Lecture 28

408/508 *Computational Techniques for Linguists*

Reminder

- You should be working on your term project
- *Short write-up etc. due end of next week*
- Please fill out the course survey!

Today's Topic

- We've seen `nltk.FreqDist(`*`words`*`)` previously
  - counts how many times each word appears in *words*
- Conditional Frequency Distributions
  - `nltk.ConditionalFreqDist(`*`tuples`*`)`
  - Pairs of words: list of `tuples (`*`word1, word2`*`)`
- Bigrams
  - *n*-gram: n = 2, **bigram** is a sequence *word1 word2* in text
  - n = 3, **trigram** is *word1 word2 word3* in text
- Application: Random Text Generation

Bigrams

NLTK Book 2.4 Generating Random Text with Bigrams

```
>>> import nltk
>>> from nltk.corpus import genesis
>>> genesis
<PlaintextCorpusReader in '.../corpora/genesis' (not loaded yet)>
>>> g = genesis.words('english-kjv.txt')
>>> len(g)
44764
>>> g[:30]
['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven',
'and', 'the', 'earth', ',', 'And', 'the', 'earth', 'was',
'without', 'form', ',', 'and', 'void', ';', 'and', 'darkness',
'was', 'upon', 'the', 'face', 'of', 'the', 'deep']
```

Bigrams

- **Unigrams** (*simple list of words*):
  - `['In', 'the', 'beginning', 'God', 'created', 'the', 'heaven', 'and', 'the', 'earth', '.', 'And', 'the', 'earth', 'was', 'without', 'form', ',', 'and', 'void', ';', 'and', 'darkness', 'was', 'upon', 'the', 'face', 'of', 'the', 'deep']`
- Bigrams (*list of tuples* (*word1, word2*)):
  - `[('In', 'the'), ('the', 'beginning'), ('beginning', 'God'), ('God', 'created'), ('created', 'the'), …]`

nltk.bigrams()

```
>>> bigrams = nltk.bigrams(g)
>>> bigrams
<generator object bigrams at 0x11d540200>
>>> b = list(bigrams)
>>> len(b)
44763
>>> b[:30]
[('In', 'the'), ('the', 'beginning'), ('beginning', 'God'), ('God',
'created'), ('created', 'the'), ('the', 'heaven'), ('heaven', 'and'),
('and', 'the'), ('the', 'earth'), ('earth', '.'), ('.', 'And'), ('And',
'the'), ('the', 'earth'), ('earth', 'was'), ('was', 'without'),
('without', 'form'), ('form', ','), (',', 'and'), ('and', 'void'),
('void', ';'), (';', 'and'), ('and', 'darkness'), ('darkness', 'was'),
('was', 'upon'), ('upon', 'the'), ('the', 'face'), ('face', 'of'), ('of',
'the'), ('the', 'deep'), ('deep', '.')]
```

nltk.ConditionalFreqDist()

Summary:

- `nltk.ConditionalFreqDist()` takes as input a bigram corpus (actually a list of (word1,word2) tuples) produced by `nltk.bigrams(`*`text`*`)`

Usage:

- `cfd = nltk.ConditionalFreqDist(nltk.bigrams(`*`text`*`))`
- `cfd[`*`word1`*`]` returns a `nltk.FreqDist` for *word2*.

Example:

```
>>> cfd = nltk.ConditionalFreqDist(b)
>>> cfd['In']
FreqDist({'the': 9, 'my': 2, 'thee': 1})
```

- `cfd[word]` gives a count dict of which words can follow `word`

nltk.ConditionalFreqDist()

Case sensitive: *the* vs. *The*

```
>>> cfd['The']
FreqDist({'sons': 5, 'children': 4, 'LORD': 4, 'man': 3, 'days': 2,
'three': 2, 'name': 1, 'woman': 1, 'serpent': 1, 'earth': 1, ...})
>>> cfd['the']
FreqDist({'land': 156, 'LORD': 154, 'earth': 105, 'sons': 69, 'name':
42, 'field': 39, 'men': 36, 'man': 34, 'waters': 30, 'children': 30,
...})
```

`cfd[`*word*`].N()` tells us the size (total # words in the corpus) that can follow *word*:

```
>>> cfd['the'].N()
2411
>>> cfd['The'].N()
50
```

nltk.ConditionalFreqDist()

- `cfd[word1].max()` tells us what word is most likely to follow *word1*.

```
>>> cfd['In'].max()
'the'
```

- `cfd[word1].N()` tells us the size (total # words in the corpus) that can follow *word1*.

```
>>> cfd['In'].N()
12
```

- `cfd[word1].keys()` gives the keys to the count dict; i.e. list of words that can follow *word1*.

```
>>> cfd['In'].keys()
dict_keys(['the', 'my', 'thee'])
```

- `cfd[word1][word2]` gives the number of times `word2` can follow `word1`.

nltk.ConditionalFreqDist()

- `cfd[word1][word2]` gives the number of times `word2` can follow `word1`.

```
>>> cfd['The']['name']
1
>>> cfd['The']['sons']
5
```

nltk.ConditionalFreqDist()

Table 2.1:

NLTK's Conditional Frequency Distributions: commonly-used methods and idioms for defining, accessing, and visualizing a conditional frequency distribution of counters.

Example	Description
`cfdist = ConditionalFreqDist(pairs)`	create a conditional frequency distribution from a list of pairs
`cfdist.conditions()`	the conditions
`cfdist[condition]`	the frequency distribution for this condition
`cfdist[condition][sample]`	frequency for the given sample for this condition
`cfdist.tabulate()`	tabulate the conditional frequency distribution
`cfdist.tabulate(samples, conditions)`	tabulation limited to the specified samples and conditions
`cfdist.plot()`	graphical plot of the conditional frequency distribution
`cfdist.plot(samples, conditions)`	graphical plot limited to the specified samples and conditions
`cfdist1 < cfdist2`	test if samples in `cfdist1` occur less frequently than in `cfdist2`

NLTK Book: Chapter 2

- File: nwords.py

```
def next_word(w):
    return choice([k for k in cfd[w].keys() for i in range(cfd[w][k])])
def nwords(n, w):
    for i in range(n):
        print(w, end=' ')
        w = next_word(w)
    print()
```

```
python -i nwords.py
>>> nwords(5, 'The')
The field , by two
>>> nwords(5, 'The')
The sons of many colours
>>> nwords(5, 'The')
The sons ' wives ,
>>> nwords(5, 'The')
The sons of Haran ;
```

NLTK Book: Chapter 2

```
def next_word(w):
    return choice([k for k in cfd[w].keys() for i in range(cfd[w][k])])
```

Function explained:

```
>>> cfd['beginning']
FreqDist({'of': 2, ',': 1, '.': 1, 'God': 1})
>>> cfd['beginning'].keys()
dict_keys(['God', 'of', '.', ','])
>>> cfd['beginning']['God']
1
>>> cfd['beginning']['of']
2
>>> [k for k in cfd['beginning'].keys() for i in range(cfd['beginning'][k])]
['God', 'of', 'of', '.', ',']
```

NB. Python 3.6 has a function called `choices()`

random.choice(*seq*)

Return a random element from the non-empty sequence *seq*. If *seq* is empty, raises IndexError.

NLTK Book: Chapter 2

```
>>> from random import *
>>> choice(['a','a','b'])
'a'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'heads'
>>> choice(['heads','heads','tails','heads','tails'])
'tails'
>>> count = 0
>>> for i in range(1000):
...     if choice(['heads','heads','tails','heads','tails']) == 'heads':
...             count += 1
...
>>> count
580
>>>
```

- 1000 trials:
  - 580 heads
- Expected number of heads?
  - 600
  - Since 3 out of 5 in ['heads','heads','tails','heads','tails'] are heads

NLTK Book: Chapter 2

```
def nwords(n, w):
    for i in range(n):
        print(w, end=' ')
        w = next_word(w)
    print()
>>> nwords(5,'The')
The purchase of the LORD
```

i in range(5)	w
0	The
1	purchase
2	of
3	the
4	LORD

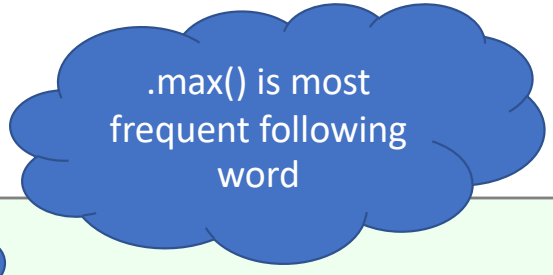
NLTK Book: Chapter 2

```
def nwords2(n, w):          # A more useful function returning a list of words
    result = []
    for i in range(n):
        result.append(w)
        w = next_word(w)
    return result
```

- Examples:

```
>>> nwords(8, 'The')
The children of Abram ' s cup into
>>> nwords2(8, 'The')
['The', 'sons', 'with', 'me', ',', 'for', 'yet', 'alive']
>>> nwords2(8, 'The')
['The', 'man', 'child', 'grew', ',', 'and', 'a', 'dream']
```

NLTK Book: Chapter 2




```
def generate_model(cfdist, word, num=15):
    for i in range(num):
        print(word, end=' ')
        word = cfdist[word].max()

text = nltk.corpus.genesis.words('english-kjv.txt')
bigrams = nltk.bigrams(text)
cfd = nltk.ConditionalFreqDist(bigrams)  ❶
```

```
>>> cfd['living']
FreqDist({'creature': 7, 'thing': 4, 'substance': 2, ',': 1, '.': 1, 'soul': 1})
>>> generate_model(cfd, 'living')
living creature that he said , and the land of the land of the land
```

Let's play with *Oliver Twist* by Charles Dickens

- File: oliver_twist0-53.txt

```
>>> raw = open('oliver_twist0-53.txt', encoding='utf-8').read()
>>> words = nltk.word_tokenize(raw)
>>> len(words)
197947
>>> cfd = nltk.ConditionalFreqDist(nltk.bigrams(words))
>>> cfd
<ConditionalFreqDist with 12379 conditions>
```

Let's play with *Oliver Twist* by Charles Dickens

- File: oliver_twist0-53.txt

```
>>> cfd['The']
FreqDist({'old': 35, 'Jew': 35, 'man': 25, 'girl': 25, 'boy': 18,
'two': 12, 'dog': 11, 'gentleman': 9, 'child': 8, 'young': 8, ...})
>>> cfd['the']
FreqDist({'Jew': 268, 'old': 201, 'girl': 173, 'door': 150, 'same':
136, 'boy': 122, 'other': 106, 'doctor': 103, 'young': 102, 'man':
100, ...})
nwords(20, 'The')
```