Python Notes for Assignment 3

Again, also see previous assignments' Python notes.

tokenize()

List comprehension: This may have come up before, but it's worth reiterating as a very Python-y construction. Iterating through a list, this calculates an anonymous function on each item and returns a list with the function applied to each item. Here

word_tokenize: From nltk; breaks the sentence down into words or other meaningful units (tokens). Follow the assignment instructions to download the necessary module and data.

ModelInfo

count initializations: [{}, {}, {}, {}, {}] is a list of 5 empty dictionaries.

s_word_counts.get(token, 0): This sets the default value if token isn't found to 0, which is useful for effectively treating all unseen items as having counts of 0.

dict and list access: Notice that both use the same square bracket notation to retrieve items.

bigrams and generators: You might expect bigrams to return a list of bigrams, but actually, it returns an *iterator* that will return bigrams when you ask for them. This subtle difference means that you need to call bigrams() again if you reach the end of the sentence for one call, and want to start again for a different class.

classify_sentences():

Multiple return values: Notice how this is expecting two return values from each of the functions you're to fill in. Return the values with a comma between them, as in the placeholder code.

main:

if __name__ = "__main__": This line keeps the following lines from executing when the file is imported, as opposed to when run from the command line.