CIS 122

Random Text Generation
Logistics

● General trouble on Assignment 4
  ○ Few homework submissions

● Few signs of trouble beforehand
  ○ Few people in office hours
  ○ Few emails

● Assignment 4 deadline extension
  ○ Submit by tonight

● Assignment 5 posted soon
Random Paper Generator


- Generates random academic computer science papers
  - Randomly generated graphs
  - Randomly generated tables
  - Randomly generated citations

- 2005 paper accepted to conference
This week's project

- Write a text generator
- Using same general methods as SCIlgen
  - A little less coherent
  - But still cool
- We have many of the tools we need already
  - We'll pick up more as the week progresses
Markov Text Generation

- How do we generate random text?
  - Start by generating a single sentence
- Find a word that could start a sentence
  - Put it at the beginning
- Find words which could come after that word
  - Pick one to continue the sentence
- Repeat until you've formed a sentence
  - Now do it again!
Markov Text Generation

- How do we know which words come after other words?
  - Need a reference corpus

  fuzzy wuzzy was a bear.
  fuzzy wuzzy had no hair.
  fuzzy wuzzy wasn't very fuzzy was he.
Markov Text Generation

- For each word in corpus, see what words come afterwards

  fuzzy wuzzy was a bear.
  fuzzy wuzzy had no hair.
  fuzzy wuzzy wasn't very fuzzy was he.
Markov Text Generation

- For each word in corpus, see what words come afterwards

  - fuzzy wuzzy was a bear.
  - fuzzy wuzzy had no hair.
  - fuzzy wuzzy wasn't very fuzzy was he.
Markov Text Generation

- For each word in corpus, see what words come afterwards

  fuzzy wuzzy was a bear.
  fuzzy wuzzy had no hair.
  fuzzy wuzzy wasn't very fuzzy was he.

fuzzy → [ wuzzy, wuzzy, wuzzy, was ]
Markov Text Generation

- For each word in corpus, see what words come afterwards

fuzzy → [ wuzzy, wuzzy, wuzzy, was ]
wuzzy → [ was, had, wasn't ]
was → [ a, he ]
a → [ bear ]
bear → [ . ]
had → [ no ]
no → [ hair ]
hair → [ . ]
wasn't → [ very ]
very → [ fuzzy ]
he → [ . ]
Markov Text Generation

- Given a word, we can look up which words come next
  - And pick one of them randomly

- How do we know where to start/stop?

- Treat the '.' character as a special kind of word
  - Any word following a '.' can start a sentence
  - Reaching a period ends a sentence
Markov Text Generation

- This is a large problem
  - Where do we start?

- Break it down into pieces
  - What components do we need?
  - What do we need to be able to do?
One possible problem breakdown

- Read in corpus text from file as string
- Break string into list of words
- Process word list to separate out periods
- Produce markov dictionary from processed word list
- Produce single sentence from markov dictionary
- Generate text by producing as many sentences as desired
Back to Lists

- We saw lists briefly last week
  - Let's take a closer look

- Lists are sequences of values
  - [1, 2, 3]
  - ['apple', 'banana', 'carrot']
  - [True, 'B', 3]
Back to Lists

- Lists are mutable
  - We can modify them

```python
>>> L = [1, 2, 3]
>>> L[0] = 99
>>> L
[99, 2, 3]
```

- What happens if we try this with a string?
• Lists are mutable
  ○ We can grow them

  >>> L = [1, 2, 3]
  >>> L.append(4)
  >>> L
  [1, 2, 3, 4]

• The `append` method doesn't return anything
  ○ But it changes the list
Back to Lists

● Modifying a list is not the same as performing reassignment

● The variable still points to the same object
  ○ But that object has changed!

```python
>>> original = [ 1, 2, 3 ]
>>> copy = original
>>> copy.append(4)
>>> original
[ 1, 2, 3, 4 ]
```
Back to Lists

- Appending is a great tool for constructing lists
  - Start with an empty list
  - Repeatedly append elements

- The accumulator pattern for lists!
Let's get some list-building practice!

Write a method zeros(n)
  ○ Returns a list containing n zeros

>>> zeros(5)
[0, 0, 0, 0, 0]

>>> zeros(0)
[]
List Practice

- Let's get some list-building practice!

- Write a method zeros(n)
  - Returns a list containing n zeros

```python
def zeros(n):
    """Returns a list containing n 0's"""
    zeroList = [ ]
    for x in range(n):
        zeroList.append(0)
    return zeroList
```