

COMPUTING WITH PROBABILITIES

Linguistics 684.02

JOINT PROBABILITIES

Word 2

Word 1

	a	b	c	d	e
a	$p(a,a)$	$p(a,b)$	$p(a,c)$	$p(a,d)$	$p(a,e)$
b	$p(b,a)$	$p(b,b)$	$p(b,c)$	$p(b,d)$	$p(b,e)$
c	$p(c,a)$	$p(c,b)$	$p(c,c)$	$p(c,d)$	$p(c,e)$
d	$p(d,a)$	$p(d,b)$	$p(d,c)$	$p(d,d)$	$p(d,e)$
e	$p(e,a)$	$p(e,b)$	$p(e,c)$	$p(e,d)$	$p(e,e)$

JOINT PROBABILITIES

Word 2

Word 1

	a	b	c	d	e
a	$p(a,a)$	$p(a,b)$	$p(a,c)$	$p(a,d)$	$p(a,e)$
b	$p(b,a)$	$p(b,b)$	$p(b,c)$	$p(b,d)$	$p(b,e)$
c	$p(c,a)$	$p(c,b)$	$p(c,c)$	$p(c,d)$	$p(c,e)$
d	$p(d,a)$	$p(d,b)$	$p(d,c)$	$p(d,d)$	$p(d,e)$
e	$p(e,a)$	$p(e,b)$	$p(e,c)$	$p(e,d)$	$p(e,e)$

Sum of colored cells is 1

All these possibilities are disjoint

JOINT COUNTS

Word 2

Word 1

	a	b	c	d	e
a	a,a	a,b	a,c	a,d	a,e
b	b,a	b,b	b,c	b,d	b,e
c	c,a	c,b	c,c	c,d	c,e
d	d,a	d,b	d,c	d,d	d,e
e	e,a	e,b	e,c	e,d	e,e

This adds up to the total number of bigrams in the sample

JOINT PROBABILITIES

Word 2

Word 1

	a	b	c	d	e
a	$\frac{ a, a }{N}$	$\frac{ a, b }{N}$	$\frac{ a, e }{N}$
b
c
d
e	$\frac{ e, a }{N}$	$\frac{ e, e }{N}$

Divide
everything
by grand
total (N)
Yields joint
probs

JOINT COUNTS

Word 2

Word 1

	a	b	c	d	e	Sum
a	a,a	a,b	a,c	a,d	a,e	a,*
b	b,a	b,b	b,c	b,d	b,e	b,*
c	c,a	c,b	c,c	c,d	c,e	c,*
d	d,a	d,b	d,c	d,d	d,e	d,*
e	e,a	e,b	e,c	e,d	e,e	e,*
Sum	*,a	*,b	*,c	*,d	*,e	*,*

JOINT COUNTS

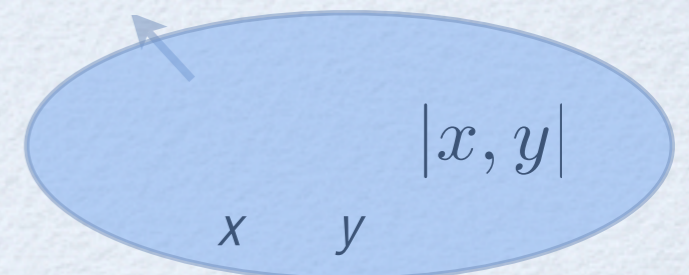
Word 2

Word 1

	a	b	c	d	e	Sum
a	a,a	a,b	a,c	a,d	a,e	a,*
b	b,a	b,b	b,c	b,d	b,e	b,*
c	c,a	c,b	c,c	c,d	c,e	c,*
d	d,a	d,b	d,c	d,d	d,e	d,*
e	e,a	e,b	e,c	e,d	e,e	e,*
Sum	*,a	*,b	*,c	*,d	*,e	*,*

$$\sum_y |b, y|$$

$$\sum_x |x, b|$$



MARGINAL PROBABILITIES

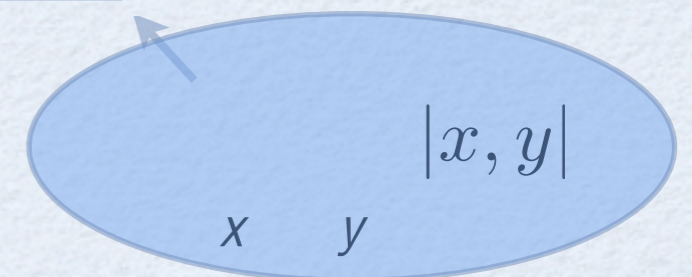
Word 2

	a	b	c	d	e	Sum
a	a,a	a,b	a,c	a,d	a,e	a,*
b	b,a	b,b	b,c	b,d	b,e	b,*
c	c,a	c,b	c,c	c,d	c,e	c,*
d	d,a	d,b	d,c	d,d	d,e	d,*
e	e,a	e,b	e,c	e,d	e,e	e,*
Sum	*,a	*,b	*,c	*,d	*,e	*,*

Word 1

$$\sum_y |b, y|$$

$$\sum_x |x, b|$$



MARGINS

$$p(b, *) = \frac{\sum_y |b, y|}{\sum_x \sum_y |x, y|}$$

$$p(*, b) = \frac{\sum_x |x, b|}{\sum_x \sum_y |x, y|}$$

SUFFICIENT STATISTICS

- The table of counts contains all the information needed to compute probabilities for all bigrams in the sample.
- You could have a cube for trigrams, and hypercubes for larger n-grams.
- The counts are called the sufficient statistics

CONDITIONAL

Word 2

Word 1

	a	b	c	d	e	
a	$ a,a $	$ a,b $	$ a,c $	$ a,d $	$ a,e $	$\sum_y a,y $
b	$ b,a $	$ b,b $	$ b,c $	$ b,d $	$ b,e $	$\sum_y b,y $
c	$ c,a $	$ c,b $	$ c,c $	$ c,d $	$ c,e $	$\sum_y c,y $
d	$ d,a $	$ d,b $	$ d,c $	$ d,d $	$ d,e $	$\sum_y d,y $
e	$ e,a $	$ e,b $	$ e,c $	$ e,d $	$ e,e $	$\sum_y e,y $

CONDITIONAL

Word 2

Word 1

	a	b	c	d	e	
a	$p(a a)$	$p(b a)$	$p(c a)$	$p(d a)$	$p(e a)$	1
b	$p(a b)$	$p(b b)$	$p(c b)$	$p(d b)$	$p(e b)$	1
c	$p(a c)$	$p(b c)$	$p(c c)$	$p(d c)$	$p(e c)$	1
d	$p(a d)$	$p(b d)$	$p(c d)$	$p(d d)$	$p(e d)$	1
e	$p(a e)$	$p(b e)$	$p(c e)$	$p(d e)$	$p(e e)$	1

CONDITIONAL

Word 2

Word 1

	a	b	c	d	e
a	a,a	a,b	a,c	a,d	a,e
b	b,a	b,b	b,c	b,d	b,e
c	c,a	c,b	c,c	c,d	c,e
d	d,a	d,b	d,c	d,d	d,e
e	e,a	e,b	e,c	e,d	e,e

CONDITIONAL

Word 2

Word 1

	a	b	c	d	e
a	$p(a,a *,a)$	$p(a,b *,b)$	$p(a,c *,c)$	$p(a,d *,d)$	$p(a,e *,e)$
b	$p(b,a *,a)$	$p(b,b *,b)$	$p(b,c *,b)$	$p(b,d *,d)$	$p(b,e *,e)$
c	$p(a *,a)$	$p(c,b *,b)$	$p(c,c *,c)$	$p(c,d *,d)$	$p(c,e *,e)$
d	$p(d,a *,a)$	$p(d,b *,b)$	$p(d,c *,c)$	$p(d,d *,d)$	$p(d,e *,e)$
e	$p(e,a *,a)$	$p(e,b *,b)$	$p(e,c *,c)$	$p(e,d *,d)$	$p(e,e *,e)$
	1	1	1	1	1

PROGRAMMING IT

- Collect bigrams and count them
- Calculate probabilities from counts

```
Python Shell
Python 2.5.2 (r252:60911, Feb 22 2008, 07:57:53)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
Type "copyright", "credits" or "license()" for more information.

*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface.  This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****

IDLE 1.2.2
>>> "My dear fellow said Sherlock Holmes"
'My dear fellow said Sherlock Holmes'
>>> "My dear fellow said Sherlock Holmes".split()
['My', 'dear', 'fellow', 'said', 'Sherlock', 'Holmes']
>>> for word in "My dear fellow said Sherlock Holmes".split():
    print word

My|
dear
fellow
said
Sherlock
Holmes
>>>
```

DIVERSION INTO NLTK

- Run through chapter 2 of nltk book

WHAT WE NEED TO DO

- Arrange things so we get bigrams of strings s_i, s_j
- Convert strings into pairs of indices (i, j)
- Add 1 to the $a(i, j)$ for each time we see i, j

GET BIGRAMS

- Assume we can get a sequence of words
- Pair it up with a shifted version of itself
- These are the bigrams

```
import nltk

ws1 = nltk.corpus.gutenberg.words('blake-songs.txt')
ws2 = nltk.corpus.gutenberg.words('blake-songs.txt')

bigrams = zip(ws1,ws2[1:])

words = set(ws1)
bigrams = set(bigrams)

print len(words),len(bigrams), len(words)*len(words) - len(bigrams)
```

```
Python 2.5.2 (r252:60911, Feb 22 2008, 07:57:53)
[GCC 4.0.1 (Apple Computer, Inc. build 5363)] on darwin
Type "copyright", "credits" or "license()" for more information.
```

```
*****
Personal firewall software may warn about the connection IDLE
makes to its subprocess using this computer's internal loopback
interface. This connection is not visible on any external
interface and no data is sent to or received from the Internet.
*****
```

```
IDLE 1.2.2
>>> import bigrams
1589 5052 2519869
>>>
```

COMPUTING IT

- The table has $2.5e6$ empty cells and 5052 non-empty cells.

SPARSITY

- If there are 300 different words, there are 90,000 cells, most of which will be empty
- Our sums would be the same if we ignored all the zero entries. We don't really need all those cells.

SPARSITY

- Store only the non-zero entries

```
import nltk

ws1 = nltk.corpus.gutenberg.words('blake-songs.txt')
ws2 = nltk.corpus.gutenberg.words('blake-songs.txt')

bigrams = zip(ws1,ws2[1:])

unique_words = set(ws1)
unique_bigrams = set(bigrams)

count = {}
for bigram in bigrams:
    if bigram not in count:
        count[bigram] = 0
    count[bigram] = count[bigram]+1

for (w1,w2) in unique_bigrams:
    print w1,w2,count[w1,w2]
```