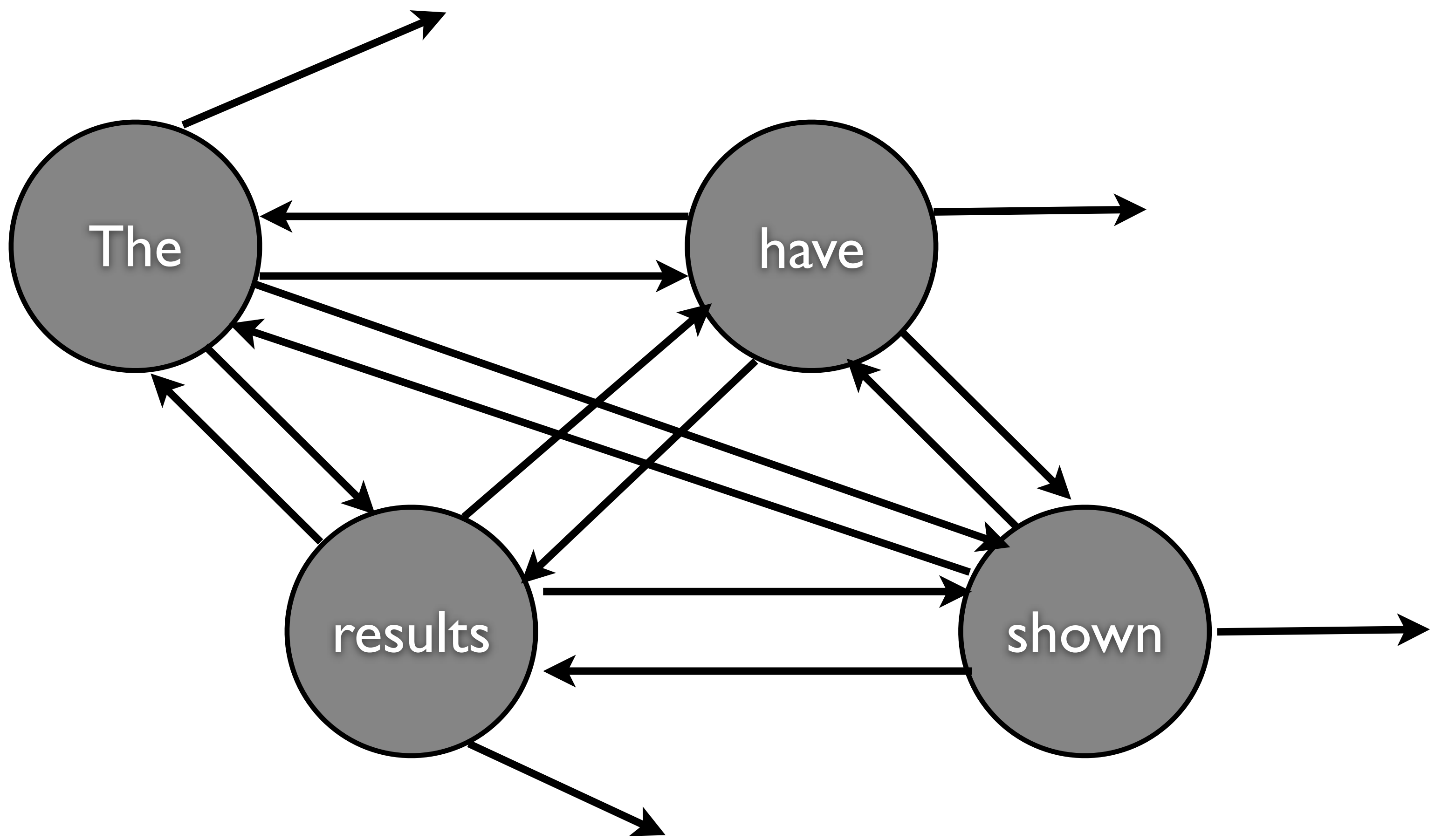


Bigram model as (dynamic) Bayes net



Trigram model as (dynamic) Bayes net

Yet Another View



Bigram model as finite state machine

What about a trigram model?

Classifiers: Language under Different Conditions

Movie Reviews

Movie Reviews

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

Movie Reviews

☺

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

Movie Reviews

☺

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

seen at : amc old pasadena 8 , pasadena , ca (in sdds) paul verhoeven ' s last movie , showgirls , had a bad script , bad acting , and a " plot " (i use the word in its loosest possible sense) that served only to allow lots of sex and nudity . it stank . starship troopers has a bad script , bad acting , and a " plot " that serves only to allow lots of violence and gore . it stinks . nobody will watch this movie for the plot , ...

Movie Reviews

☺

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

☹

seen at : amc old pasadena 8 , pasadena , ca (in sdds) paul verhoeven ' s last movie , showgirls , had a bad script , bad acting , and a " plot " (i use the word in its loosest possible sense) that served only to allow lots of sex and nudity . it stank . starship troopers has a bad script , bad acting , and a " plot " that serves only to allow lots of violence and gore . it stinks . nobody will watch this movie for the plot , ...

Movie Reviews

☺

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

☹

seen at : amc old pasadena 8 , pasadena , ca (in sdds) paul verhoeven ' s last movie , showgirls , had a bad script , bad acting , and a " plot " (i use the word in its loosest possible sense) that served only to allow lots of sex and nudity . it stank . starship troopers has a bad script , bad acting , and a " plot " that serves only to allow lots of violence and gore . it stinks . nobody will watch this movie for the plot , ...

the rich legacy of cinema has left us with certain indelible images . the tinkling christmas tree bell in " it ' s a wonderful life . " bogie ' s speech at the airport in " casablanca . " little elliott ' s flying bicycle , silhouetted by the moon in " e . t . " and now , " starship troopers " director paul verhoeven adds one more image that will live in our memories forever : doogie houser doing a vulcan mind meld with a giant slug . " starship troopers , " loosely based on

Movie Reviews

☺

there ' s some movies i enjoy even though i know i probably shouldn ' t and have a difficult time trying to explain why i did . " lucky numbers " is a perfect example of this because it ' s such a blatant rip - off of " fargo " and every movie based on an elmore leonard novel and yet it somehow still works for me . i know i ' m in the minority here but let me explain . the film takes place in harrisburg , pa in 1988 during an unseasonably warm winter . ...

☹

seen at : amc old pasadena 8 , pasadena , ca (in sdds) paul verhoeven ' s last movie , showgirls , had a bad script , bad acting , and a " plot " (i use the word in its loosest possible sense) that served only to allow lots of sex and nudity . it stank . starship troopers has a bad script , bad acting , and a " plot " that serves only to allow lots of violence and gore . it stinks . nobody will watch this movie for the plot , ...

☹

the rich legacy of cinema has left us with certain indelible images . the tinkling christmas tree bell in " it ' s a wonderful life . " bogie ' s speech at the airport in " casablanca . " little elliott ' s flying bicycle , silhouetted by the moon in " e . t . " and now , " starship troopers " director paul verhoeven adds one more image that will live in our memories forever : doogie houser doing a vulcan mind meld with a giant slug . " starship troopers , " loosely based on

Setting up a Classifier

Setting up a Classifier

- What we want:

  p(☺ | w_1, w_2, ..., w_n) > p(☹ | w_1, w_2, ..., w_n) ?

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:

Setting up a Classifier

- What we want:

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

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

Setting up a Classifier

- What we want:

  p(☺ | w_1, w_2, ..., w_n) > p(☹ | 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 | ☺)

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

A “Bayesian” Classifier

Nowadays also means modeling uncertainty about *p*

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

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

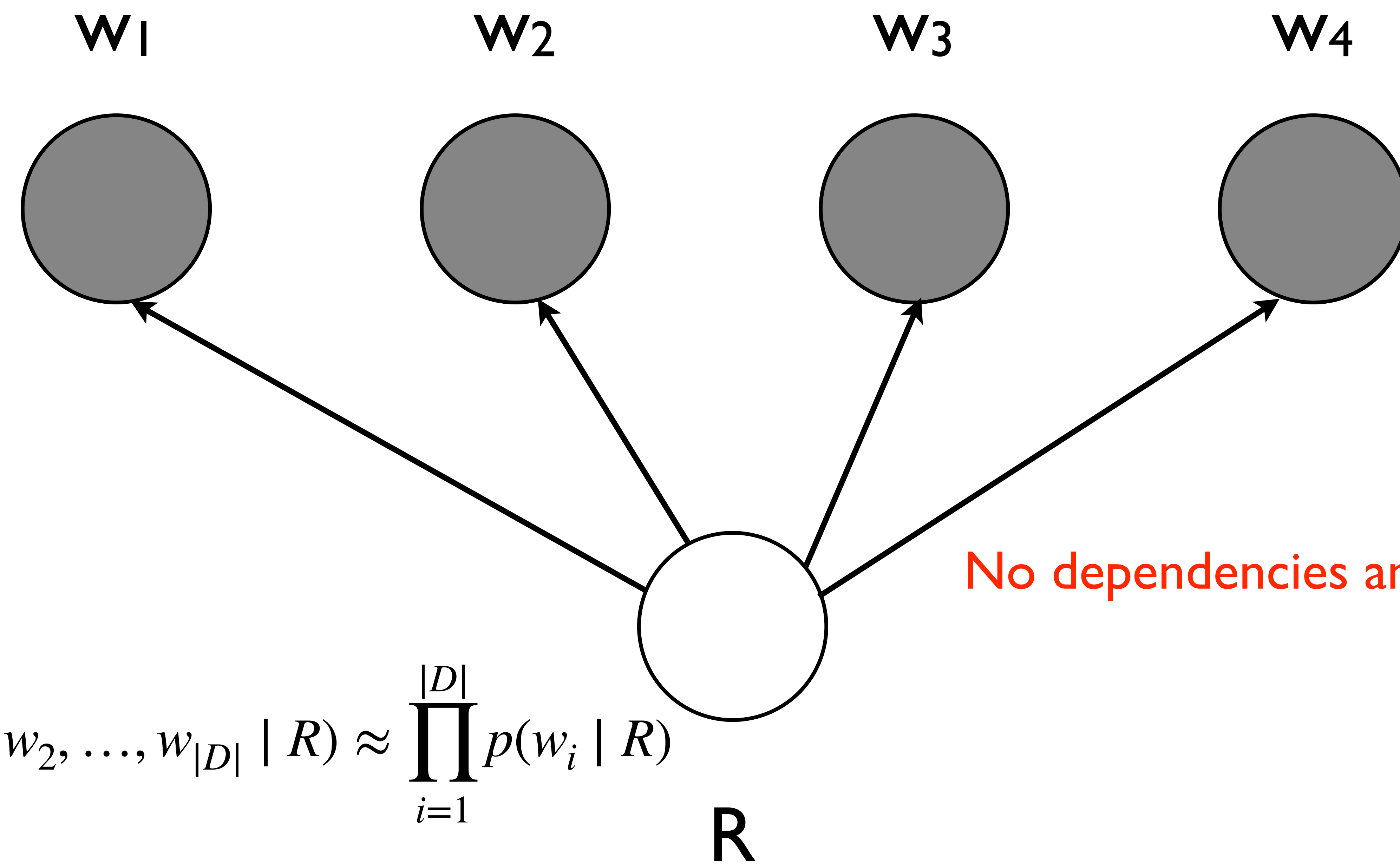
Posterior

Prior

Likelihood

Naive Bayes Classifier

One variable per **token** in document

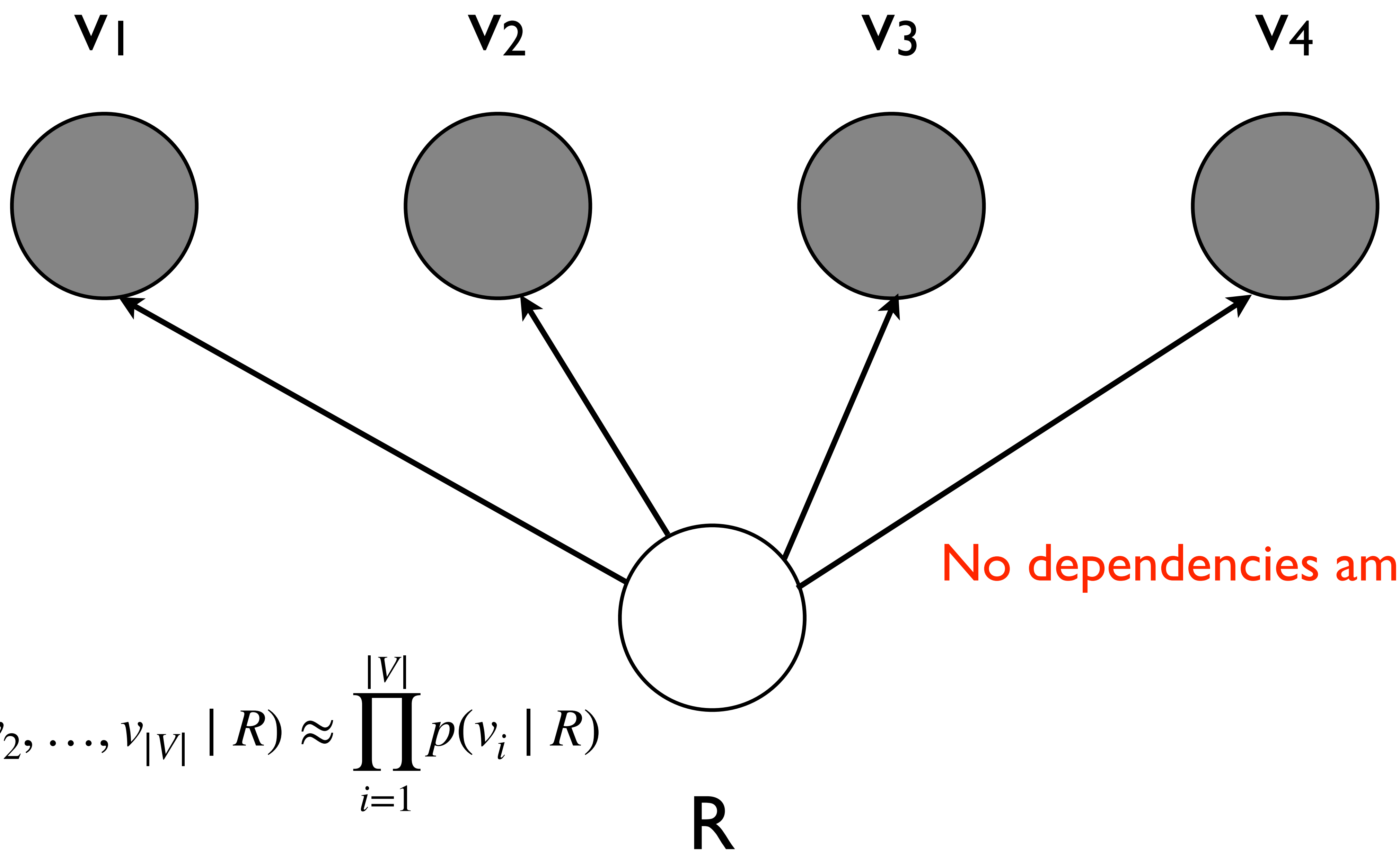


No dependencies among words!

$$p(w_1, w_2, \ldots, w_{|D|} \mid R) \approx \prod_{i=1}^{|D|} p(w_i \mid R)$$

Alternate NB Classifier

One variable per word **type** in vocabulary



No dependencies among words!

$$p(v_1, v_2, \ldots, v_{|V|} \mid R) \approx \prod_{i=1}^{|V|} p(v_i \mid R)$$

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

Estimation for Markov (n-gram) models

Simple Estimation

- Probability courses usually start with equiprobable events
  - Coins, dice, cards used by 17c gamblers
- How likely to get a 6 rolling 1 die?
- How likely the sum of two dice is 6?
- How likely to see 3 heads in 10 flips?

Binomial Distribution

For *n* trials, *k* successes, and success probability *p*:

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Prob. mass function

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Estimation problem: If we observe *n* and *k*, ***what is p?***

Maximum Likelihood

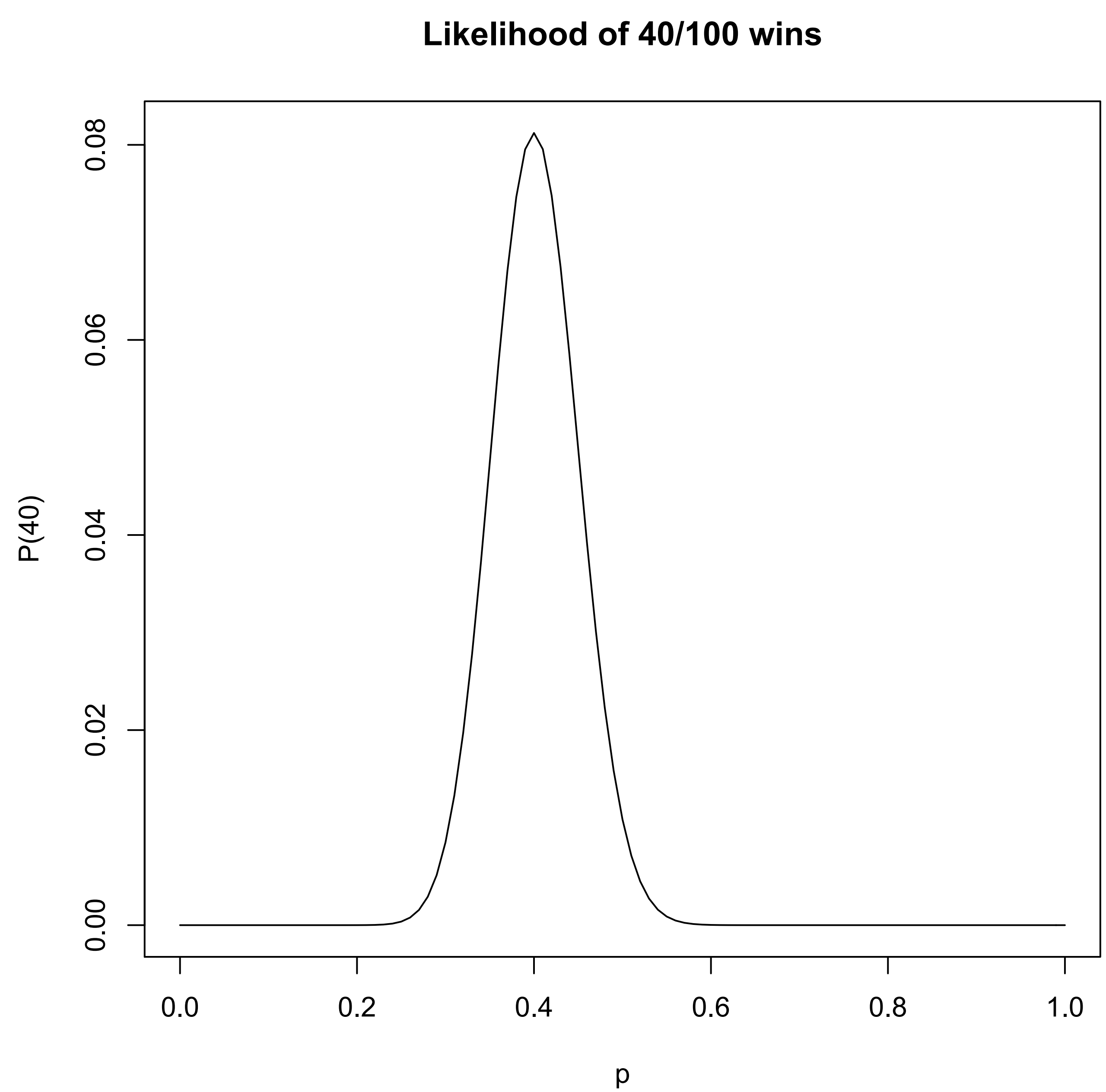
Say we win 40 games out of 100.

$$P(40) = \binom{100}{40} p^{40}(1-p)^{60}$$

The maximum likelihood estimator for *p* solves:

$$\max_p P(\text{observed data}) = \max_p \binom{100}{40} p^{40}(1-p)^{60}$$

Maximum Likelihood



Maximum Likelihood

How to solve $$\max_p \binom{100}{40} p^{40}(1-p)^{60}$$

Maximum Likelihood

How to solve $\max_p \binom{100}{40} p^{40}(1-p)^{60}$

$$
\begin{aligned}
0 &= \frac{\partial}{\partial p}\binom{100}{40} p^{40}(1-p)^{60} \\
&= 40p^{39}(1-p)^{60} - 60p^{40}(1-p)^{59} \\
&= p^{39}(1-p)^{59}[40(1-p) - 60p] \\
&= p^{39}(1-p)^{59}40 - 100p
\end{aligned}
$$

Maximum Likelihood

How to solve $$\max_p \binom{100}{40} p^{40}(1-p)^{60}$$

$$\begin{aligned}
0 &= \frac{\partial}{\partial p}\binom{100}{40} p^{40}(1-p)^{60} \\
&= 40p^{39}(1-p)^{60} - 60p^{40}(1-p)^{59} \\
&= p^{39}(1-p)^{59}[40(1-p) - 60p] \\
&= p^{39}(1-p)^{59}40 - 100p
\end{aligned}$$

Solutions: 0, 1, .4

Maximum Likelihood

How to solve $\max_p \binom{100}{40} p^{40}(1-p)^{60}$

$$
\begin{aligned}
0 &= \frac{\partial}{\partial p}\binom{100}{40} p^{40}(1-p)^{60} \\
&= 40p^{39}(1-p)^{60} - 60p^{40}(1-p)^{59} \\
&= p^{39}(1-p)^{59}[40(1-p) - 60p] \\
&= p^{39}(1-p)^{59} 40 - 100p
\end{aligned}
$$

The maximizer!

Solutions: 0, 1, .4

Maximum Likelihood

How to solve $\max_p \binom{100}{40} p^{40}(1-p)^{60}$

$$\begin{aligned} 0 &= \frac{\partial}{\partial p}\binom{100}{40} p^{40}(1-p)^{60} \\ &= 40p^{39}(1-p)^{60} - 60p^{40}(1-p)^{59} \\ &= p^{39}(1-p)^{59}[40(1-p) - 60p] \\ &= p^{39}(1-p)^{59}40 - 100p \end{aligned}$$

The maximizer!

In general, *k*/*n*

Solutions: 0, 1, .4

Maximum Likelihood

How to solve $\max_p \binom{100}{40} p^{40}(1-p)^{60}$

$$\begin{aligned} 0 &= \frac{\partial}{\partial p}\binom{100}{40} p^{40}(1-p)^{60} \\ &= 40p^{39}(1-p)^{60} - 60p^{40}(1-p)^{59} \\ &= p^{39}(1-p)^{59}[40(1-p) - 60p] \\ &= p^{39}(1-p)^{59} 40 - 100p \end{aligned}$$

The maximizer!

In general, *k*/*n*

Solutions: 0, 1, .4

This is trivial here, but a widely useful approach.

ML for Language Models

- Say the corpus has “in the” 100 times
- If we see “in the beginning” 5 times,

  p_{ML}(beginning | in the) = ?
- If we see “in the end” 8 times,

  p_{ML}(end | in the) = ?
- If we see “in the kitchen” 0 times,

  p_{ML}(kitchen | in the) = ?

ML for Naive Bayes

- Recall: p(+ | Damon movie)

  = p(Damon | +) p(movie | +) p(+)

- If corpus of positive reviews has 1000 words, and "Damon" occurs 50 times,

  p_{ML}(Damon | +) = ?

- If pos. corpus has "Affleck" 0 times,

  p(+ | Affleck Damon movie) = ?

Will the Sun Rise Tomorrow?

Will the Sun Rise Tomorrow?

Laplace's Rule of Succession:
On day n+1, we've observed that the sun has risen s times before.

$$p_{Lap}(S_{n+1} = 1 \mid S_1 + \cdots + S_n = s) = \frac{s+1}{n+2}$$

What's the probability on day 0?
On day 1?
On day 10^6?
Start with prior assumption of equal rise/not-rise probabilities; *update* after every observation.

Laplace (Add One) Smoothing

- From our earlier example:

  p_{ML}(beginning | in the) = 5/100? reduce!

  p_{ML}(end | in the) = 8/100? reduce!

  p_{ML}(kitchen | in the) = 0/100? increase!

Laplace (Add One) Smoothing

- Let V be the vocabulary size:

  i.e., the number of unique words that could follow "in the"

- From our earlier example:

  p_{Lap}(beginning | in the) = (5 + 1)/(100 + V)

  p_{Lap}(end | in the) = (8 + 1)/(100 + V)

  p_{Lap}(kitchen | in the) = (0 + 1) / (100 + V)

Generalized Additive Smoothing

- Laplace add-one smoothing generally assigns *too much* probability to unseen words
- More common to use λ instead of 1:

$$
\begin{aligned}
p(w_3 \mid w_1, w_2) &= \frac{C(w_1, w_2, w_3) + \lambda}{C(w_1, w_2) + \lambda V} \\
&= \mu \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} + (1 - \mu) \frac{1}{V} \\
\mu &= \frac{C(w_1, w_2)}{C(w_1, w_2) + \lambda V}
\end{aligned}
$$

Generalized Additive Smoothing

- Laplace add-one smoothing generally assigns *too much* probability to unseen words
- More common to use λ instead of 1:

$$
\begin{aligned}
p(w_3 \mid w_1, w_2) &= \frac{C(w_1, w_2, w_3) + \lambda}{C(w_1, w_2) + \lambda V} \\
&= \mu \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} + (1 - \mu) \frac{1}{V} \\
\mu &= \frac{C(w_1, w_2)}{C(w_1, w_2) + \lambda V}
\end{aligned}
$$

interpolation

Generalized Additive Smoothing

- Laplace add-one smoothing generally assigns *too much* probability to unseen words

- More common to use λ instead of 1:

What's the right λ?

$$
\begin{aligned}
p(w_3 \mid w_1, w_2) &= \frac{C(w_1, w_2, w_3) + \lambda}{C(w_1, w_2) + \lambda V} \\
&= \mu \frac{C(w_1, w_2, w_3)}{C(w_1, w_2)} + (1 - \mu) \frac{1}{V} \\
\mu &= \frac{C(w_1, w_2)}{C(w_1, w_2) + \lambda V}
\end{aligned}
$$

interpolation

Bias vs. Variance

- Maximum likelihood is unbiased, but smoothing reduces variance
- Unbiased classifiers may **overfit** the training data, performing poorly out of sample
- Too much smoothing can lead to **underfitting**: as $\lambda \to \infty$ or $\mu \to 0$ we approach a uniform distribution, i.e., data are ignored

Picking Parameters

- What happens if we optimize parameters on training data, i.e. the same corpus we use to get counts?
- Maximum likelihood estimate!
- Use *held-out data* aka *development data*
  - or K-fold cross-validation (jackknife)
  - or leave-one-out cross-validation

Good-Turing Smoothing

- Intuition: Can judge rate of novel events by rate of singletons
  - Developed to estimate # of unseen species in field biology
- Let N_r = # of word types with r training tokens
  - e.g., N_0 = number of unobserved words
  - e.g., N_1 = number of singletons (hapax legomena)
- Let $N = \sum r N_r$ = total # of training tokens

Good-Turing Smoothing

- Max. likelihood estimate if w has r tokens? r/N
- Total max. likelihood probability of all words with r tokens? N_r r / N
- Good-Turing estimate of this total probability:
  - Defined as: N_{r+1} (r+1) / N
  - So proportion of novel words in test data is estimated by proportion of singletons in training data.
  - Proportion in test data of the N_1 singletons is estimated by proportion of the N_2 doubletons in training data. etc.
  - p(any given word w/freq. r) = N_{r+1} (r+1) / (N N_r)
- NB: No parameters to tune on held-out data

Backoff

- Say we have the counts:

  $C(\text{in the kitchen}) = 0$

  $C(\text{the kitchen}) = 3$

  $C(\text{kitchen}) = 4$

  $C(\text{arboretum}) = 0$

- ML estimates seem counterintuitive:

  $p(\text{kitchen} \mid \text{in the}) = p(\text{arboretum} \mid \text{in the}) = 0$

Backoff

- Clearly we shouldn't treat "kitchen" the same as "arboretum"
- Basic add-λ (and similar) smoothing methods assign the same prob. to *all* unseen events
- **Backoff** divides up prob. of unseen unevenly in proportion to, e.g., lower-order n-grams
- If $p(z \mid x,y) = 0$, use $p(z \mid y)$, etc.

Deleted Interpolation

- Simplest form of backoff (Jelinek-Mercer)
- Form a *mixture* of different order n-gram models; learn weights on held-out data

$$
\begin{aligned}
p_{del}(z \mid x, y) &= \alpha_3 p(z \mid x, y) + \alpha_2 p(z \mid y) + \alpha_1 p(z) \\
\sum \alpha_i &= 1
\end{aligned}
$$

- How else could we back off?

LMs in IR

- Three possibilities:
  - probability of generating the query text from a document language model
  - probability of generating the document text from a query language model
  - comparing the language models representing the query and document topics

Query Likelihood in IR

- Rank documents by the probability that the query could be generated by language model estimated from that document (a noisy channel model)
- Given user query, start with *p(D | Q)*
- Using Bayes' Rule

$$p(D \mid Q) \stackrel{rank}{=} p(Q \mid D)P(D)$$

$$p(Q \mid D) = \prod_{i=1}^{n} p(q_i \mid D)$$