Homework 3

Syllabification

Due 11:59 pm, 24 April, 2020

In this assignment, you will develop a syllabification algorithm, and run it on the CMU pronouncing dictionary, which you processed in HW 2.

You will also print your syllabified version of CMU to a file, and compare that output with a 'correct' output file, counting the number of errors.

Background: Syllabification

Syllables are composed of three ordered components: Onsets, Codas, and Nuclei. All syllables must have nuclei, which are usually vowels (though not absolutely always), but they vary in how many consonants make up an onset, or a coda. A large syllable like the word "sprint" has three onset consonants, and two coda consonants. Meanwhile, a tiny syllable like the word 'a' has zero onset and zero coda consonants.

The tricky thing about figuring out syllabification is dealing with consonants that appear in the middle of a word, in between vowels. Some of these are onsets, some codas. For example, the word 'party' would be split up as 'par.ty', with the 'r' as a coda and the 't' as an onset. ("." delimits syllable boundaries) A similar word, 'deprive' is split up differently 'de.prive'. Here, both the 'p' and the 'r' are in the onset of the second syllable, and the first syllable has no coda at all.

Even a single consonant between two vowels can sometimes be an onset and sometimes a coda: The [b] of 'hobby' is an onset, while the [ŋ] of 'ringer' is a coda. The arrangement of these word-medial[1] consonants is not random, though; it is governed by a very strict set of rules. These rules refer to the **sonority** of each sound in the word. Sonority is acoustially related to how long and loud a sound is, as well as how well it holds a pitch. Vowels are the most **sonorant** sounds, and stops and fricatives are the least. Syllables like to be set up so that the highest sonority thing (the vowel) is in the middle, sonority drops off to either side. So, rising sonority (pl, kr, sm...) is preferred in onsets, while falling sonority is preferred in codas (lp, rk, ms,...).

[1] Linguist-y way of saying 'in the middle of the word'

We can give sounds numbers to indicate their sonority:

4 – Vowels and syllabic consonants
3 – [j] and [w], as in 'yell' and 'wow'
2 – [l] and [ɹ]
1 – nasals [n, m, ŋ]
0 – stops, fricatives, and affricates ("obstruents")

Your syllabification algorithm should follow these principles:

- All vowels (sonority level 4) are nuclei
- All initial consonants are onsets, and all final consonants are codas (as in HW2)
- **Onset Maximization**: put as many consonants into the onset as possible, so long as the difference in sonority between consecutive consonants is at least 2. So, [pr] would be a good onset because it's sonority difference is exactly 2 (sonority(r) – sonority(p)), but [pt] would not be a good onset, since its sonority difference is 0.
- [s] is an exception to **Onset Maximization**. It should always go with the following syllable (so 'tapestry' is syllabified as 'ta.pe.stry')
- Any consonants not in the onset should go in the coda of the previous syllable.

You may use whatever method you like for achieving these principles. You may find it helpful to process the words in reverse order.

Task 1

Create a python file called syllabify.py, or open the one on CCLE, and use the template to write a function called syllabify(). syllabify() should take as input a list of phonemes, such as those in CMU, and output either a string or a new list (up to you), where the syllables are separated by "+".

absorption:	AH B + Z AO R P + SH AH N
accountant:	AH + K AW N + T AH N T
actors:	AE K + T ER Z
administering:	AE D + M IH + N AH + S T ER + IH NG
advised:	AE D + V AY Z D
aged:	EY JH D

Implement the syllabification principles given above. The file on CCLE contains a dictionary called 'son' which you may find useful. It simply holds the sonority value of each segment you'll encounter.

Task 2

Read in the file test.txt, syllabify each item (each line), and print the results to a file called output.txt. The first few lines should look like this:

```
absorption AH B + Z AO R P + SH AH N
accountant AH + K AW N + T AH N T
actors AE K + T ER Z
administering AE D + M IH + N AH + S T ER + IH NG
advised AE D + V AY Z D
aged EY JH D
...
```

Note that unlike the CMU file you read in last time, only spaces separate the spelling from the transcription. Also note that there are no numbers to worry about.

Task 3

Open evaluate.py. Make sure the two lines at the top are pointing to the correct files on your computer (the output you printed in Task 2, and the file correct.txt from CCLE).

Run evaluate.py. This script will check your syllabifications against the correct syllabifications for all words in the file. The bottom of your output should look like this:

```
...
Misparsed:   RAE+DIHK+LIY     as
             RAE+DIH+KLIY
Misparsed:   RAH+GAARD+LAHS   as
             RAH+GAAR+DLAHS
Misparsed:   RAH+PYUW+TAHD+LIY      as
             RAH+PYUW+TAH+DLIY
Misparsed:   RAHS+PEHK+TIHV   as
             RAH+SPEHK+TIHV
Misparsed:   TEYK+OW+VER      as
             TEY+KOW+VER
Misparsed:   AHN+FAOR+CHAH+NAHT+LIY       as
             AHN+FAOR+CHAH+NAH+TLIY
syllable level accuracy is    0.916364
word level accuracy is        0.897898
```

Look at the full results of evaluate.py. What kinds of mistakes is the algorithm making? How could you improve its performance?

What to turn in:

1) Your syllabify.py file
2) Your output.txt file
3) Your answer to the questions at the end of Task 3