

Natural Language Processing:

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

Jordan Boyd-Graber

Out: **25. August 2014**

Due: **12. September 2014**

Introduction

First, check out the Github repository with the course homework templates:

[git://github.com/ezubarc/cl1-hw.git](https://github.com/ezubarc/cl1-hw.git)

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 not the A 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.)

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 its supposed to do.

- **rhyme**: detect whether two words rhyme or not
- **limerick**: given a candidate limerick, return whether it meets the constraint or not.

More requirements / information appear in the source files.

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). (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 shorter word (except for its first consonant) 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. Initially, many of them 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. *This is strongly encouraged, as it will be similar to how your code is graded.*

- **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 starts with a vowel?**

Then it has no initial consonant, and then the entire word should be a suffix of the other word.

- **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; dont 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 “cant” 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`).

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