lesson1_rulebased

April 27, 2018

```
In [1]: # Run this every time you open the spreadsheet
        % load_ext autoreload
        % autoreload 2
        from collections import Counter
        import lib
```

1 Load and inspect the data

```
In [2]: # Load the data.
        tweets, test_tweets = lib.read_data()
```

```
In [3]: # The variable "tweets" is a list of tweets.
        # Mini-exercise 1: Print the number of tweets in the list

        print("Number of tweets:", len(tweets))

        # Mini-exercise 2: Assign the 10th tweet to the variable "tweet" and print it

        tweet = tweets[9]
        print(tweet)

        # To view the category of a tweet, we access the attribute tweet.category
        # Mini-exercise 3: Print the category of this tweet.

        category = tweet.category
        print("Category of the tweet:", category)
```

```
Number of tweets: 1120
food prep yoga classes supply runs organization
Category of the tweet: Food
```

```
In [4]: # This function prints out a table containing all the tweets, along with their categor
        lib.show_tweets(tweets)
```

```
<IPython.core.display.HTML object>
```

2 Python refresher

First, let's do some exercises to refresh our memory of a few Python concepts.

2.0.1 Functions

A Python function is written like this:

```
def add_one(x):
    return x+1
```

The name of the function is `add_one`, `x` is the input variable, and the `return` keyword tells us what to give as output.

```
In [5]: # Exercise 1. Define a function called "square_minus_1" that takes one variable (x),
        # squares it, subtracts 1, and returns the result.

        #### YOUR CODE STARTS HERE ####

        def square_minus_1(x):
            return x**2 - 1;

        #### YOUR CODE ENDS HERE ####

        print("Testing:")
        for x in [3,-4,6.5,0]:
            print(str(x) + " -> " + str(square_minus_1(x)))
            print("CORRECT" if square_minus_1(x)==(x**2-1) else "INCORRECT")
```

```
Testing:
3 -> 8
CORRECT
-4 -> 15
CORRECT
6.5 -> 41.25
CORRECT
0 -> -1
CORRECT
```

2.0.2 If-else statements

An if/else statement looks like this:

```
if electoral_votes >= 270:
    print "You win the election"
else:
    print "You lose the election"
```

The if-statement is evaluated (`electoral_votes >= 270`); if it's true then the code under the `if` is executed, if it's false then the code under the `else` is executed.

```
In [6]: # Exercise 2. Define a function called "contains_ss" that takes one variable (word)
        # and returns True if the word contains a double-s and False if it doesn't.
        # Hint: to test whether a string e.g. "ss" is inside another string variable e.g. word
        #    if "ss" in word:

        #### YOUR CODE STARTS HERE ####

        def contains_ss(word):
            return "ss" in word

        #### YOUR CODE ENDS HERE ####

        print("Testing:")
        for word in ["computer", "science", "lesson"]:
            print("%s ->" % word, contains_ss(word))
            print("CORRECT" if contains_ss(word) == ("ss" in word) else "INCORRECT")
```

```
Testing:
computer -> False
CORRECT
science -> False
CORRECT
lesson -> True
CORRECT
```

2.1 More complex if-else statements

Maybe you want to check *several* conditions? You can use an if/elif/else statement.

```
if teamA_score > teamB_score:
    print "Team A wins"
elif teamA_score < teamB_score:
    print "Team B wins"
else:
    print "It's a tie!"
```

`elif` stands for "else if". In fact, the above code is just a neater way of writing this:

```
if teamA_score > teamB_score:
    print "Team A wins"
else:
    if teamA_score < teamB_score:
        print "Team B wins"
    else:
        print "It's a tie!"
```

You can have as many `elif` statments as you like. These are useful for when you want several options.

```
In [7]: # Exercise 3. Define a function called "grade" that takes one input (score).
        # If score >= 90, return the string "A"
        # Otherwise, if score >= 80, return the string "B"
        # Otherwise, if score >= 70, return the string "C"
        # Otherwise, if score >= 60, return the string "D"
        # Otherwise, if score >= 50, return the string "E"
        # Otherwise, return the string "F"

        #### YOUR CODE STARTS HERE ####

        def grade(score):
            if score >= 90:
                return "A"
            elif score >= 80:
                return "B"
            elif score >= 70:
                return "C"
            elif score >= 60:
                return "D"
            elif score >= 50:
                return "E"
            else:
                return "F"

        #### YOUR CODE ENDS HERE ####


        print("Testing:")
        for (score, g) in [(77, "C"), (80, "B"), (32, "F"), (100, "A"), (69, "D")]:
            print("%i -> %s" % (score, grade(score)))
            print("CORRECT" if grade(score) == g else "INCORRECT")
```

```
Testing:
77 -> C
CORRECT
80 -> B
CORRECT
32 -> F
CORRECT
100 -> A
CORRECT
69 -> D
CORRECT
```

3 Write a rule-based tweet classifier

Time to write our rule-based classifier! The function outline below uses a `if/elif/else` statement to return the predicted category of a tweet.

Fill in the missing `if` and `elif` statements with something sensible (there is no one right answer)!

Start with something simple; we'll build it into something more complicated later.

```
In [8]: def classify_rb(tweet):
            tweet = str(tweet).lower()  # this makes the tweet lower-case, so we don't have to

            if "medicine" in tweet:
                return "Medical"
            elif "power" in tweet:
                return "Energy"
            elif "water" in tweet:
                return "Water"
            elif "milk" in tweet:
                return "Food"
            else:
                return "None"
```

4 Test your rule-based classifier on some examples

Run the cell below to see the results of your rule-based classifier. You should see a table showing each tweet, along with its true category and the category predicted by your system.

Which types of tweets does your system get right? Which types of tweets does your system get wrong and why?

How would you measure the accuracy of your system?

```
In [9]: predictions = [(tweet, classify_rb(tweet)) for tweet in test_tweets]  # a list of (twe

        lib.show_predictions(predictions, True)

<IPython.core.display.HTML object>
```

5 Break your rule-based classifier!

It's time to FOOL THE RULES!

You'll be deliberately trying to break each others' rule-based classifiers by writing tricky tweets that fool your neighbor's rule-based classifier. Once your own classifier has been fooled by a tricky tweet, it's your job to amend the rules in your classifier to account for the new case.

```
In [10]: def classify_rb_game(tweet):
             tweet = str(tweet).lower()  # this makes the tweet lower-case, so we don't have t

             if "medicine" in tweet:
                 return "Medical"
             elif "power" in tweet:
                 return "Energy"
             elif "water" in tweet:
                 return "Water"
             elif "milk" in tweet:
                 return "Food"
             else:
                 return "None"
             pass
```

5.0.1 Write a tweet about Food that will be misclassified

Below, write a disaster-scenario tweet about Food that the classification function above will get wrong (i.e. fail to recognize it's about food).

Hint: think of less-obvious food-related keywords that aren't included in the rule-based system above.

Then run the cell - make sure the tweet is classified as something other than Food!

```
In [11]: food_tweet = "I love watermelon."
         print("This tweet is classified as: %s\n" % classify_rb_game(food_tweet))
```

```
This tweet is classified as: Water
```

5.0.2 Write a tweet about Energy that will be misclassified

```
In [12]: energy_tweet = "What's the medicine to energy crisis?"
         print("This tweet is classified as: %s\n" % classify_rb_game(energy_tweet))
```

```
This tweet is classified as: Medical
```

5.0.3 Write a tweet about Water that will be misclassified

```
In [13]: water_tweet = "What powers our life is water."
         print("This tweet is classified as: %s\n" % classify_rb_game(water_tweet))
```

```
This tweet is classified as: Energy
```

5.0.4 Write a tweet about Medical that will be misclassified

```
In [14]: medical_tweet = "Aspirin is not water, we should be careful."
         print("This tweet is classified as: %s\n" % classify_rb_game(medical_tweet))
```

```
This tweet is classified as: Water
```

5.0.5 Write a tweet NOT about Food, that will be falsely classified as Food

Below, write a disaster-scenario tweet that is NOT about Food, but that the classifier above will classify as Food.

Hint: you want to trick the classifier into thinking you're talking about food when you're not. Look at the keywords the rule-based system associates with food. Can you find a way to use them while actually talking about not-food?

- For example, if the system looks for the word "food" you could write *"Waiting out #Sandy by reading Plato. Food for thought."*
- If the system looks for the word "cook", you could write *"I hear the power's out in Cook County."*
- More simply, you could mention food incidentally but the real subject of the tweet is something else e.g. *"Was out food shopping when I heard about the power outage on the news. Hope everyone's OK."*

Then run the cell - make sure the tweet is classified as Food!

```
In [15]: not_food_tweet = "Milky way is beautiful"
         print("This tweet is classified as: %s\n" % classify_rb_game(not_food_tweet))
```

```
This tweet is classified as: Food
```

5.0.6 Write a tweet NOT about Energy, that will be falsely classified as Energy

```
In [16]: not_energy_tweet = "I am out of power"
         print("This tweet is classified as: %s\n" % classify_rb_game(not_energy_tweet))
```

```
This tweet is classified as: Energy
```

5.0.7 Write a tweet NOT about Water, that will be falsely classified as Water

```
In [17]: not_water_tweet = "I love watermelon"
         print("This tweet is classified as: %s\n" % classify_rb_game(not_water_tweet))
```

```
This tweet is classified as: Water
```

5.0.8 Write a tweet NOT about Medical, that will be falsely classified as Medical

```
In [18]: not_medical_tweet = "skdfjmedicinesjwjer"
         print("This tweet is classified as: %s\n" % classify_rb_game(not_medical_tweet))
```

```
This tweet is classified as: Medical
```