lesson2_evaluation

April 29, 2018

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

1 Load the data and our rule-based classifier

```
In [2]: # Load the data.
        # This function returns "tweets" and "test_tweets", both lists of tweets
        import nltk
        nltk.download('punkt')
        tweets, test_tweets = lib.read_data()
```

```
[nltk_data] Downloading package punkt to /Users/xiaoxing/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
In [3]: def classify_rb(tweet):
            tweet = str(tweet).lower()  # this makes the tweet lower-case, so we don't have to
        worry about matching case

            if "medicine" in tweet or "first aid" in tweet:
                return "Medical"
            elif "power" in tweet or "battery" in tweet:
                return "Energy"
            elif "water" in tweet or "bottled" in tweet:
                return "Water"
            elif "food" in tweet or "perishable" in tweet or "canned" in tweet:
                return "Food"
            else:
                return "None"
```

2 Python refresher

Let's review some Python concepts before we write our evaluation code.

2.0.1 Lists

In Python, a *list* is an ordered collection of items. The items can be strings, numbers, booleans, or any other kind of Python object.

You can create lists like this:

```
integer_list = [5, 6, 7, 8]
string_list = ['hello', 'world']
bool_list = [False, True, False, False, True]
```

If you want a list of the numbers up to (but not including) 10, you can use the `range` function.

```
upto10_list = range(10)
```

This gives you [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

```
In [4]: # Exercise 1(a).
        # Create a list called "my_numbers" that contains the numbers from 0 to 6 (inclusive),
        and then print it
        my_numbers = [0, 1, 2, 3, 4, 5, 6]
        print(my_numbers)
```

```
[0, 1, 2, 3, 4, 5, 6]
```

```
In [15]: # Exercise 1(b).
         # Now use the range() function to create "my_numbers", and print the result.
         # It should match the previous cell.
         # Hint: look carefully at the range(10) example above.
         my_numbers = list(range(7))
         print(my_numbers)
```

```
[0, 1, 2, 3, 4, 5, 6]
```

2.0.2 For loops

In Python, a *for loop* allows you to iterate over a list.

```
shopping_list = ['bread', 'bananas', 'milk']

for item in shopping_list:
    print item
```

For example, the code above prints out the following output:

```
bread
bananas
milk
```

```
In [6]: # Exercise 2.
        # Write a for-loop that iterates through my_numbers, and prints the square of each
        number
        # You should see the following numbers print out, one per line: 0, 1, 4, 9, 16, 25, 36
        for item in my_numbers:
            print(item**2)
```

```
0
1
4
9
16
25
36
```

```
In [7]: # Exercise 3.
        # Use a for-loop to calculate the sum of the squares of my_numbers.
        # Save the result in a variable called "sum_squares".
        # Hint: start by setting sum_squares to 0 before starting the for-loop.

        #### YOUR CODE STARTS HERE ####
        sum_squares = 0
        for item in my_numbers:
            sum_squares += item**2

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

        print("Testing: sum_squares = %i" % sum_squares)
        print("CORRECT" if sum_squares == 91 else "INCORRECT")
```

```
Testing: sum_squares = 91
CORRECT
```

2.0.3 Incrementing

If you have an integer variable e.g. `x=3` and you want to increase `x` by 1 (which is called *incrementing*), then you can write

```
x = x+1
```

or, in shorthand:

```
x += 1
```

This can be useful when you're using `x` to count something. For example:

```
ages = [7, 14, 23, 3, 10, 19]

num_adults = 0
for age in ages:
    if age >= 18:
        num_adults += 1

print num_adults
```

What should this code print out?

```
In [8]: # Exercise 4.
        # Count the number of Weasleys in the list of characters, and save the result to the
        variable "num_weasleys".
        # Use incrementation with the "x += 1" notation.

        characters = ['Harry Potter', 'Ron Weasley', 'Albus Dumbledore', 'Ginny Weasley', 'Percy
        Weasley', 'Hermione Granger',
                      'Fred Weasley', 'George Weasley']

        #### YOUR CODE STARTS HERE ####
        num_weasleys = 0
        for item in characters:
            if "Weasley" in item:
                num_weasleys += 1

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

        print("Testing: num_weasleys = %i" % num_weasleys)
        print("CORRECT" if num_weasleys == 5 else "INCORRECT")
```

```
Testing: num_weasleys = 5
CORRECT
```

2.0.4 Testing for equality and inequality

Sometimes you want to check if two values are equal, perhaps using an `if` statement. To check for equality you need to use a *double* equals sign `==`.

```
x = 5
y = 8
if x == y:
    print "x and y are equal"
```

To check for *inequality*, i.e. if two things aren't equal, use `!=`.

```
x = 5
y = 8
if x != y:
    print "x and y are NOT equal"
```

```
In [9]: # Exercise 5.
        # Use a for-loop, incrementation and equality testing to count the number of cats in my
        list of pets.
        # Assign the result to the variable "num_cats"

        my_pets = ['cat', 'lizard', 'cat', 'dog', 'cat', 'snake', 'dog', 'cat', 'dog', 'parrot']

        #### YOUR CODE STARTS HERE ####
        num_cats = 0
        for item in my_pets:
            if "cat" == item:
                num_cats += 1


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

        print("Testing: num_cats = %i" % num_cats)
        print("CORRECT" if num_cats == 4 else "INCORRECT")
```

```
Testing: num_cats = 4
CORRECT
```

```
In [10]: # Exercise 6.
         # Use a for-loop, incrementation and inequality testing to count the number of pets that
         are neither cats nor dogs.
         # Assign the result to the variable "num_unusual".

         #### YOUR CODE STARTS HERE ####
         num_unusual = 0
         for item in my_pets:
             if "cat" != item and "dog" != item:
                 num_unusual += 1

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

         print("Testing: num_unusual = %i" % num_unusual)
         print("CORRECT" if num_unusual == 3 else "INCORRECT")
```

```
Testing: num_unusual = 3
CORRECT
```

3 Measure the accuracy of your rule-based classifier

Complete the function below to calculate the Precision, Recall and F1 for a given category (e.g. Food)

```
In [11]: def evaluate(predictions, c):
             """This function calculate the precision, recall and F1 for a single category c
         (e.g. Food)
             Inputs:
                 predictions: a list of (tweet, predicted_category) pairs
                 c: a category
             Returns:
                 The F1 score.
             """

             # Initialize variables to count the number of true positives, false positives and
         false negatives
             true_positives = 0.0
             false_positives = 0.0
             false_negatives = 0.0

             # Iterate through the tweets, counting the number of true positives, false positives
         and false negatives
             for (tweet, predicted_category) in predictions:
                 true_category = tweet.category

                 # Hint: true positives for category c are tweets that have
                 # true category c and predicted category c
                 if c == predicted_category and tweet.category == c:
                     true_positives += 1

                 # Finish the statement: false negatives for category c are tweets that have
                 # true category not c and predicted category not c
                 elif c != predicted_category and tweet.category !=c:
                     false_negatives += 1

                 # Finish the statement: false positives for category c are tweets that have
                 # true category c and predicted category not c
                 elif c != predicted_category and tweet.category ==c:
                     false_positives += 1

             # Before we calculate Precision, Recall and F1 we need to check whether
         true_positives = 0. Why?
             if true_positives == 0:
                 precision = 0.0
                 recall = 0.0
                 f1 = 0.0
             else:
                 # Calculate Precision, Recall and F1
                 # Consult the formulae on the slides
                 precision = true_positives / (false_positives + true_positives)
                 recall = true_positives / (false_negatives + true_positives)
                 f1 = 2 * precision * recall / (precision + recall)

             # Print the category name, Precision, Recall and F1
             print(c)
             print("Precision: ", precision)
             print("Recall: ", recall)
             print("F1: ", f1)
             print("")

             # Return the F1 score
             return f1


         predictions = [(tweet, classify_rb(tweet)) for tweet in test_tweets]  # Make a list of
         (tweet, predicted_category) pairs
```

```
        # Get the F1 scores for each category
        food_f1 = evaluate(predictions, "Food")
        water_f1 = evaluate(predictions, "Water")
        energy_f1 = evaluate(predictions, "Energy")
        medical_f1 = evaluate(predictions, "Medical")
        none_f1 = evaluate(predictions, "None")
```

```
Food
Precision:  0.8217054263565892
Recall:  0.4608695652173913
F1:  0.5905292479108635

Water
Precision:  0.95
Recall:  0.07063197026022305
F1:  0.1314878892733564

Energy
Precision:  0.4
Recall:  0.06477732793522267
F1:  0.11149825783972125

Medical
Precision:  0.5384615384615384
Recall:  0.025454545454545455
F1:  0.048611111111111105

None
Precision:  0.5569620253164557
Recall:  0.21359223300970873
F1:  0.30877192982456136
```

Complete the cell below to calculate the average F1 score, which should be the average of the F1 scores for each category.

```
In [12]: average_f1 = (food_f1 + water_f1 + energy_f1 + medical_f1 + none_f1)/5
         print("Average F1: ", average_f1)
```

```
Average F1:  0.2381796871919227
```

3.1 Look at the confusion matrix

- *Rows* represent the *true category* of the tweet
- *Columns* represent the *predicted category* from your classifier
- So numbers on the diagonal represent correct classifications, and off-diagonal numbers represent misclassification

```
In [13]: lib.show_confusion_matrix(predictions)
```

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

3.2 Look at the predictions

```
In [14]: lib.show_predictions(predictions)
```

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