Think Python 2e, Chapter 13 Notes

Markov process

October 12, 2022

Random numbers

$$0 \leq x < 1.0$$

```
for i in range(10):
    print(random.random())


0.1696556850562424
0.19127742543142245
0.1662683832804004
0.8333450811174378
0.820128992713088
0.26810519386624454
0.1359756795625645
0.174948156405357
0.9324127088396631
0.3904256608447578
```

Random integers

$$lower \leq n \leq upper$$

```
1  for i in range(10):
2    print(random.randint(20,25))
3
4
5  20
6  20
7  25
8  22
9  21
10 25
11 25
12 22
13 25
14 22
```

Random choice

```
1
2  for i in range(10):
3      print(random.choice([11, 33, 100, 44]))
4
5
6  100
7  100
8  44
9  11
10 33
11 44
12 33
13 100
14 100
15 100
```

Choose from histogram with probability

Write a function named `choose_from_hist` that takes a histogram as defined in Section 11.2 and returns a random value from the histogram, chosen with probability in proportion to frequency. For example, for this histogram:

```
>>> t = ['a', 'a', 'b']
>>> hist = histogram(t)
>>> hist
{'a': 2, 'b': 1}
```

your function should return 'a' with probability 2/3 and 'b' with probability 1/3.

Histogram of words in a text

Note different definition of word.

```
def process_file(filename):
    hist = dict()
    fp = open(filename)
    for line in fp:
        process_line(line, hist)
    return hist

def process_line(line, hist):
    line = line.replace('-', ' ')
    strips = string.punctuation + string.whitespace
    for word in line.split():
        word = word.strip(strips)
        word = word.lower()
        hist[word] = hist.get(word, 0) + 1

hist = process_file('emma.txt')
```

Histogram of words in a text

- Giben a histogram, how can you find the total number of words in a text?

Histogram of words in a text

- Giben a histogram, how can you find the total number of words in a text?

```
1 def total_words(hist):
2     return sum(hist.values())
```

- How can you find the total number of different words in a text?

Histogram of words in a text

- Giben a histogram, how can you find the total number of words in a text?

```
1 def total_words(hist):
2     return sum(hist.values())
```

- How can you find the total number of different words in a text?

```
1 def different_words(hist):
2     return len(hist)
```

Most common words

```
1 def most_common(hist):
2     t = []
3     for key, value in hist.items():
4         t.append((value, key))
5 
6     t.sort(reverse=True)
7     return t
```

Most common words

```
def most_common(hist):
    t = []
    for key, value in hist.items():
        t.append((value, key))

    t.sort(reverse=True)
    return t
```

Alternatively:

```
def oneth(item):
    return item[1]
def most_common(hist):
    t = hist.items()
    t.sort(reverse=True, key=oneth)
    return t
```

We used some **optional parameters**.

lambda

- Python's `lambda` feature is not described in the book. There are many online tutorials, like this one: `www.w3schools.com/python/python_lambda.asp`
- `lambda` creates a small anonymous function:

```
>>> x = lambda a : a + 10
>>> print(x(5))
15
>>> f = lambda a, b : 2*(a + b)
>>> print(f(3,4))
14
```

- Note there is no `return` keyword

Use of `lambda`

- `lambda` is very convenient when you need a short function for a short time.
- Instead of:

```
def oneth(item):
    return item[1]
def most_common(hist):
    t = hist.items()
    t.sort(reverse=True, key=oneth)
    return t
```

- We can write:

```
def most_common(hist):
    t = hist.items()
    t.sort(reverse=True, key=lambda x : x[1])
    return t
```

Optional parameters in user written functions

```
def print_most_common(hist, num=10):
    t = most_common(hist)
    print('The most common words are:')
    for freq, word in t[:num]:
        print(word, freq, sep='\t')
```

10 is a **default value**.

```
print_most_common(hist)
print_most_common(hist, 20)
```

The value 20 **overrides** the default value.

Dictionary subtraction

- Find all the words in a dictionary that are not in another dictionary.

Dictionary subtraction

- Find all the words in a dictionary that are not in another dictionary.

```
1  def subtract(d1, d2):
2      res = dict()
3      for key in d1:
4          if key not in d2:
5              res[key] = None
6      return res
```

Dictionary subtraction

- Find all the words in a dictionary that are not in another dictionary.

```
1  def subtract(d1, d2):
2      res = dict()
3      for key in d1:
4          if key not in d2:
5              res[key] = None
6      return res
```

- Why might we use dictionaries instead of lists?

Dictionary subtraction

- Find all the words in a dictionary that are not in another dictionary.

```
1  def subtract(d1, d2):
2      res = dict()
3      for key in d1:
4          if key not in d2:
5              res[key] = None
6      return res
```

- Why might we use dictionaries instead of lists?
- Look at the documentation for the `set` data structure in Python.

Random words

```
1 def random_word(h):
2     t = []
3     for word, freq in h.items():
4         t.extend([word] * freq)
5
6     return random.choice(t)
```

Inefficient. Why?

Random words

An alternative:

1. Use keys to get a list of the words in the book.
2. Build a list that contains the cumulative sum of the word frequencies. The last item in this list is the total number of words in the book, n.
3. Choose a random number from 1 to n. Use a bisection search to find the index where the random number would be inserted in the cumulative sum.
4. Use the index to find the corresponding word in the word list.

Markov processing of texts

File: emma.txt
Order: 2
Bates's door to let me lose it. I assure you, excepting those views on the subject." "You mistake me, you know–in summer there is no necessity for my intimate friend! Not regret her having no female connexions of his feelings. The touch seemed immediate. "Ah! Miss Woodhouse, and defying Miss Smith. How do you do, Mr. Richard?–I saw you first came. Go and eat and drink a little reserve of Enscombe. Captain Weston, and run away from the Crown chaise, and the best to say, with a most unfortunate–most deplorable mistake!–What is to hear about the lawn the next news, think.

Markov processing of texts

File: emma.txt
Order: 3
I saw symptoms of attachment between them–certain expressive looks, which I did the very last thing before I am aware. I am a very slow walker, and my pace would be tedious to you; and, besides, you have another long walk before you, to Donwell Abbey." "Thank you, sir, thank you; I am going to Kingston. He has passed you very often." "That may be, and I may have underrated him. My acquaintance with him has been but trifling.–And even if I have not the smallest doubt that, could he see his little effusion honoured as _I_ see it, (looking the book

Markov processing of texts

File: emma.txt
Order: 4
Knightley came–I think the very evening.–Do not you remember his cutting his finger with your new penknife, and your recommending court-plaister?–But, as you had none about you, and knew I had, you desired me to supply him; and so I took mine out and cut him a piece; but it was a hard case to be obliged still to lower herself in his opinion. She went on, however. “I have very little to say for my own conduct.–I was tempted by his attentions, and allowed myself to appear pleased.–An old story, probably–a common case–and no more than has happened to of my sex

Markov processing of texts

File: greeneggsandham.txt
Order: 2
THERE. SAY! I WILL NOT EAT GREEN EGGS AND HAM? I DO SO LIKE GREEN EGGS AND HAM? I DO NOT LIKE GREEN EGGS AND HAM" (by Doctor Seuss) I AM SAM. I AM SAM. I AM SAM. SAM I AM. THAT SAM-I-AM! DO WOULD YOU COULD YOU IN THE RAIN. AND IN A TREE. NOT IN A BOX. NOT WITH A FOX? NOT IN THE DARK! NOT IN A CAR! SAM! LET ME BE! I WOULD NOT, COULD NOT WITH A GOAT. I WILL EAT THEM WITH A MOUSE. AND I WILL EAT THEM IN THE RAIN? I WOULD EAT

Markov processing of texts

File: mobydick.txt
Order: 2
Seeing how matters were, dived down and lost in the waves; there, that blood-dripping head hung to the surprise of all, one grand feature. For example,—after a weary and perilous chase and fang that flying-fish? Where do murderers go, man! Who's over him, as over a mouthful of Grenadier's steak. And thus with the silken pearl-colored membrane, like the cleft drooping boughs of a strange sight that, Parsee:—a hearse and its upper end of the whale and a second to another, the sperm whale had sounded; but intending to be elsewhere. While yet some way be stripped of that strange observable

Markov processing of texts

File: tarzanoftheapes.txt
Order: 2
Behind trailed the women, yelling and beating upon her face with a scream of terror for some means to cope with circumstances whatever they may even be a land-locked harbor. To Tarzan this was Tarzan's only reply. And again he drew out the vision of a great figure towering above her, but might as well that the lion had scarce finished his repast he dipped his finger-ends into a state of the higher branches, for if she could be naught of tanning, he was armed with a very different direction from which the cries of his mouth ere he, too, caught

Markov processing of texts

File: theadventuresofsherlockholmes.txt
Order: 2
I lounged up the steps; but she was the name of the _American Encyclopædia_ which stands near the Museum—we are to hush the matter between this and then. Good-bye; it is a hard fight against a smoke-laden and uncongenial atmosphere. Three gilt balls and a short thick man with a frowning brow and a baboon." "Ah, yes, of course obvious upon the previous night. Were it not only hear the deep mystery through which he was borne into Briony Lodge was open, and that one I was arrested as his client paused and refreshed his memory with a twinkle in rooms.

Markov processing of texts

File: thesunalsorises.txt
Order: 2
Our friend and his net all in the war. There was a good body, and the streets early in the reading-room and took a little money. I spent all my time reading the paper. “See Mike?” “Yes.” “Let’s go out of the trout-bags in. I carried the other. “Well,” said Bill. “Quite. They’re not really living it.” “Nobody ever lives their life all the walls, and I pointed out to Bill. Bill handed him the local Syndicat d’Initiative office, where the count reached down and eat in the street and eat. See you later, Jake.” He walked with a number motor-cars

Markov processing of texts

File: thefederalistpapers.txt
Order: 2
In its council of Pennsylvania, the representatives of the government into distinct and independent governments, as to give their votes upon merchants and navigators, and which promise a sincere zeal for the loss of a government commensurate to its aid the collective bodies of electors, will be thoroughly masters of the adversaries to liberty that the propriety of his own reputation, and, in a clear indication, as far as respects individuals, must, like all other powers declared in its various branches, must form a correct judgment of the United States might, at this time, or can be regulated by uniform without

Design decisions for Markov text

- How to represent prefixes.
- How to represent the collection of suffixes.
- How to represent the mapping from each prefix to the collection of suffixes.

Benchmarking

- Implement more than one choice of data structure, and test to see which is faster in actual practice.
- An alternative is to make your best guess as to datastructure and see if it is *good enough.*

Space *vs.* Time

- For choosing a random word with probability according to frequency, we can use a histogram or a list with duplicates.
- A histogram will take less space than a list with duplicates, if there are a lot of duplicates.

Different data structures for different purposes

- Might use one data structure for finding prefixes and suffixes.
- And a different data structure for generating text.

Debugging

- Reading
- Running
- Ruminating
- Refactoring
- Rubberducking
- Retreating

Vocabulary

deterministic: Pertaining to a program that does the same thing each time it runs, given the same inputs.

pseudorandom: Pertaining to a sequence of numbers that appears to be random, but is generated by a deterministic program.

default value: The value given to an optional parameter if no argument is provided.

override: To replace a default value with an argument.

Vocabulary

benchmarking: The process of choosing between data structures by implementing alternatives and testing them on a sample of the possible inputs.

rubber duck debugging: Debugging by explaining your problem to an inanimate object such as a rubber duck. Articulating the problem can help you solve it, even if the rubber duck doesn't know Python.