Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

This section explores some techniques for solving the problems based on probability theory.

A probability function, PROB , assigns a probability to every value of a random variable

A probability function must have following properties, where e_1 , …, e_n are the possible distinct values of a random variable E.

1. PROB (e_i) ≥ 0 for all i ($\forall$ i)
2. PROB (e_i) ≤ 1 for all i ($\forall$ i)
3. $\sum_{i=1, n}$ PROB (e_i) $= 1$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

- Conditional probability is defined by the formula

$$\text{PROB}(e \mid e') = \text{PROB}(e \,\&\, e') \,/\, \text{PROB}(e')$$

where PROB (e & e’) is probability of two events e and e’ occurring simultaneously

- An important theorem relating conditional probabilities is Bayes’ rule. This rule relates conditional probability of an event A given B to the conditional probability of an event B given A:

$$\text{PROB}(A \mid B) = \frac{\text{PROB}(B \mid A) \; * \; \text{PROB}(A)}{\text{PROB}(B)}$$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

- Two events A and B are independent of each other if only if

PROB (A / B) = PROB (A)

which using the definition of conditional probability, is equivalent to saying

PROB (A & B) = PROB (A) * PROB (B)

Example: PROB(Win/Rain) = PROB (Win & Rain)/ PRO(Rain) = .15/.3. If Win and Rain are independent of each other then PROB(Win/Rain) = PROB (Win)/ PRO(Rain) = 0.2*0.3= 0.06 While PROB(Win&Rain) = 0.15. Note ! Probability of winning and raining occur together at a rate much greater than random chance.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

▪Consider an application of probability theory related to language, namely part- of- speech indentification: given a sentence with ambiguous words. Now, determine the most likely lexical category for each word.

Example: word ***flies*** can be either V or N.

C that ranges over the part-of-speech (V, N), W that ranges over all possible words. The problem can be state as determining either PROB (C = N / W = flies) or PROB (C = V / W = flies) is greater.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

The conditional probability for the word *flies* with lexical categories N and V:

PROB (N / flies) = PROB (flies & N) / PROB (flies)

PROB (V / flies) = PROB (flies & V) / PROB (flies)

Hence, we reduce to finding which of PROB (flies & N) and PROB (flies & V) is greater, because the denominator - PROB (flies) is the same in each formula.

Let's say we have a corpus of simple sentence obtaining 1273000 words. Say, there are 1000 uses of word ***flies*** , 400 of them in the N sense and 600 in the V sense.

Then: PROB (flies) = 1000 / 127 3000 = 0.0008

PROB (flies & N) = 400 / 1273000 = 0.0003

Chapter 6: Ambiguity Resolutions: Statistical methods

6.1 basic Probability theory

PROB (flies & V) = 600 / 1273000 = 0.0005

Finally,

PROB (V / flies) = PROB (V & flies) / PROB (flies)
= 0.0005 / 0.0008 = 0.625

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

- For instance, we use information on Harry's past performance (Harry's winning probability of 100 races) to predict how likely he is to win his 101st race.

- We interested in parsing sentences that have never been seen before. Thus we need to use data on previously occurring sentences to predict the next sentences.

- We will always be working with *estimates of probability* rather than actual probability.

✔*Maximum likelihood estimate (MLE)*

If we have seen the word flies **1000 times** before, and **600** of them were as a verb, we assume that **PROB(V/flies) = 0.6**, and use that to guide our guess with 1001st case.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

Maximum likelihood estimate (MLE)

This simple ratio estimate is called Maximum likelihood estimate –MLE.

The accuracy of an estimate increases as the amount of data expands. The estimate is accuracy enough if it falls between 0.25 and 0.75. This range will be called ***margin error.***

result	Estimate of Prob H)	Acceptable estimate
HH	1.0	NO
HT	0.5	YES
TH	0.5	YES
TT	0.0	NO

Figure 6.1: Probabilities with two trails

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

Maximum likelihood estimate (MLE)

Results	Estimate of Prob (H)	Acceptable Estimate
HHH	1.0	NO
HHT	0.66	YES
HTH	0.66	YES
HTT	0.33	YES
THH	0.66	YES
THT	0.33	YES
TTH	0.33	YES
TTT	0.0	NO

Figure 6.2: Probabilities with three trails

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

Sparse data

There are a vast number of estimates needed for natural language applications, and large proportion of these events are quite rare. This is the problem of sparse data.

For instance, the Brown corpus contains about a million words, but due to duplication there are only 49.000 different words and 40.000 of the words occur five times or less.

The worst case occurs, if low-frequency word does not occur at all in one of its possible category.

Its probability in this category would then be estimated as 0, then the probability of the overall sentence containing the word would be 0.

There are other techniques attempt to address the problem of estimating probabilities of low-frequency events.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

Sparse data

Random variable X, technique start with a set of values Vi, computed from the count of the number of times X = xi. MLE uses $V_i = |x_i$ where V_i is exactly the count of number of times $X= x_i$: $\text{PROB}(X = x_i) \equiv V_1 / \Sigma_I x_i$,

One technique to solve the zero probability is to sure that no Vi has value 0 by $V_i = |x_i| + 0.5$, that 0.5 is added to every count

This estimation technique is called **expected likelihood estimate (ELE)**

Different between MLE and ELE:

For instance: consider a word w that does not occur in the corpus.

Consider estimating the probability that w occurs in one of 40 words classes L1…L40 categories. Consider comparing between MLE and ELE.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.2 Estimating Probabilities

Evaluation

How well our new technique performs compared with other algorithms or variants of our algorithms.

The general method for doing this is to divide the corpus into two parts, the training set and the test set. The test set consists of 10 – 20% of total data.

- The training set is then used to estimate the probabilities and
- the algorithm is run on the test set to see how well it does on new data.
- A more thorough method of testing is called **cross-validation:**

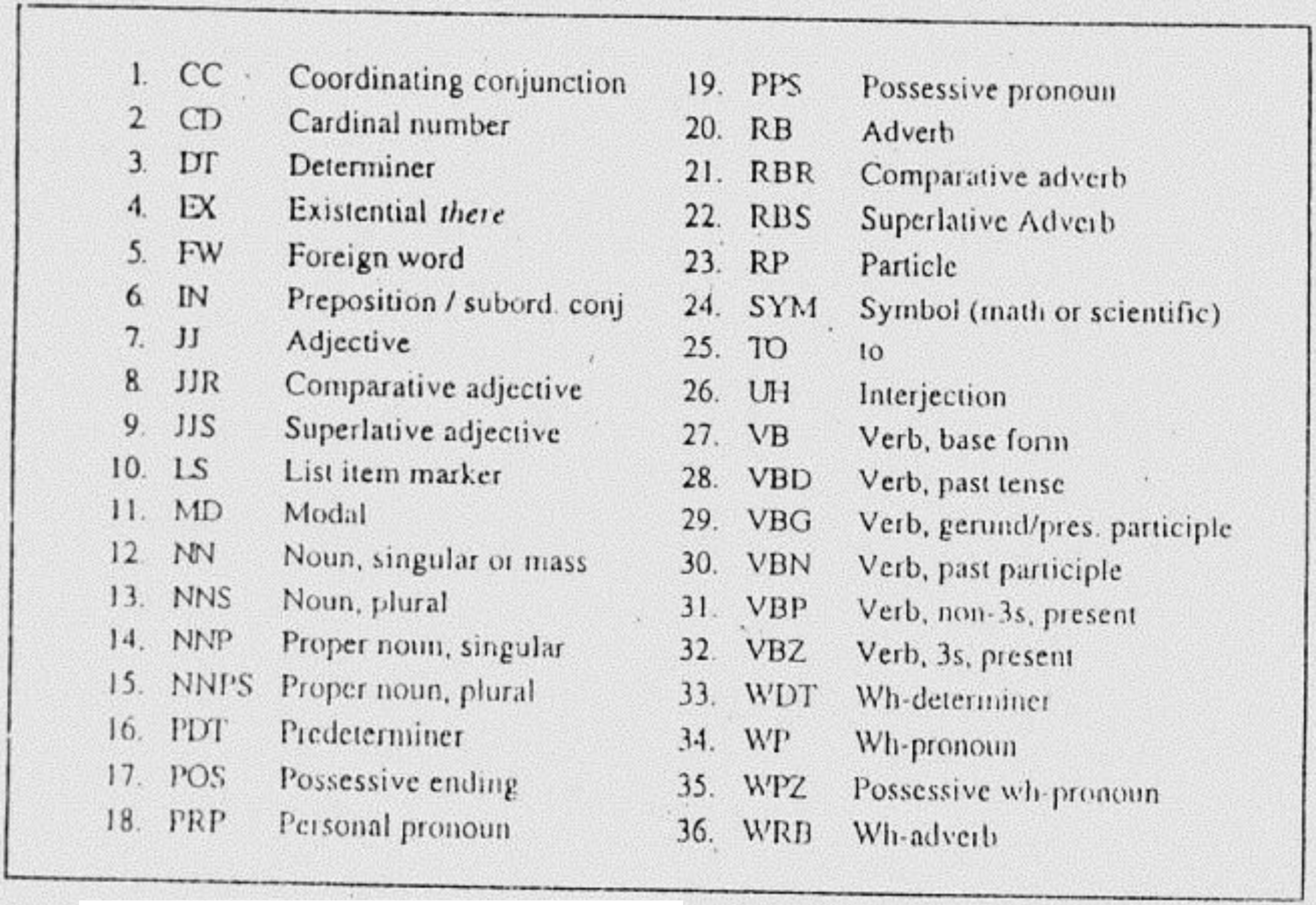
Removing repeatedly different parts of the corpus as the test set, training on the remainder of the corpus, and then evaluating on the new set.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

- Part-of-speech tagging involves selecting the most likely sequence of the syntactic categories for the words a sentence.
- A typical set of tags is used in the Penn Treebank project, is shown in figure 6.3.
- The general method to improve reliability is to use some of the local context of the sentence in which the word appears.
- For instance, if the word preceded by the word ***the ,*** it is much more likely to be N. In the section, we use this technique to exploit such information.

Let W_1 , …, W_T be a sequence of words. We want to find a sequence of lexical categories C_1 , …, $C_{T,}$ that maximizes.



1.	CC	Coordinating conjunction	19.	PPS	Possessive pronoun
2	CD	Cardinal number	20.	RB	Adverb
3.	DT	Determiner	21.	RBR	Comparative adverb
4.	EX	Existential *there*	22.	RBS	Superlative Adverb
5.	FW	Foreign word	23.	RP	Particle
6	IN	Preposition / subord. conj	24.	SYM	Symbol (math or scientific)
7.	JJ	Adjective	25.	TO	to
8	JJR	Comparative adjective	26.	UH	Interjection
9.	JJS	Superlative adjective	27.	VB	Verb, base form
10.	LS	List item marker	28.	VBD	Verb, past tense
11.	MD	Modal	29.	VBG	Verb, gerund/pres. participle
12	NN	Noun, singular or mass	30.	VBN	Verb, past participle
13.	NNS	Noun, plural	31.	VBP	Verb, non-3s, present
14.	NNP	Proper noun, singular	32.	VBZ	Verb, 3s, present
15.	NNPS	Proper noun, plural	33.	WDT	Wh-determiner
16.	PDT	Predeterminer	34.	WP	Wh-pronoun
17.	POS	Possessive ending	35.	WPZ	Possessive wh-pronoun
18.	PRP	Personal pronoun	36.	WRB	Wh-adverb

Figure 6.3 The Penn Treebank tagset

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

1) $PROB\,(C_1, \ldots, C_T / W_1, \ldots, W_T)$

We solve this problem by Bayes' rule, which says that this conditional probability equals

2) $$\frac{PROB\,(C_1, \ldots, C_T) * PROB\,(W_1, \ldots, W_T / C_1, \ldots, C_T)}{PROB\,(W_1, \ldots, W_T)}$$

Finding C1,…, Cn, that gives a maximum value, the common denominator in all these cases will not affect the answer. Thus the problem reduces to finding C1,…, Cn, that maximizes the formula:

3) $PROB\,(C_1, \ldots, C_T) * PROB\,(W_1, \ldots, W_T / C_1, \ldots, C_T)$

There are still no effect methods for calculating the probability of these long sequences accurately, as it would require far too much data.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

But the probabilities can be approximated by marking some independence assumptions. Each of the two expressions in formula 3 will be approximated.

The most common assumptions use either one or two previous categories.

Thẻ **bigram model** looks at pairs of categories (or words) and use the conditional probability that C_i will follow C_{i-1}, written as PROB(C_i / C_{i-1}).

The **trigram model** use conditional probability of one category (or word) given the two preceding categories (or words), that is PROB(C_i / C_{i-2} C_{i-1})

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

n-gram model, in which n represents the number of word used in the pattern.

While the trigram model will produce better result in practice.

We use bigram here for simplicity

$$PROB (C_1 , \ldots, C_T) \cong \prod_{i = 1, T} PROB (C_i / C_{i-1})$$

To account for beginning of the sentence, we posit a pseudocategory $\varnothing$ at the position 0 as value of C_0

If ART at the beginning of a sentence that the first bigram will be PROB (ART / $\varnothing$).

Example: approximation of the probability of the sequence ART N V N using bigram would be

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

- $\text{PROB}(\text{ART N V N}) \cong \text{PROB}(\text{ART} / \varnothing) * \text{PROB}(\text{N} / \text{ART}) * \text{PROB}(\text{V} / \text{N}) * \text{PROB}(\text{N} / \text{V})$

- The second probability in formula 3:

- $\text{PROB}(W_1, \ldots, W_T / C_i, \ldots, C_T) \cong \prod_{i=1,T} \text{PROB}(W_i / C_i)$

- Can be approximated by assuming that a word appears in a category independent of the words in the preceding or succeeding categories.

-With these two approximations, the problem has changed into finding the sequence C1…CT that maximizes the value of

3’) $\text{PROB}(C_1, \ldots, C_T) * \text{PROB}(W_1, \ldots, W_T / C_1, \ldots, C_T) \cong \prod_{i=1,T} \text{PROB}(C_i / C_{i-1}) * \text{PROB}(W_i / C_i)$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

The advantage of formula 3') is that the probabilities involved can be readily estimated from a corpus of text labeled with parts of speech.

In particular, given a database of text, the bigram probabilities can be estimated simply by counting the number of times each pair of categories occurs compared to the individual category counts.

Example: the probability that a V follows an N would be estimated as follows:

$$\text{PROB}(Ci=V/Ci\text{-}1=N) \cong \frac{\text{count (N at position i-1 and V at i)}}{\text{count (N at position i-1)}}$$

To deal with the problem of the sparse data, any bigram is not listed in figure 6.4, will be assumed to have a token probability 0.0001.

Chapter 6: Ambiguity Resolutions: Statistical methods

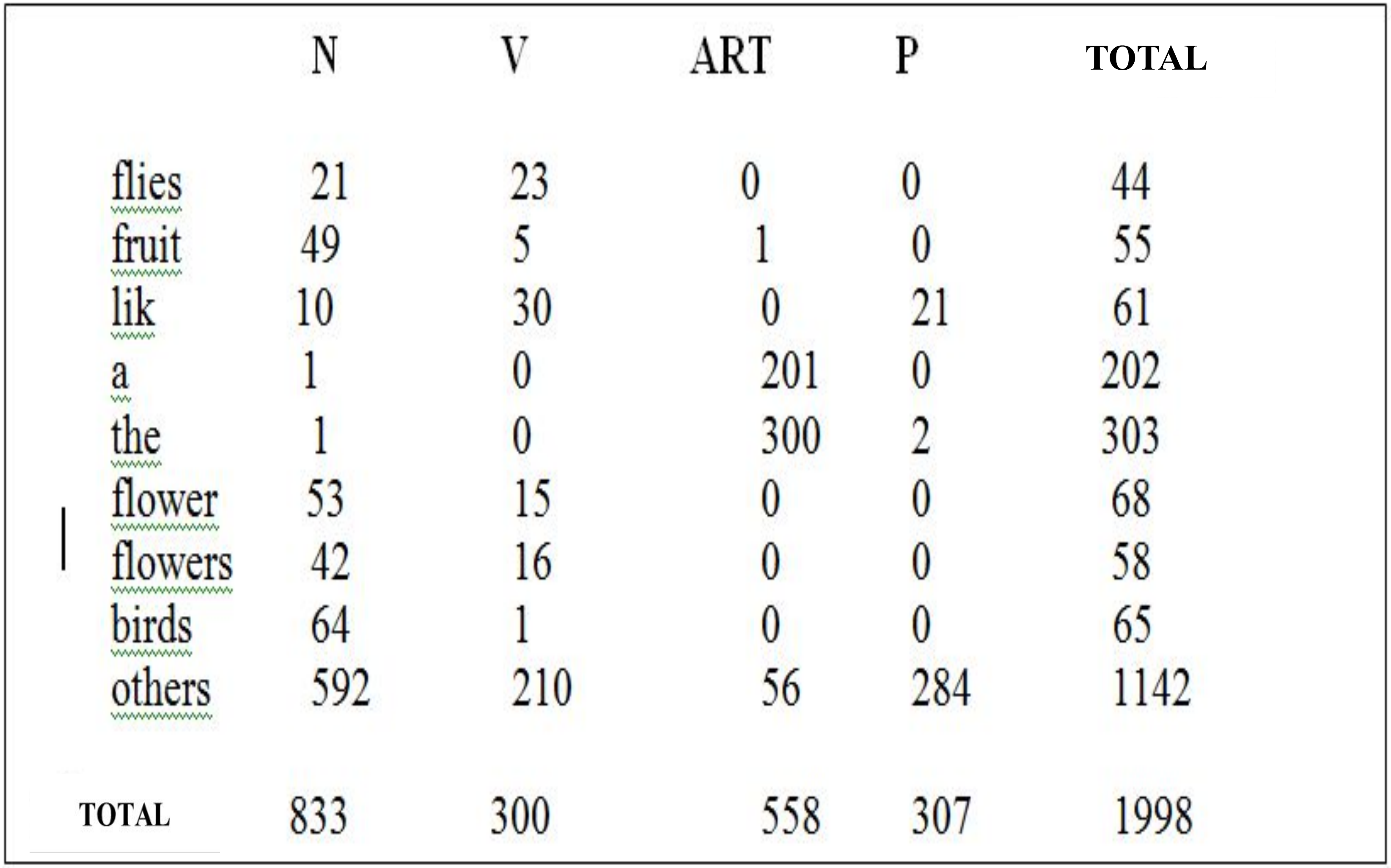
6.3 Part – of- speech Tagging

category	Count at i	pair	Count at i,i+1	bigram	estimate
O	300	O, ART	213	PROB(ART/O)	0.71
O	300	O, N	87	PORB(N/O)	0.29
ART	558	ART, N	558	PROB(N/ART)	1.0
N	833	N,V	258	PROB(V/N)	0.43
N	833	N, N	108	PROB(N/N)	0.13
N	833	N, P	366	PROB(P/N)	0.44
V	300	V, N	75	PROB(N/V)	0.35
V	300	V, ART	194	PROB(ART/V)	0.65
P	307	P, ART	226	PROB(ART/P)	0.74
P	307	P, N	81	PROB(N/P)	0.26

Figure 6.4 Bigram probabilities from a corpus [1]

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging



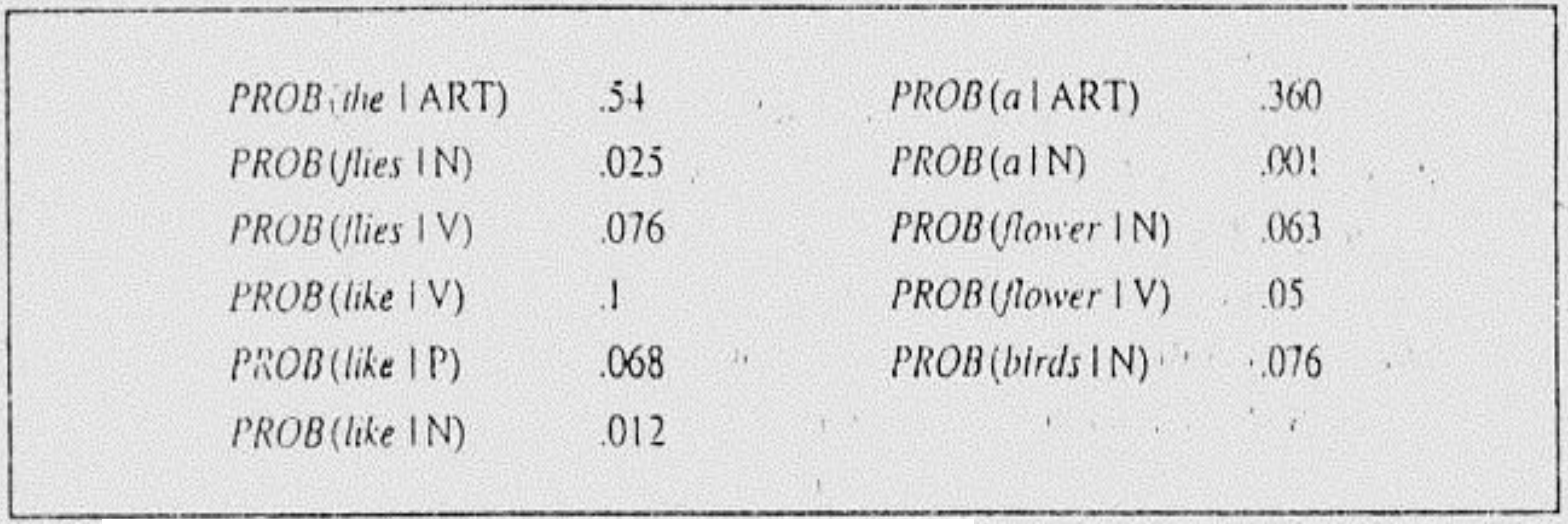
	N	V	ART	P	TOTAL
flies	21	23	0	0	44
fruit	49	5	1	0	55
lik	10	30	0	21	61
a	1	0	201	0	202
the	1	0	300	2	303
flower	53	15	0	0	68
flowers	42	16	0	0	58
birds	64	1	0	0	65
others	592	210	56	284	1142
TOTAL	833	300	558	307	1998

Figure 6.5 A summary of some of the words counts in the corpus

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

- Lexical probabilities PROB (Wi / Ci) can be estimated simply by counting the number of occurrence of each word by category. ***Example***: some lexical probabilities are estimated based on data of figure 6.5, and are shown on figure 6.6 .



PROB(*the* | ART)	.54	PROB(*a* | ART)	.360
PROB(*flies* | N)	.025	PROB(*a* | N)	.001
PROB(*flies* | V)	.076	PROB(*flower* | N)	.063
PROB(*like* | V)	.1	PROB(*flower* | V)	.05
PROB(*like* | P)	.068	PROB(*birds* | N)	.076
PROB(*like* | N)	.012		

Figure 6.6 The lexical generation probabilities

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

- We can find the sequence of categories that has the highest probability of generating a specific sentence (POST) by the independent assumption that were made about the data.

- Since we only deal with bigram probabilities, the probability is that the i'th word in category Ci depends only on category of the (i-1)th word, Ci-1.

-Thus the process can be modeled by a special form of probabilistic finite state, as shown in Figure 6.7.Each node represents a possible lexical category and the transition probabilities. Networks like that in Figure 6.7 are called Markov Chains.

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

***Example:* Given a sequence of categories** ART N V N, that has the probability as follows:

0.71 * 1 * 0.43 * 0.35 = 0.107 (data from Figure 7.7)

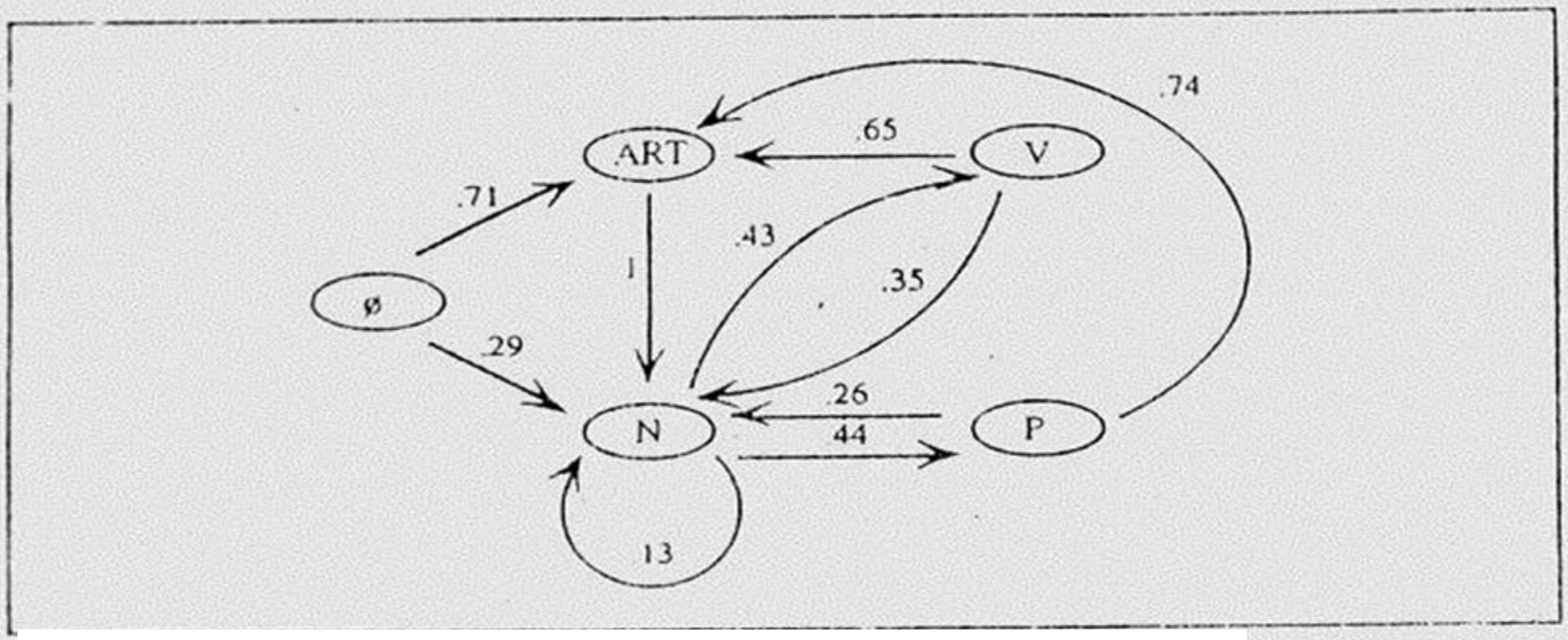


Figure 6.7 A Markov chain capturing the bigram probabilities

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

Example: The probability that the sequence N V ART N generates the output ***flies like a flower*** is computed as follows:

- The probability of the path N V ART N, given the Markov Model in Figure 6.7: **PROB (N V ART N) = 0.29 * 0.43 * 0.65 * 1 = 0.081**

- The probability of the output ***flies like a flowers*** for this sequence is computed from probabilities in Figure 6.6:

PROB (flies / N) * PROB (like / V) * PROB (a / ART)

*** PROB (flower / N) = 0.025 * 1 * 0.36 * 0.063 = 5.4 * 10^{-5}**

- Multiplying these together give us the likelihood, that the HMM would generates the sentence:

PROB (N V ART N / flies like a flower) = 5.4 * 10^{-5} * 0.081 ≡ $\underline{4.37 * 10^{-6}}$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

- The formula for computing probability of sentence $w_1 \dots w_T$ given sequence $C_1 \dots C_N$ is:

$$\prod_{i = 1, T} \text{PROB} (C_i / C_{i-1}) * \text{PROB} (W_i / C_i)$$

If we keep track of the most likely sequence found so far for each possible ending category, so we can ignore all the other less likely sequences.

Example: To find the most likely categories for sentence *flies like a flower* are shown in Figure 6.8. There are 256 different sequences of length four (words).

To find the most likely sequence we sweep forward through the words one at the time finding for each ending category. This algorithm usually is called Viterbi algorithm

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

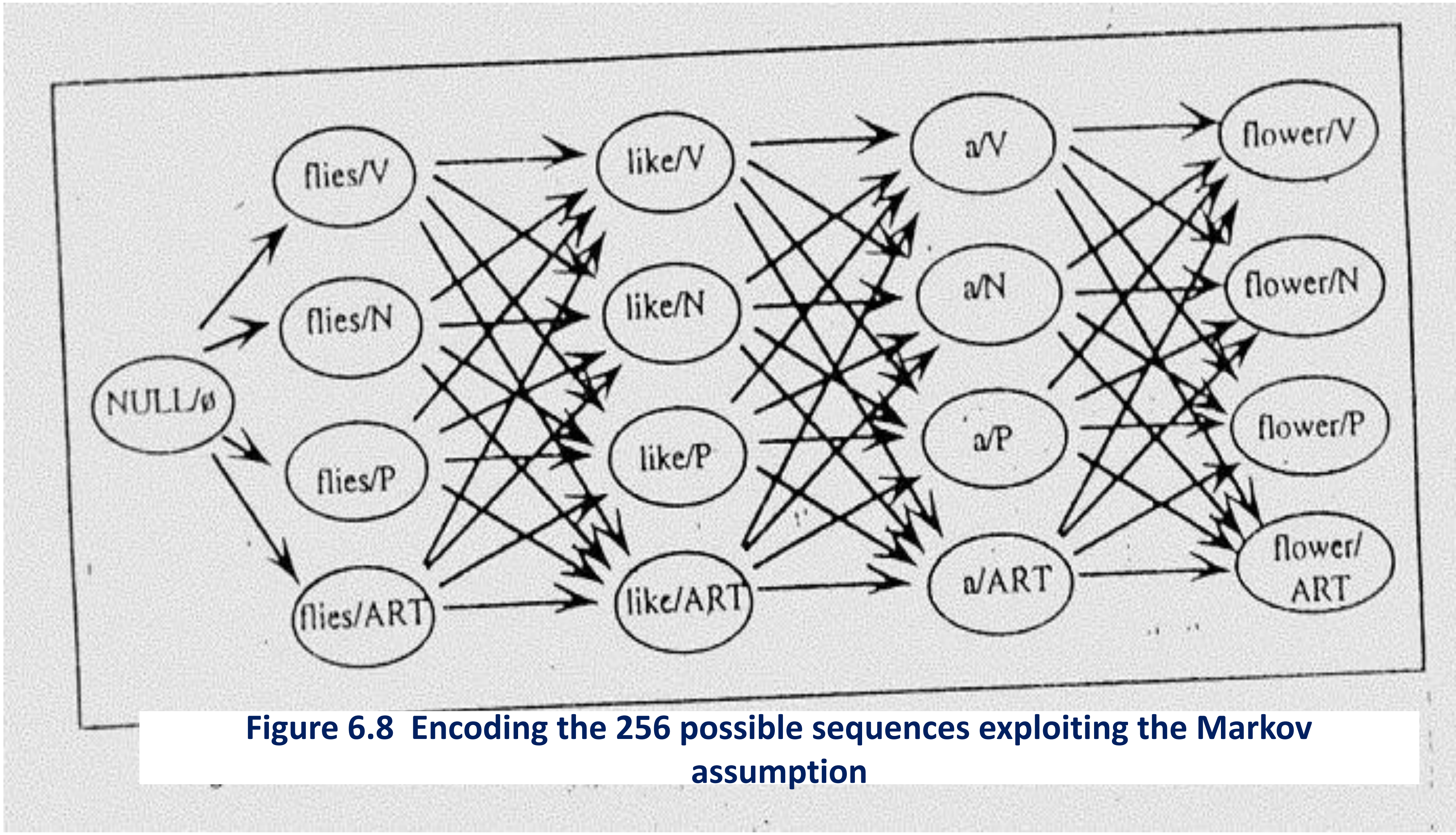


Figure 6.8 Encoding the 256 possible sequences exploiting the Markov assumption

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

The Viterbi algorithm

Given word sequence: w1…wT, lexical categories: L1…LN,lexical probabilities PROB(wi/Li) and bigram probabilities PROB(Li/Li-1), find the most likely sequence of lexical categories C1….CT for the word sequence.

Initialization step:

```
For i := 1 to N do
SEQSCORE (i, 1) = PROB( $w_1$ | $l_i$ )* PROB( $l_i$ | $\varnothing$ )
BACKPTR( i |1) = 0
```

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

The Viterbi Algorithm (cons)

Iteration step:

```
For t := 2 to T do
  for i := 1 to N do
SEQSCORE ( I,t) = MAX j=1,N (SEQSCORE ( j,t-1)
                        *PROB(li | lj))* PROB(wt | li )
BACKPTR(i,t) = index of j that gave the max above
```

Sequence identification step

```
C(T) = I that maximizes SEQSCORE ( i, t)
For  i := T-1 to 1 do
C(i) = BACKPTR ( C(i+1),i+1)
```

Chapter 6: Ambiguity Resolutions: Statistical methods

6.3 Part – of- speech Tagging

Example: Using the Viterbi algorithm compute a probability of the sequence W= *flies like a flower*. ***L= V,N,ART, P***

i=1 to N	SEQ(1,1) = PROB(flies/V)*PROB(V/O)
	SEQ(2,1) = PROB(flies/V)*PROB(N/O)
	SEQ(3,1) = PROB(flies/V)*PROB(ART/O)
	SEQ(4,1) = PROB(flies/V)*PROB(P/O)
		7.6*10-2 * 10-4 = 7.6 *10-6
		0.025*0.29 = **7.25 * 10-3**
		0
		0

t= 2 to 4
I = 1 to 4
	SEQ(1,2) = max j=1,4 (SEQ(1,1)* PROB(V/V), SEQ(2,1)*PROB(V/N))* PROB(like/V)
	SEQ(2,2) = max j=1,4 (SEQ(1,1)* PROB(N/V), SEQ(2,1)*PROB(N/N))* PROB(like/N)
	
	SEQ(3,2) = max j=1,4 (SEQ(1,1)* PROB(ART/V), SEQ(2,1)*PROB(ART/N))*PROB(like/ART)
	SEQ(4,2) max j=1,4 (SEQ(1,1)* PROB(P/V), SEQ(2,1)*PROB(P/N))*PROB(like/P)
		Max(7.6 *10-8, **7.25 *10-3 *0.43**) *0.1 = **3.1 *10-4**
		Max (7.6*10-6*0.35, 7.25 *10-3 *0.13)*0.012= **1.13 *10-5**
		
		0
		
		Max(7.6*10-6*10-4, 7.25*10-3*0.44)*0.068 = **2.2 *10-4**

t=3 to 4
i=1 to 4
	SEQ(1,3) = max j=1,4 (SEQ(1,2)* PROB(V/V),
	SEQ(2,2)*PROB(V/N),SEQ(4,2)*PROB(V/P))
	*PROB(a/V)
	
	SEQ(2,3) = max j=1,4 (SEQ(1,2)* PROB(N/V),
	SEQ(2,2)*PROB(N/N),SEQ(4,2)*PROB(N/P))
	*PROB(a/N)
	
	SEQ(3,3) = max j=1,4 (SEQ(1,2)* PROB(ART/V),
	SEQ(2,2)*PROB(ART/N),SEQ(4,2)*
	PROB(ART/P))*PROB(a/ART)
	
	SEQ(4,3) = max j=1,4 (SEQ(1,2)* PROB(P/V),
	SEQ(2,2)*PROB(P/N),SEQ(4,2)*
	PROB(P/P))*PROB(a/P)
		=max(3.1 *10-4*10-4,
		1.13 *10-5*0.43,
		2.2 *10-4*10-4)*0 **=0**
		
		=max(3.1 *10-4*0.35,
		1.13 *10-5*0.13,
		2.2 *10-4*0.26)*0.01
		=**5.7*10-7**
		max(3.1 *10-4* 0.65,
		1.13 *10-5*10-4,
		2.2 *10-4*0.74)*0.36= **2.01*10-5**
		
		max(3.1 *10-4* 10-4,
		1.13 *10-5*0.44,
		2.2 *10-4*10-4)*0 = **0**

t=4
i=1 to 4
	SEQ(1,4) = max j=1,4 (SEQ(2,3)* PROB(V/N),
	SEQ(3,3)*PROB(V/ART)) *PROB(flower/V)
	
	SEQ(2,4) = max j=1,4 (SEQ(2,3)* PROB(N/N),
	SEQ(3,3)*PROB(N/ART)) *PROB(flower/N)
	
	SEQ(3,4) = max j=1,4 (SEQ(2,3)* PROB(ART/N),
	SEQ(3,3)*PROB(ART/ART)) *PROB(flower/ART)
	SEQ(4,4) = max j=1,4 (SEQ(2,3)* PROB(P/N),
	SEQ(3,3)*PROB(P/ART)) *PROB(flower/P)
		- Max(5.7*10-7*0.43,
		2.01*10-5*10-4)*0.05= 1.2*10-8
		
		- Max(5.7*10-7*0.13,
		2.01*10-5)*0.63= **1.26*10-5**
		
		0
		
		0

Example: flies like a flower. L= V,N,ART, P

- **Sequence Identification Step**
- C(T) = i that maximizes SEQ (I,T) SEQ (2,4) = 4.5*10-6
- C(4) = 2
- For i = 4– 1 to 1 do
- C(3) = BACKPTR(C(3+1), 4) = BACKPTR(2,4) = 3
- C(2) =BACKPTR(C(3), 3) = BACKPTR(3, 3) = 1
- C(1) = BACKPTR(C(2), 2) = BACKPTR(4,2) = 2
- **2-1-3-2 -> N V ART N**

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 Obtaining lexical probabilities

-The simplest technique for estimating lexical probability is computed by a number of times each word appears in the corpus in each category.

-The probability that the word w appears in a lexical category L_j out of possible categories $L_1 \ldots L_N$ could be estimated by a formula:

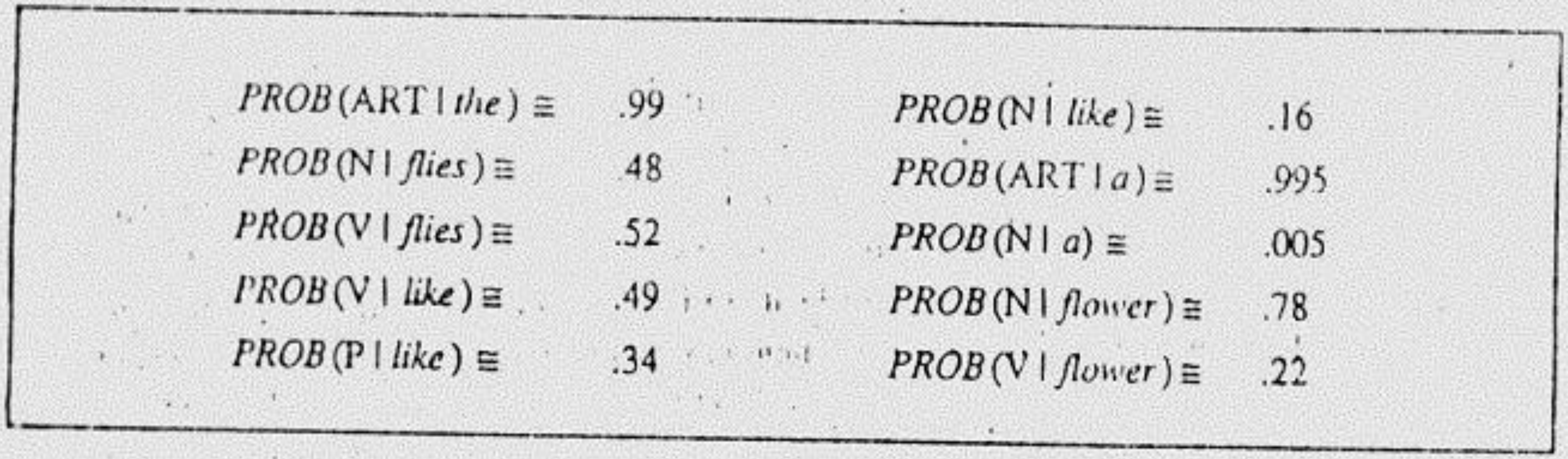
$$PROB (Lj / w) \cong count (Lj \& w) / \Sigma_{i=1, N} \; count (L_i / w)$$

- A better estimate would be obtained by computing how likely it is that category Lj occurs at a position t over all sequences given

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 obtaining lexical probabilities

the input $w_1 \ldots w_t$. In the other word, to find the one sequence that yields the maximum probability for the input from all sequences.



$PROB(ART \mid the) \cong$	.99	$PROB(N \mid like) \cong$	.16
$PROB(N \mid flies) \cong$	.48	$PROB(ART \mid a) \cong$	.995
$PROB(V \mid flies) \cong$	.52	$PROB(N \mid a) \cong$	.005
$PROB(V \mid like) \cong$	.49	$PROB(N \mid flower) \cong$	.78
$PROB(P \mid like) \cong$	.34	$PROB(V \mid flower) \cong$	.22

Figure 6.9 Context independent estimates for the lexical categories

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 Obtaining lexical probabilities

Example: The probability that flies is as N in the sentence *The flies like flowers* would be calculated by summing the probability of all sequences that end with *flies* as N.

Given transition (Figure 7.4) and lexical generation probabilities (Figure 7.6), the sequences would be calculated as follows:

The/ART flies/N	9.58*10-3
The/ N flies/N	1.13* 10-6
The/P flies/N	4.55*10-9

Three nonzero sequences above have sum that is up 9.58*10-3

Likewise, three nonzero sequences end with flies as V, yielding a total sum of 1.13*10-5. The sum of all sequences will be the probability of the sequence *The flies,* namely 9.591*10-3

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 Obtaining lexical probabilities

The probability that flies is a noun as follows:

PROB(flies/N | The flies) =PROB(flies/N&The flies)/PROB(The flies)

= 9.58*10-3/9.591*10-3 = **0.9988**

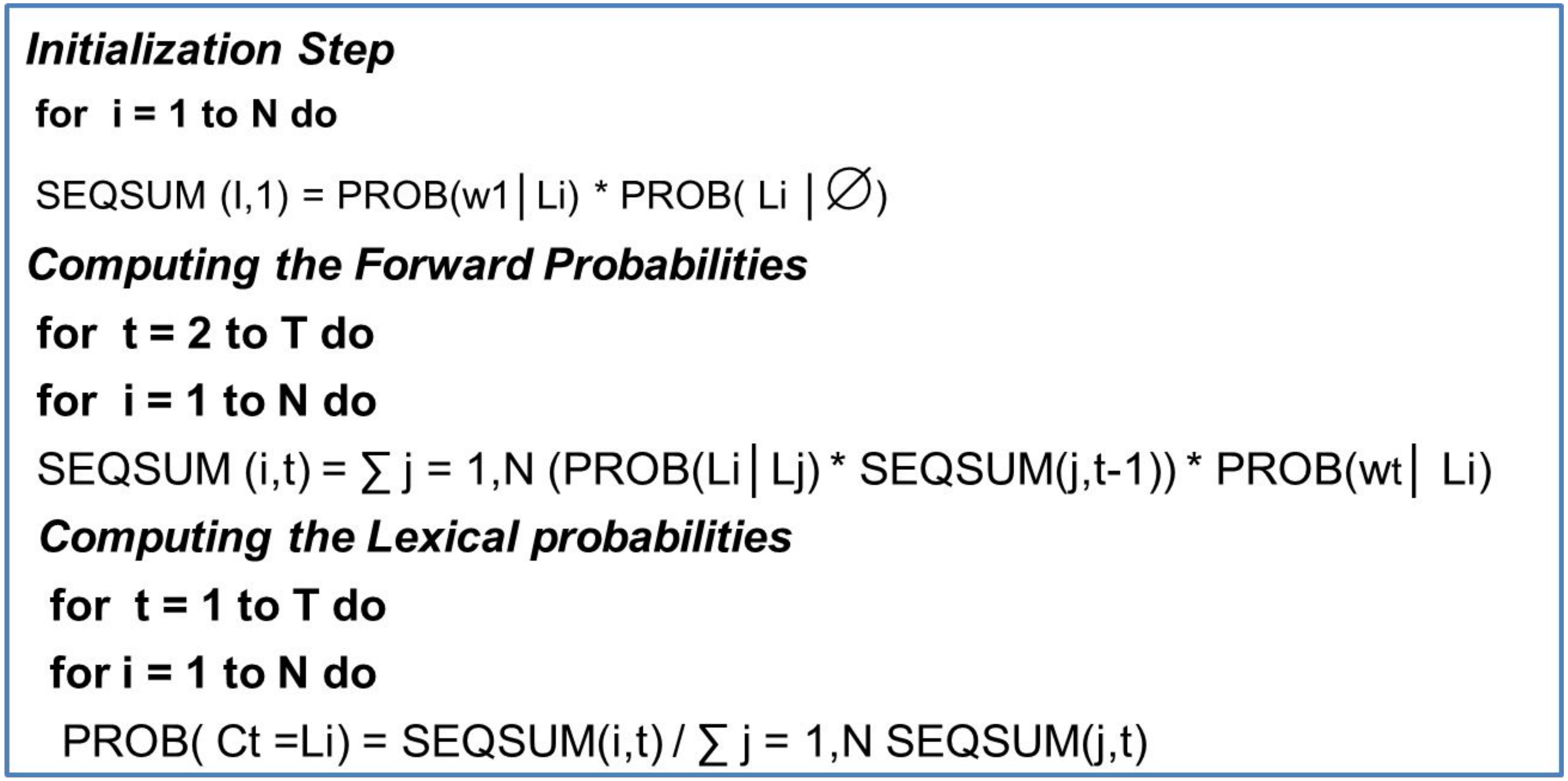
Likewise, the probability that flies is a verb would be 0.0012.

Rather than selecting the maximum score for each node at each stage of the algorithm (the Viterbi algorithm), we compute the sum of all scores.

We define the **forward probability** (Figure 7.14), written as $\alpha j(t)$, which is the probability of producing the words: $w_1 \ldots w_t$ and ending in state w_t/L_i :   **$\alpha j(t)$= PROB(w_t/L_i, $w_1,..,w_t$)**

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 Obtaining lexical probabilities



```
Initialization Step
for i = 1 to N do
    SEQSUM (I,1) = PROB(w1 | Li) * PROB( Li | ∅)
Computing the Forward Probabilities
for t = 2 to T do
    for i = 1 to N do
        SEQSUM (i,t) = ∑ j = 1,N (PROB(Li | Lj) * SEQSUM(j,t-1)) * PROB(wt | Li)
Computing the Lexical probabilities
for t = 1 to T do
    for i = 1 to N do
        PROB( Ct =Li) = SEQSUM(i,t) / ∑ j = 1,N SEQSUM(j,t)
```

Figure 6.10 The forward algorithm for computing the lexical probabilities

Chapter 6: Ambiguity Resolutions: Statistical methods

64 Obtaining lexical probabilities

Example: with sentence *The flies like flowers,* α2(3)= would be the sum of values computed for all sequences ending V(V is the second category) in position 3 given the input *The flies like.*

- Using the conditional probability we derive the probability that word **wt** is an instance of the lexical category Li:

$$\text{PROB}(w_t / L_i | w_1 \ldots w_t) \equiv$$
$$\text{PROB}(w_t / L_i, w_1 \ldots w_t) / \text{PROB}(w_1 \ldots w_t) \quad (1)$$

$$\text{PROB}(w_1, \ldots w_t) \equiv \Sigma_{j=1,N}\, \alpha_j(t) \qquad (2)$$

From (1) and (2):

$$\text{PROB}(w_t / L_i / w_1 \ldots w_t) \equiv \alpha_I(t) / \Sigma_{j=1,N}\, \alpha_j(t)$$

- **Bước khởi động:**

For i = 1 to N do /* N = 4 là ART, N, V, P */

SEQSUM(1,1) = PROB(the/ART)* PROB(ART/Θ) = 0.54*0.71= **0.3834**

SEQSUM(2,1) = PROB(the/N)* PROB(N/Θ) = 1/833*0.29= 0.0012 *0.29=**0.0003481**

SEQSUM(3,1) = PROB(the/V)* PROB(V/Θ)= 0.0 *10-4= **0.0**

SEQSUM(4,1) = PROB(the/P)* PROB(P/Θ)= 2/307*10-4 =0.0065147*10-4

Bước lặp

t = 2 (flies)

i = 1**: SEQSUM(1,2)** = (PROB(ART/ART)*SEQSUM(1,1)+PROB(ART/N)*SEQSUM(2,1)
+ PROB(ART/V)*SEQSUM(3,1) + PROB(ART/P)*SEQSUM(4,1))* PROB(flies/ART)
= (10-4*0.3834 + 10-4 * 0.0003481 + 0 +0.74*0.0065147*10-4)* 0= **0.0**

i = 2: **SEQSUM(2,2)**= (PROB(N/ART)*SEQSUM(1,1) + PROB(N/N)*SEQSUM(2,1)
+ PROB(N/V)*SEQSUM(3,1) + PROB(N/P)*SEQSUM(4,1))* PROB(flies/N)
= (1.0*0.3834 + 10-4*0.0003481 + 0.0 +0.26*0.0065147*10-4)* 0.025= **9.585*10-3**

i= 3: **SEQSUM(3,2)** = (PROB(V/ART)*SEQSUM(1,1)+ PROB(V/N)*SEQSUM(2,1)
+ PROB(V/V)*SEQSUM(3,1) + PROB(V/P)*SEQSUM(4,1))* PROB(flies/V)
= (10-4*0.3834 + 0.43*0.0003481 + 0.0 + 10-4*0.0065147*10-4)*0.076 = **1.4*10-5**

i = 4 : **SEQSUM (4,2)** = (PROB(P/ART)*SEQSUM(1,1)
+ PROB(P/N)*SEQSUM(2,1)
+ PROB(P/V)*SEQSUM(3,1)
+ PROB(P/P)*SEQSUM(4,1))* PROB(flies/P) = (……..)* 0.0 = **0.0**

t = 3 (like)

i = 1 **SEQSUM(1,3)** = (PROB(ART/ART)*SEQSUM(1,2)+ PROB(ART/N)*SEQSUM(2,2)
+ PROB(ART/V)*SEQSUM(3,2) + PROB(ART/P)*SEQSUM(4,2))* PROB(like/ART)
= (…….)* 0.0 = **0.0**

i = 2 **SEQSUM(2,3)** = (PROB(N/ART)*SEQSUM(1,2) + PROB(N/N)*SEQSUM(2,2)
+ PROB(N/V)*SEQSUM(3,2) + PROB(N/P)*SEQSUM(4,2))* PROB(like/N)
= (1*0.0 + 0.13*9.585*10-3 + 0.35*1.4*10-5)*0.012 = **1.5*10-5**

i = 3 **SEQSUM(3,3)** = (PROB(V/ART)*SEQSUM(1,2) + PROB(V/N)*SEQSUM(2,2)
+ PROB(V/V)*SEQSUM(3,2) + PROB(V/P)*SEQSUM(4,2))* PROB(like/V)
= (10-4*0.0 + 0.43*9.585*10-3 + 10-4*1.4*10-5 + 10-4*0.0)*0.1= **0.412*10-3**

i = 4 **SEQSUM(4,3)** = (PROB(P/ART)*SEQSUM(1,2) + PROB(P/N)*SEQSUM(2,2)
+ PROB(P/V)*SEQSUM(3,2) + PROB(P/P)*SEQSUM(4,2))* PROB(like/P)= **0.2867*10-3**

t= 4, (flower)

i = 1 SEQSUM(1,4) = (PROB(ART/ART)*SEQSUM(1,3) + PROB(ART/N)*SEQSUM(2,3) + PROB(ART/V)*SEQSUM(3,3) + PROB(ART/P)*SEQSUM(4,3))* PROB(flowers/ART)
= (…….)* 0.0 = **0.0**

i=2 SEQSUM(2,4) = (PROB(N/ART)*SEQSUM(1,3) + PROB(N/N)*SEQSUM(2,3) + PROB(N/V)*SEQSUM(3,3) + PROB(N/P)*SEQSUM(4,3))* PROB(flowers/N)
(1.0*0.0 + 0.13*1.5*10-5 + 0.35*0.412*10-3 + 0.26*0.2867*10-3)*2.4*10-3= **0.52*10-6**

i = 3 SEQSUM(3,4) = (PROB(V/ART)*SEQSUM(1,3) + PROB(V/N)*SEQSUM(2,3) + PROB(V/V)*SEQSUM(3,3) + PROB(V/P)*SEQSUM(4,3))* PROB(flowers/V)
(10-4*0.0 + 0.43*1.5*10-5 + 10-4*0.412*10-3 + 10-4*0.2867*10-3)*0.0533 = **3.475*10-7**

i = 4 SEQSUM(4,4) = (PROB(P/ART)*SEQSUM(1,3) + PROB(P/N)*SEQSUM(2,3) + PROB(P/V)*SEQSUM(3,3) + PROB(P/P)*SEQSUM(4,3))* PROB(flowers/P) = **0.0**

Tính xác suất từ vựng

1. **PROB(the/ART/the) = 0.9992**
   SEQSUM(1,1)/(SEQSUM(1,1)+SEQSUM(2,1)+SEQSUM(3,1)+SEQSUM(4,1))
   = 0.3834/(0.3834 + 0.0003481 + 0.0 + 0.00651*10-7)= **0.9992**
   PROB(the/N/the)= SEQSUM(2,1)/0.3837= 0.0003481/0.3837=9.02*10-4
   PROB(the/V/the) =SEQSUM(3,1)/0.3837= 0.0/0.3837= 0.0
   PROB(the/P/the) =SEQSUM(4,1)/0.3837 = 0.00651*10-4/0.3837=1.696*10-6
2. **PROB(flies/ART/the flies)** =0.0
   SEQSUM(1,2)/(SEQSUM(1,2)+SEQSUM(2,2)+SEQSUM(3,2)+SEQSUM(4,2)) = 0.0
   PROB(flies/N/the flies)= 0.9985
   SEQSUM(2,2)/9.599*10-3 = 9.585*10-3/9.599*103=**0.9985**
   PROB(flies/V/the flies) = SEQSUM(3,2)/9.599*10-3 = 1.4*10-5/9.599*10-3= 1.45*10-3
   PROB(flies/P/the flies) = SEQSUM(4,2)/9.599*10-3= 0.0
3. **PROB(like/ART/the flies like)** = 0.0
   SEQSUM(1,3)/(SEQSUM(1,3)+SEQSUM(2,3)+SEQSUM(3,3)+SEQSUM(4,3))=
   0.0/0.71*10-3= 0.0
   PROB(like/N/the flies like) = SEQSUM(2,3)/0.71*10-3 =1.5*10-5/0.71*10-3=0.0211
   PROB(like/V/the flies like) =0.58
   SEQSUM(3,3)/0.71*10-3 = 0.412*10-3/0.71*10-3= **0.58**
   PROB(like/P/the flies like)=SEQSUM(4,3)/0.71*10-3= 0.2867*10-3/0.71*10-3=0.4038

4-**PROB(flowers/ART/the flies like flowers)** =
   SEQSUM(1,4)/(SEQSUM(1,4)+SEQSUM(2,4)+SEQSUM(3,4)+SEQSUM(4,4)) = 0.0
   PROB(flowers/N/the flies like flowers)= 0.604
   SEQSUM(2,4)/0.86*10-6 = 0.52*10-6/0.86*10-6= **0.604**
   PROB(flowers/V/the flies like flowers)=
   SEQSUM(3,4)/0.86*10-6= 0.3475*10-6/0.86*10-6 =0.404
   PROB(flowers/P/the flies like flowers) = SEQSUM(4,4)/0.86*10-6=0.0

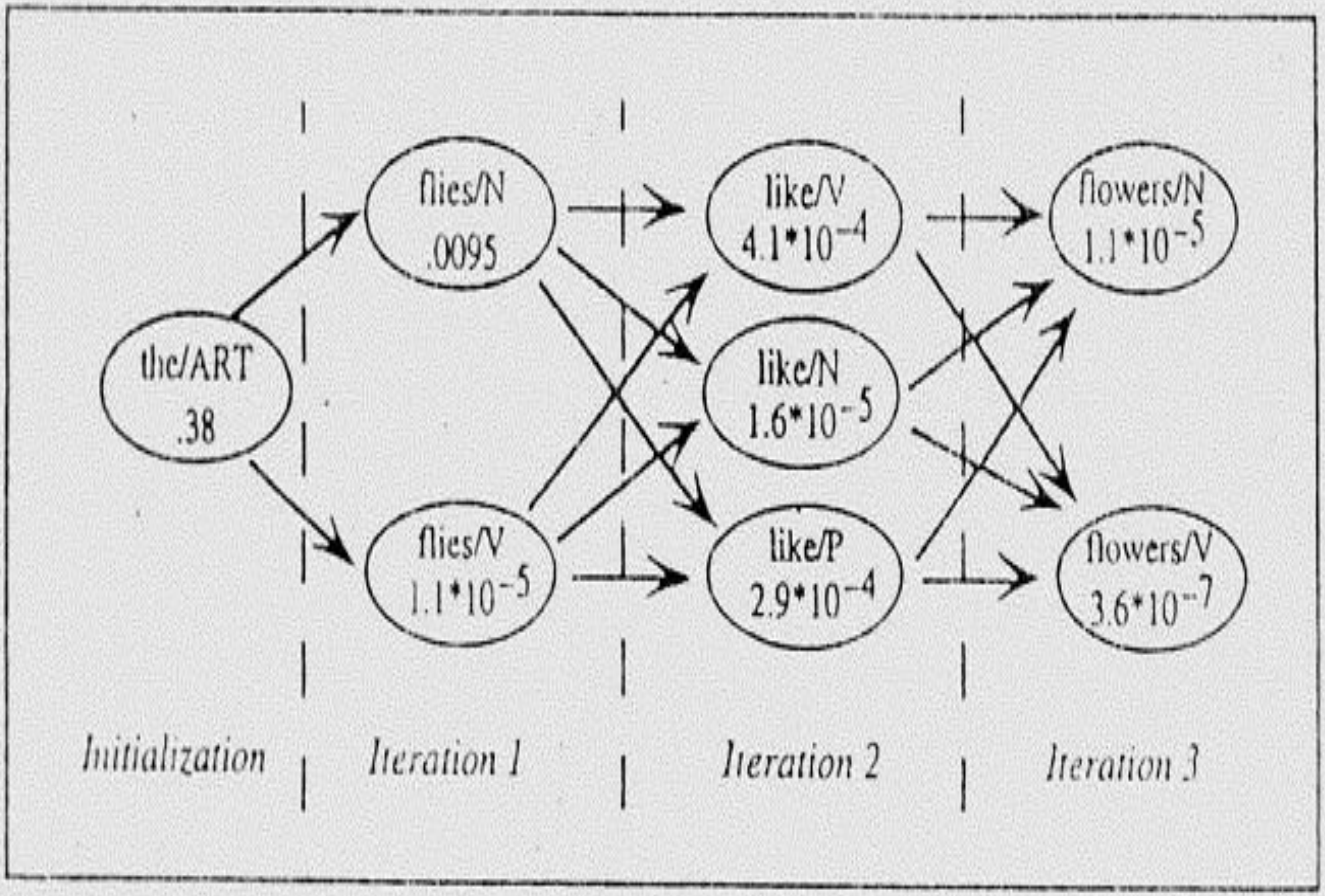
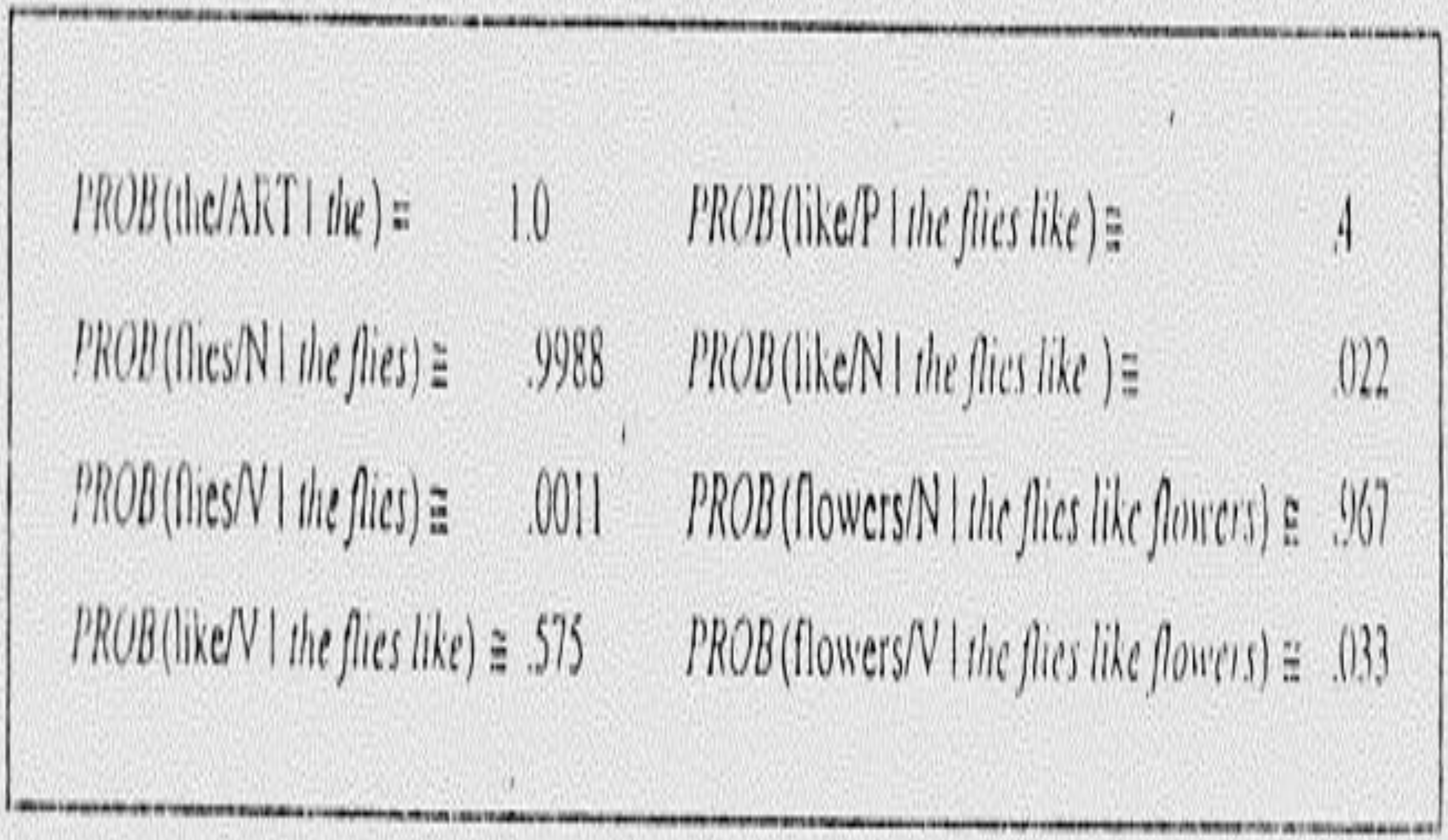


Figure 6.11 Computing the sums of the probabilities of the sequences



$PROB(\text{the/ART} \mid \textit{the}) \cong$	1.0	$PROB(\text{like/P} \mid \textit{the flies like}) \cong$	.4
$PROB(\text{flies/N} \mid \textit{the flies}) \cong$	.9988	$PROB(\text{like/N} \mid \textit{the flies like}) \cong$	.022
$PROB(\text{flies/V} \mid \textit{the flies}) \cong$	.0011	$PROB(\text{flowers/N} \mid \textit{the flies like flowers}) \cong$	.967
$PROB(\text{like/V} \mid \textit{the flies like}) \cong$	.575	$PROB(\text{flowers/V} \mid \textit{the flies like flowers}) \cong$	.033

Figure 6.12 Context dependent estimates for lexical categories in the sentence: The flies like flowers

Chapter 6: Ambiguity Resolutions: Statistical methods

6.4 Obtaining lexical probabilities

- We could also consider **the backward probability $\beta_i(t)$**, the probability of producing the sequence $w_t, \ldots, W_T$ beginning from the state wt/Lj.

A better method of estimating the lexical probability for word w_t:

$$PROB(w_t / L_i) = \alpha_i(t) * \beta_i(t) / \Sigma_{j:=1,N} \; \alpha_j(t) * \beta_j(t)$$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.5 Language Modeling : *Introduction*

Application

Today's goal: assign a probability to a sentence

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- + Summarization, question-answering, etc., etc.!!

6.5 Language Modeling : *Introduction*

- Goal: compute the probability of a sentence or sequence of words:

  $$P(W) = P(w_1,w_2,w_3,w_4,w_5\ldots w_n)$$

- Related task: probability of an upcoming word:

  $$P(w_5|w_1,w_2,w_3,w_4)$$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2\ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

6.5 Language Modeling : *Introduction*

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

Example:

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

- **Markov Assumption**

Bigram model

$$P(w_i \mid w_1 w_2 \rightleftharpoons w_{i-1}) \approx P(w_i \mid w_{i-1})$$

6.5 Language Modeling : *Introduction*

Zeros

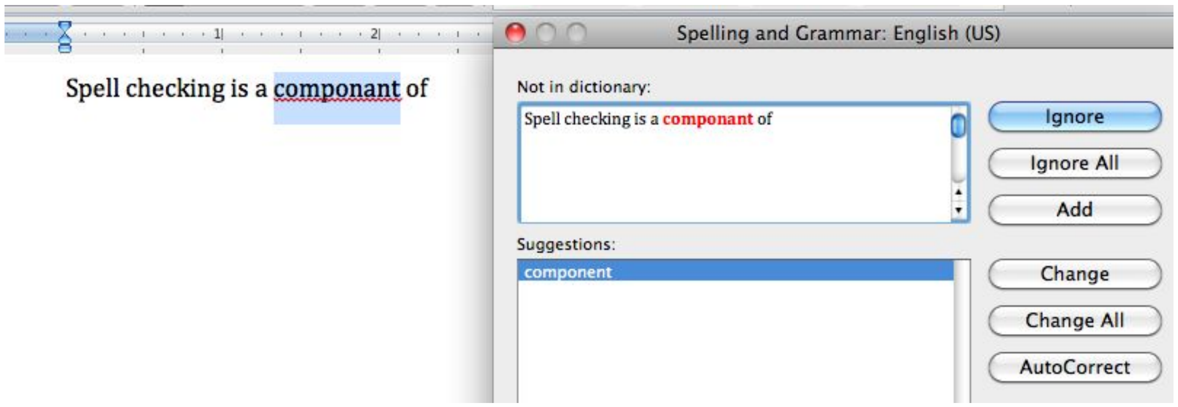
- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

  P(“offer” | denied the) = 0

- Test set
  … denied the offer
  … denied the loan

6.6 Applications for spelling correction

Word processing



Spell checking is a componant of

Spelling and Grammar: English (US)

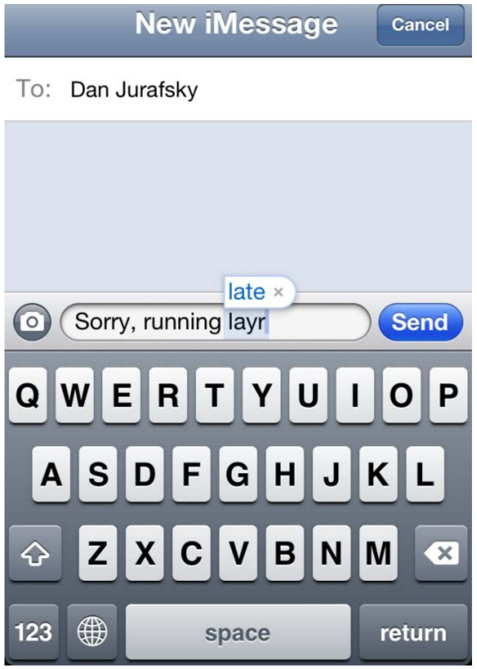
Not in dictionary:

Spell checking is a componant of

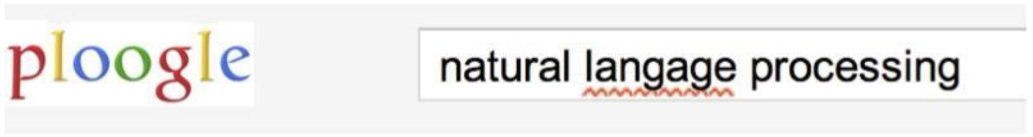
Suggestions:

component

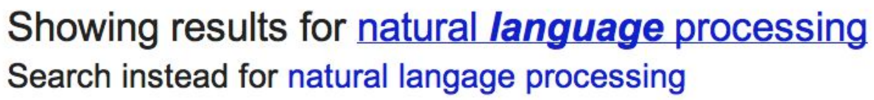
Phones



Web search



natural langage processing



Showing results for natural ***language*** processing
Search instead for natural langage processing

Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    - hte□the
  - Suggest a correction
  - Suggestion lists

Non-word spelling errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better
- Non-word spelling error correction:
  - Generate ***candidates***: real words that are similar to error
  - Choose the one which is best:
    - Shortest weighted edit distance
    - Highest noisy channel probability

Non-word spelling error example

`acress`

Non-word spelling errors

- Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021 P(whose|actress) = .0010
- P(across|versatile) =.000021 P(whose|across) = .000006

- P("versatile actress whose") = .000021*.0010 = 210 x10^{-10}
- P("versatile across whose") = .000021*.000006 = 1 x10^{-10}

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

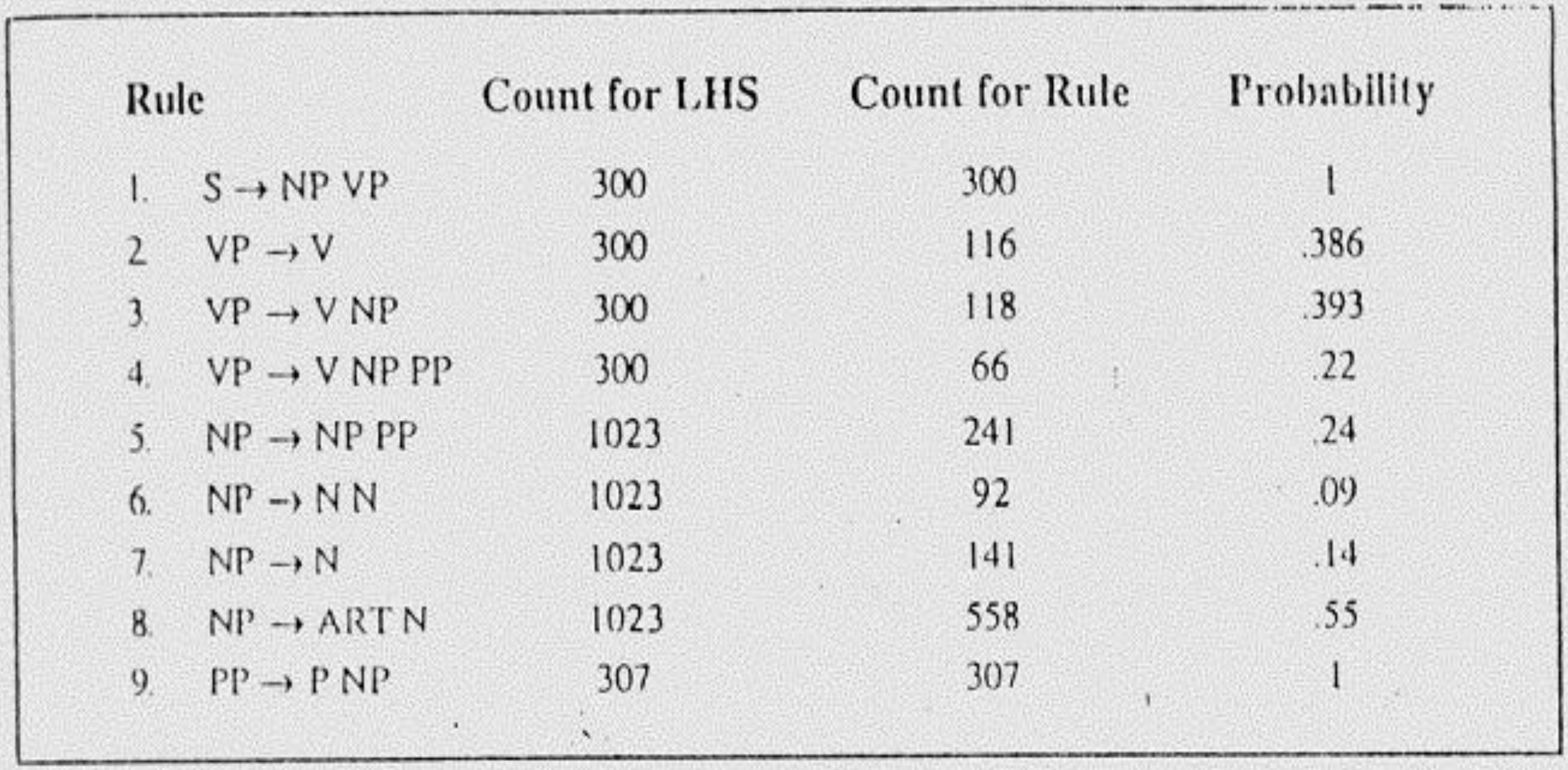
- Context free grammar can be generated to the probabilistic case.

- Counting the number of times each rule is used in a corpus containing parsed sentences and use this to estimate the probability of each rule being used.
- For instance, m rules R1,.., Rm with left-hand side C.
- Estimating the probability using the rule Rj to derive C:

$$\text{PROB}(R_j / C) \cong \text{count}(\#\text{ times } R_j \text{ used}) / \Sigma_{i=1,m}(\#\text{ times } R_i \text{ used})$$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars



Rule	Count for LHS	Count for Rule	Probability
1. S → NP VP	300	300	1
2. VP → V	300	116	.386
3. VP → V NP	300	118	.393
4. VP → V NP PP	300	66	.22
5. NP → NP PP	1023	241	.24
6. NP → N N	1023	92	.09
7. NP → N	1023	141	.14
8. NP → ART N	1023	558	.55
9. PP → P NP	307	307	1

Figure 6.13 A simple probabilistic grammar

The grammar in Figure 7.13 shows probabilities for CFG

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

- We must assume that the probability of a constituent being derived by a rule Rj is independent of how the constituent used as sub-constituent.

Inside probability PROB(w_{ij}/C):

The probability that a constituent C generates a sequence of words $w_i, \ldots w_j$, written as w_{ij}.

This type of probability is called because it assigns a probability to the word sequence inside the constituent.

Consider how to derive inside probabilities

- For lexical categories that these are exactly lexical generation probabilities in Figure 6.3, instance PROB (flower / N) that is inside probability

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

- For non-lexical categories: using such lexical generation probabilities, we can derive the probability of a constituent.

Example:

Deriving a probability that the constituent NP generates a sequence *a flower* as in figure 7.18.

The grammar in Figure 7.17, there are two rules NP which can generate two words, so the probability of NP generating *a flower* can be derived as follows:

PROB (a flower / NP) = PROB (rule_8 / NP) * PROB (a / ART) * PROB (flower / N) + PROB (rule_6 / NP) * PROB (a / ART) * PROB (flower / N) = 0.55 * 0.36 * 0.6 + 0.9 * 0.0001 * 0.06 = 0.012

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

This probability can be then used to compute a probability of large constituents.

For instance, the probability of generating the words *A flower wilted* from S could be computed by summing the probabilities from each of the possible trees shown in the Figure 7.19.

Note that in Figure 7.19 there are three trees, and the first two differ only in the derivation of *a flower as NP.*

Thus the probability of a flower wilted is:

PROB (a flower wilted/ S) = PROB (Rule1 / S) * PROB (a flower / NP) * PROB (wilted / VP) + PROB (Rule1 / S) * PROB (a / NP) * PROB (flower wilted / VP).

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

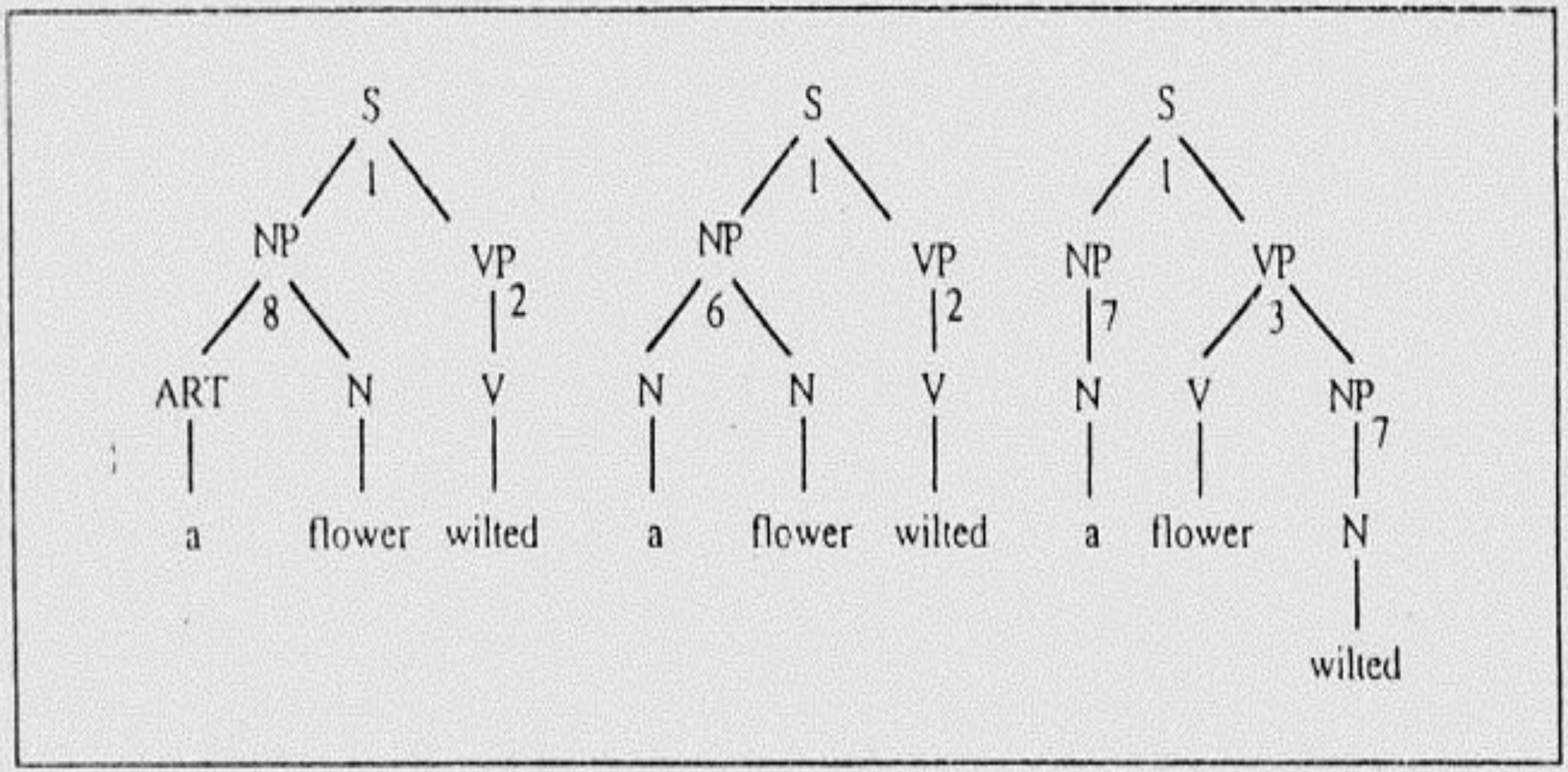


Figure 6.14 The three possible ways to generate a flower wilted as an S

Chapter 6: Ambiguity Resolutions: Statistical methods

6.7 Probabilistic Context – Free grammars

Using this method, the probability that a given sentence will be generated by the grammar can be computed efficiently

The goal of probability parsing method is to find the most likely parse rather than overall probability of the given sentence.

The probability of each constituent is computed from the probability of its sub-constituents and the probability of the rule used.

For instance, entry E of category C using a rule i with n sub-constituents corresponding to entries E_1 , …, $E_{n,}$ then:

PROB (E) = PROB (Rule i / C) * PROB (E_1) * … * PROB (E_n)

	NP 425
	1 N 422 0.14
NP 424	
1 N 417	
2 N 422 0.00011	
NP 423	
1 ART 416	
2 N 422 0.54	
S 421	
1 NP 418	
2 VP 420 3.2×10^{-8}	
NP 418	
1 N 417 0.0018	**VP 420 0.0018**
N 417 0.001	**N 422 0.00011**
ART 416 0.99	**V 410 0.00047**

Figure. 6.15 The full chart for a flower that is as NP

6.7 Probabilistic Context – Free grammars

Machine translation

The idea behind statistical machine translation comes from information theory.

-A document is translated according to the probability distribution $p(e|f)$, that a string e in the target language (for example, English) is the translation of a string f in the source language (for example, French).

- The problem of modeling the probability distribution $p(e|f)$ has been approached in a number of ways. One approach which lends itself well to computer implementation is to apply Bayes Theorem, that is $p(e|f) \alpha\, p(f|e)p(e)$, where the translation model $p(f|e)$ is the probability that the source string is the translation

6.7 Probabilistic Context – Free grammars

Machine translation

of the target string, and the language model p(e) is the probability of seeing that target language string. This decomposition is attractive as it splits the problem into two subproblems. Finding the best translation ẽ is done by picking up the one that gives the highest probability:

$$\tilde{e} = \operatorname*{argmax}_{e \in e^*} p(e|f) = \operatorname*{argmax}_{e \in e^*} p(f|e)p(e)$$

Chapter 6: Ambiguity Resolutions: Statistical methods

6.8 Best –First parsing (study oneself)

6.9 Word Similarity

- **Synonymy**: a binary relation
  - Two words are either synonymous or not
- **Similarity** (or **distance**): a looser metric
  - Two words are more similar if they share more features of meaning
- Similarity is properly a relation between **senses**
  - The word "bank" is not similar to the word "slope"
  - Bank^1 is similar to fund^3
  - Bank^2 is similar to slope^5
- But we'll compute similarity over both words and senses

Why word similarity

- Information retrieval
- Question answering
- Machine translation
- Natural language generation
- Language modeling
- Automatic essay grading
- Plagiarism detection
- Document clustering

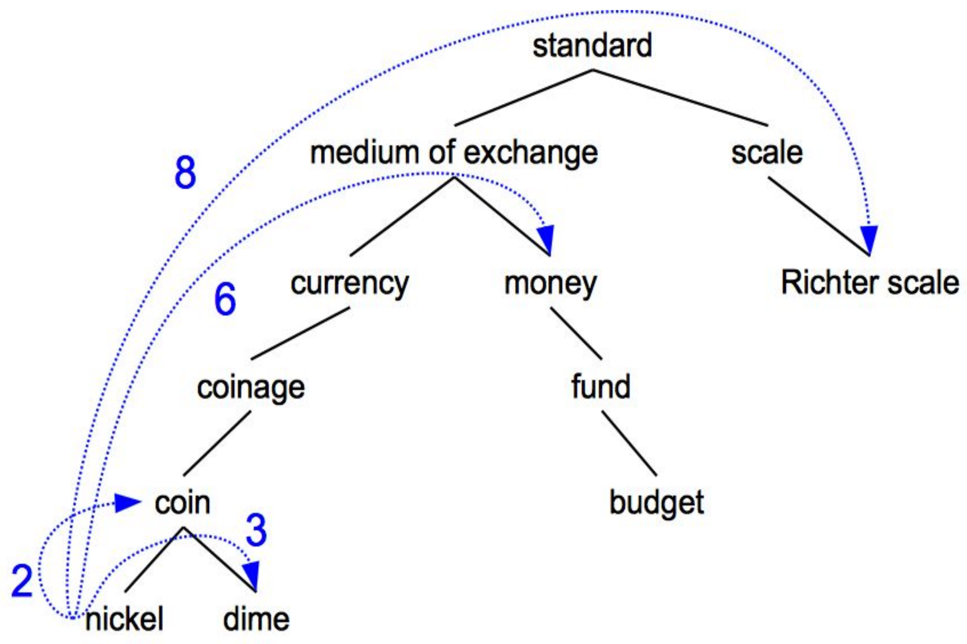
Word similarity and word relatedness

- We often distinguish **word similarity** from **word relatedness**
  - **Similar words**: near-synonyms
  - **Related words**: can be related any way
    - car, bicycle: **similar**
    - car, gasoline: **related**, not similar

Two classes of similarity algorithms

- Thesaurus-based algorithms
  - Are words "nearby" in hypernym hierarchy?
  - Do words have similar glosses (definitions)?
- Distributional algorithms
  - Do words have similar distributional contexts?

Path based similarity



- Two concepts (senses/synsets) are similar if they are near each other in the thesaurus hierarchy
  - =have a short path between them
  - concepts have path 1 to themselves

Refinements to path-based similarity

- pathlen(c_1,c_2) = 1 + number of edges in the shortest path in the hypernym graph between sense nodes c_1 and c_2
- ranges from 0 to 1 (identity)
- simpath(c_1,c_2) = $\frac{1}{\text{pathlen}(c_1, c_2)}$
- wordsim(w_1,w_2) = $\max_{c_1 \in \text{senses}(w_1), c_2 \in \text{senses}(w_2)} \text{sim}(c_1,c_2)$

Summary: thesaurus-based similarity

$$\text{sim}_{\text{path}}(c_1, c_2) = \frac{1}{pathlen(c_1, c_2)}$$

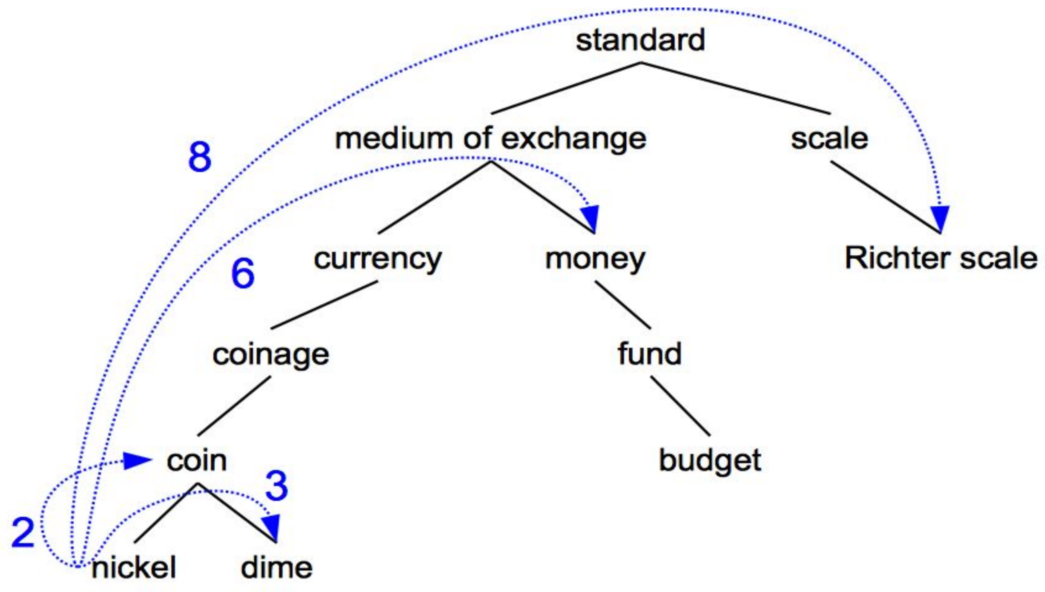
$$\text{sim}_{\text{resnik}}(c_1, c_2) = -\log P(LCS(c_1, c_2)) \quad \text{sim}_{\text{lin}}(c_1, c_2) = \frac{2\log P(LCS(c_1, c_2))}{\log P(c_1) + \log P(c_2)}$$

$$\text{sim}_{\text{jiangconrath}}(c_1, c_2) = \frac{1}{\log P(c_1) + \log P(c_2) - 2\log P(LCS(c_1, c_2))}$$

$$\text{sim}_{eLesk}(c_1, c_2) = \sum_{r,q \in RELS} \text{overlap}(\text{gloss}(r(c_1)), \text{gloss}(q(c_2)))$$

Example: path-based similarity

$$\text{simpath}(c_1,c_2) = 1/\text{pathlen}(c_1,c_2)$$



simpath(*nickel,coin*) = 1/2 = .5

simpath(*fund,budget*) = 1/2 = .5

simpath(*nickel,currency*) = 1/4 = .25

simpath(*nickel,money*) = 1/6 = .17

simpath(*coinage,Richter scale*) = 1/6 = .17

Problems with thesaurus-based meaning

- We don’t have a thesaurus for every language
- Even if we do, they have problems with **recall**
  - Many words are missing
  - Most (if not all) phrases are missing
  - Some connections between senses are missing
  - Thesauri work less well for verbs, adjectives
    - Adjectives and verbs have less structured hyponymy relations

Distributional models of meaning

- Also called vector-space models of meaning.
- Offer much higher recall than hand-built thesauri.
- Although they tend to have lower precision

- Zellig Harris (1954): "***oculist** and **eye-doctor** … occur in almost the same environments….*
  If A and B have almost identical environments we say that they are synonyms.
- Firth (1957): "*You shall know a word by the company it keeps!*"

Reminder: Term-document matrix

- Each cell: count of term *t* in a document *d*: $\text{tf}_{t,d}$:
  - Each document is a count vector in $\mathbb{N}^V$: a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Reminder: Term-document matrix

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

- Each word is a $\boxed{\text{count vector}}$ in $\mathbb{N}^D$: a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	**37**	**58**	**1**	**5**
clown	6	117	0	0

The words in a term-document matrix

- Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	**37**	**58**	**1**	**5**
clown	6	117	0	0

The Term-Context matrix

- Instead of using entire documents, use smaller contexts
  - Paragraph
  - Window of 10 words
- A word is now defined by a vector over counts of context words

Sample contexts: 20 words (Brown corpus)

- equal amount of sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of clove and nutmeg,
- on board for their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened to that of
- of a recursive type well suited to programming on the **digital** computer. In finding the optimal R-stage policy from that of
- substantially affect commerce, for the purpose of gathering data and **information** necessary for the study authorized in the first section of this

Term-context matrix for word similarity

- Two **words** are similar in meaning if their context vectors are similar

	aardvark	computer	data	pinch	result	sugar	…
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Should we use raw counts?

- For the term-document matrix
  - We used tf-idf instead of raw term counts
- For the term-context matrix
  - Positive Pointwise Mutual Information (PPMI) is common

Pointwise Mutual Information

- **Pointwise mutual information**:
  - Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

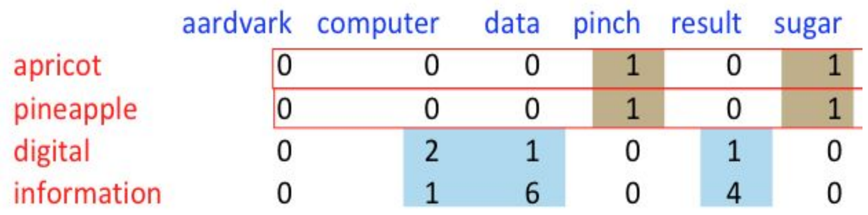
- **PMI between two words**:
  - Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1,word_2)}{P(word_1)P(word_2)}$$

- **Positive PMI between two words**
  - Replace all PMI values less than 0 with zero

Computing PPMI on a term-context matrix

- Matrix F with W rows (words) and C columns (contexts)
- f_{ij} is # of times w_i occurs in context c_i



	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

Count(w,context)

	computer	data	pinch	result	sugar
apricot	0	0	1	0	1
pineapple	0	0	1	0	1
digital	2	1	0	1	0
information	1	6	0	4	0

p(w=information,c=data) = 6/19 = .32

p(w=information) = 11/19 = .58

p(c=data) = 7/19 = .37

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N} \qquad p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

p(w,context) **p(w)**

	computer	data	pinch	result	sugar	p(w)
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

- pmi(information,data) = $\log_2$ (.32 / (.37*.58)) = .58

(.57 using full precision)

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

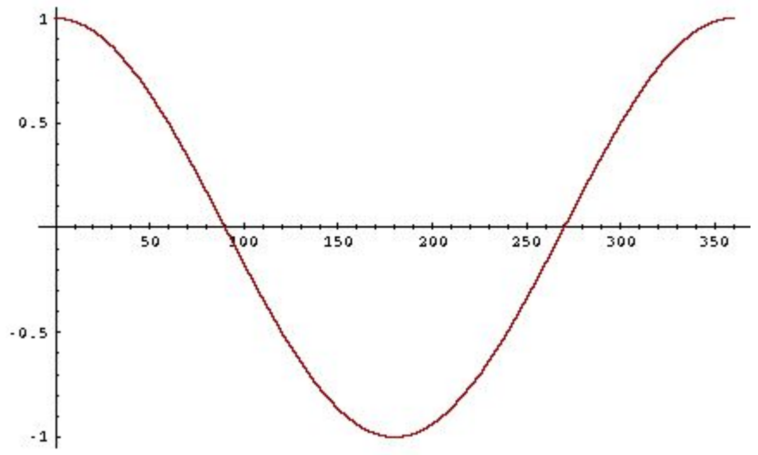
Reminder: cosine for computing similarity

$$\cos(\vec{v},\vec{w})=\frac{\overbrace{\vec{v}\bullet\vec{w}}^{\text{Dot product}}}{|\vec{v}||\vec{w}|}=\overbrace{\frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|}}^{\text{Unit vectors}}=\frac{\sum_{i=1}^{N}v_i w_i}{\sqrt{\sum_{i=1}^{N}v_i^2}\sqrt{\sum_{i=1}^{N}w_i^2}}$$

v_i is the PPMI value for word *v* in context *i*
w_i is the PPMI value for word *w* in context *i.*

Cos(*v,w*) is the cosine similarity of *v* and *w*

Cosine as a similarity metric



- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal

- Raw frequency or PPMI are non-negative, so cosine range 0-1

$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

	large	data	computer
apricot	1	0	0
digital	0	1	2
information	1	6	1

Which pair of words is more similar?

cosine(apricot,information) = $\frac{1+0+0}{\sqrt{1+0+0}\sqrt{1+36+1}} = \frac{1}{\sqrt{38}} = .16$

cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$

cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{1+0+0}\sqrt{0+1+4}} = 0$

Other possible similarity measures

$$\text{sim}_{\text{cosine}}(\vec{v},\vec{w}) = \frac{\vec{v}\cdot\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\sum_{i=1}^{N} v_i \times w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}\max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N}\min(v_i,w_i)}{\sum_{i=1}^{N}(v_i+w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

Evaluating similarity (the same as for thesaurus-based)

- Intrinsic Evaluation:
  - Correlation between algorithm and human word similarity ratings
- Extrinsic (task-based, end-to-end) Evaluation:
  - Spelling error detection, WSD, essay grading
  - Taking TOEFL multiple-choice vocabulary tests

Levied is closest in meaning to which of these:
imposed, believed, requested, correlated

EXERCISE OF CHAPTER 6

1. Hand simulate the Viterbi algorithm using the data and probability estimates in Figures7.4- 7.6 on the sentence ***Flower flowers like flowers.*** Draw transition network as in Figure 7.10-7.12 for the problem, and identify what part of speech the algorithm identifies for each word.
2. Using the bigram and lexical generation probabilities given in this chapter, calculate the word probabilities using the forward algorithm for the sentence ***The a flies like flower*** (involving a very rare use of the word ***a*** as a noun, as in the a flies, the b flies, and so on). Remember to use 0.0001 as a probability for any bigram not in the table. Are the results you get reasonable ?. If not, what is the problem and how might it be fixed ?.

EXERCISE OF CHAPTER 6

3. Consider an extended version of Grammar 7.17 with the additional rule: 10. VP → V PP

The revised rule probabilities are shown here (Any not mentioned are the same as in Grammar 7.17):

VP	→	V	0.32	VP	→	V NP PP	0.20
VP	→	V NP	0.33	VP	→	V PP	0.15

In addition, the following bigram probabilities differ from those in Figure 7.4:

PROB(N/V) = 0.53   PROB(ART/V) = 0.32   PROB(P/V) = 0.15

a) Hand simulate (or implement) the forward algorithm on ***Fruit flies like birds*** to produce the lexical probabilities.

b) Draw out the full chart for ***Fruit flies like birds,*** showing the probabilities of each constituent.

EXERCISE OF CHAPTER 6

- Specify PMI between two words, Positive PMI between two words in the below table

	aardvark	computer	data	pinch	result	sugar
Apricot	0	0	0	0	1	0
Pineapple	0	0	0	0	1	0
Digital	0	2	1	0	1	0
Information	0	1	6	0	4	0