Grade: 40/40
Vocareum: 45/52
Cannot receive score > 40 so rounds down to 40/40

CSCI 544: Applied Natural Language Processing

Assignment 1: There once was a warmup from . . .

Jonathan May (adapted from Jordan Boyd-Graber)

Out: **25. August 2017**
Due: **8. September 2017**

Edit of 2017-08-26 17:22:33-07:00

Introduction

First, log into Vocareum to find the template code.

The goal of this assignment is to create a piece of code that will determine whether a poem is a limerick or not. To do this, we will be using the CMU pronunciation dictionary (which is covered in the second chapter of the NLTK book).

A limerick is defined as a poem with the form AABBA, where the A lines rhyme with each other, the B lines rhyme with each other, and the A lines do not rhyme with the B lines. (English professors may disagree with this definition, but that's what we're using here to keep it simple.) There are also constraints on how many syllables can be in a line; these are provided in the code templates.

Programming Section (40 points)

Look at the file `limerick.py` in the hw1 folder. Your job is to fill in the missing functions in that file so that it does what it is supposed to do.

- **num_syllables**: count the number of syllables in a word
- **rhymes**: detect whether two words rhyme or not
- **is_limerick**: given a candidate limerick, return whether it meets the constraint or not.

More requirements / information appear in the source files.

Note: The stub code and test code assume you are using python 2.7. If you wish to use python3+ you should **change the name of this file to** `limerick3.py`.

Notes:

- **What does it mean for two words to rhyme?**

  They should share the same sounds in their pronunciation except for their first consonant sound(s) and anything before their first consonant sounds. (This is a very strict definition of rhyming. This makes the assignment easier.) If one word is longer than the other, then the sounds of the shorter word (except for its first consonant sound and anything before the first consonant sound) should be a suffix of the sounds of the longer.

- **How do I know if my code is working?**

  Run the provided unit tests:

  `python tests.py`

  in the homework directory. *This is strongly encouraged, as it will be similar to how your code is graded.* Note that if you choose to use python3 you may have to change `tests.py`.

  Initially, many of the tests will fail. If your program is working correctly, all of them will pass. However, the converse is not true: passing all the tests is not sufficient to demonstrate that your code is working. Thus, you may wish to add tests or send in various limericks and non-limericks to the program.

- **How do I separate words from a string of text?**

  Use the `word_tokenize` function.

- **What if a word isnt in the pronouncing dictionary?**

  Assume it doesnt rhyme with anything and only has one syllable.

- **How "hardened" should the code be?**

  It should handle ASCII text with punctuation and whitespace in upper or lower case.

- **What if a word has multiple pronunciations?**

  If a word like fire has multiple pronunciations, then you should say that it rhymes with another word if any of the pronunciations rhymes.

- **What if a word contains exclusively vowel sounds?**

Then it has no initial consonant sound. To rhyme with another word its pronunciation should be a substring of the other word's pronunciation. E.g. "aye" (`AY1`) and "die" (`D AY1`) rhyme.

- **What about end rhymes?**

  End rhymes are indeed a rhyme, but they make for less interesting limericks. So "conspire" and "fire" do rhyme.

- **What about stress?**

  The stresses of rhymes should be consistent.

Extra Credit

Extra Credit (create new functions for these features; don't put them in the required functions that will be run by the autograder):

(up to 2 points) Create a new function called `apostrophe_tokenize` that handles apostrophes in words correctly so that "can't" would rhyme with "pant" if the `is_limerick` function used `apostrophe_tokenize` instead of `word_tokenize`.

(up to 5 points) Make reasonable guesses about the number of syllables in unknown words in a function called `guess_syllables`.

(up to 5 points) Compose a funny original limerick about computational linguistics, natural language processing, or machine learning (add it to your submission as `limerick.txt`). Selected limericks will be read aloud in class; if you don't want this to happen but you want to contribute, please note this when you submit.

Add extra credit code as functions to the `LimerickDetector` class, but don't interfere with required functionality.

Note: The above extra credit can be used to improve your score on *this homework only* and cannot result in more than full credit on the assignment.

Submission

Submit `limerick.py` and, optionally, `limerick.txt` before the deadline using Vocareum. You may submit multiple times before the deadline; only your final submission will count. As noted previously, if you are using python3, you should rename your python file as `limerick3.py`.