Homework 9: Reusing code

L555

Due Wednesday, November 14

1. (a) Using the random module (http://docs.python.org/2/library/random.html), write a function which randomly selects a card from a (virtual) deck of cards. Treat the cards as 1–13 (i.e., 13=Ace, 12=King, etc.), and think of the problem as selecting both a number and a suit (spades, hearts, diamonds, clubs).

As a parameter to this function, take a list of cards which have already been selected.

(b) In a separate program, import this function. Write a function within this new program which deals a 5-card hand of poker. Make sure there are no repeats!

I will deal myself a hand of poker and will award a candy bar to whoever has a correctly-written (and fair) program that gives me the best poker hand.

(c) Bonus: allow me to discard up to 3 cards and redraw those cards.

2. Extend the program from inclass exercise 8.1 and add a class for language classes, which is a subclass of the course class and has additional information about the language taught, the level of competence. All of these types of information should be accessible and should have methods to change the values. Make sure that if an object of this class is created that the requirements state that the classroom must be a multimedia room.

3. Machine learning applications normally require information presented as a list of feature values. Let us assume that we want to have a machine learning approach to POS tagging. For this, we need to present the machine learner with each word to be tagged, combined with all the info that may be useful for tagging. This may include the previous two words, the following two words, the POS tags assigned to the previous two words, and the ambiguity classes for the focus word and the following two words. For example, a complete vector for the first 3 words in cd6.pos would look like the following:

\[
\begin{array}{cccccccc}
  w-2 & w-1 & w & w+1 & w+2 & p-2 & p-1 & ac & ac+1 & ac+2 \\
\end{array}
\]

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>well</td>
<td>I</td>
<td>think</td>
<td>NULL</td>
<td>NULL</td>
<td>RB_UH</td>
<td>LS_PP</td>
<td>VB_VBP</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td>well</td>
<td>I</td>
<td>think</td>
<td>we</td>
<td>NULL</td>
<td>UH</td>
<td>LS_PP</td>
<td>VB_VBP</td>
<td>PP</td>
<td></td>
</tr>
<tr>
<td>well</td>
<td>I</td>
<td>think</td>
<td>we</td>
<td>need</td>
<td>UH</td>
<td>PP</td>
<td>VB_VBP</td>
<td>PP</td>
<td>NN_VB_VBP</td>
<td></td>
</tr>
</tbody>
</table>

An ambiguity class is a combination of all the POS tags that a word can have, and can thus be extracted from the file. I recommend programming the feature vectors as follows: 1) As a first step, consider as features only the word itself, the two preceding words, and their POS tags. 2) Then, extend it to cover the following two words. 3) In a last step, add the ambiguity classes.