LANGUAGE AND COMPUTATION - HOMEWORK 1

INTRODUCTION

In this assignment you'll be implementing a rule-based syllabification algorithm and using the algorithm to analyze speech of young children.

There are several provided files of transcribed speech, 'test.txt', as well as a number of text files of child speech zipped up as 'ChildData.zip'. The transcriptions are all in the same format, with one word on each line, followed by a space and phonemes (represented as one or two letters) separated by spaces:

```
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
…
```

All these files use an English-specific phonetic alphabet of 39 phonemes. You should check out the sounds that they represent by examining the file and by visiting the following website for examples of all the phonemes:

http://www.speech.cs.cmu.edu/cgi-bin/cmudict#phones

PART 1

Overview

First, you'll implement the syllabification algorithm that reads in a file of phonetic transcriptions and writes out a syllabified version of the file. The files you'll need for this part are 'test.txt', 'correct.txt', and the provided script, 'evaluate.py'.

Call your syllabification program 'syllabify.py' and have it read in a file from the command line and print out (to standard output) a syllabified version of the file. Once you're done you should run it like this:

```
python syllabify.py test.txt > output.txt
```

The symbol '>' tells the shell to redirect the output to a file 'output.txt', which will be saved in the same directory as your program. Your program should read in transcriptions like those above and print a syllabified version that inserts '+' between syllables in the same format:

```
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
…
```

After you've finished writing the program, you'll be able to compare your 'output.txt' file to a correctly syllabified version of test.txt, 'correct.txt', to see how well the algorithm syllabifies:

```
python evaluate.py output.txt correct.txt
```

This program will print out the differences between the two syllabifications and the accuracies calculated at the syllable levels and word levels. If you have correctly written your syllabification program, the last part of the output should be:

```
....
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
```

The next section describes the syllabification procedure.

Syllabification

First of all, what is syllabification? Syllabification is the process of parsing phonemes into syllables. Syllables are composed of three ordered constituents: onsets, nuclei, and codas. All syllables must have nuclei, which are usually vowels, but syllables vary in how many consonants occur in the onset and coda. The syllable 'strips', for example, has 3 consonants in the onset, and two in the coda, while the syllable 'a' has zero onset consonants and zero coda consonants.

Why should we care about syllabification? From a practical point of view, a segment's position in the syllable can greatly affect how it is articulated and pronounced. Consider how the [l]s in [feel] and [leaf] sound and are articulated, for example. Speech synthesis and recognition systems need to take into account a phoneme's position in the syllable in order to accurately model this variation. From a theoretical perspective, syllable structure plays an essential role in constraining possible words (phonotactics) within and across languages and thus is an essential component of our linguistic competence.

The ambiguity in syllabification concerns the consonants between nuclei – do they go in the coda of the first syllable or the onset of the second? In 'party' we split up the consonants, 'par-ty', but in 'deprive' we put them both in the second syllable: 'de-prive'.

Linguists have determined that phonemes fall into a number of **sonority** classes, and these classes influence syllabification. Phonemes may be categorized into five classes, which we'll number 0 through 4:

- 4 – the most sonorous, vowels and syllabic consonants
- 3 – [Y] and [W] as in yell and wow
- 2 – [L] and [R]
- 1 – nasals sounds [N], [M], [NG]
- 0 – all the other sounds, like [S], [SH], [B], and [P]

LANGUAGE AND COMPUTATION - HOMEWORK 1

In general, syllables are most sonorous in the middle (the nucleus) and drop in sonority outwards. The base code provided to you 'syllabify_base.py' has encoded each of the 39 phonemes' sonority levels.

What to Do

Write a syllabification program that implements the following Syllabification Principles by placing '+' between syllables. Note that this format doesn't explicitly mark onsets, nuclei, and codas in the output. However, it may be helpful to you to keep track of some of these constituents internally in your code.

Syllabification Principles:

- All vowels (sonority level 4) are syllable nuclei
- All initial consonant sequences are onsets, and all final consonant sequences are codas
- Onset Maximization: put as many consonants into the onset as possible so long as the difference in sonority between consecutive consonants in an onset is at least 2. This means, son(c2) – son(c1) >= 2 in a sequence c1 c2 in order for c1 c2 to be syllabified together in the onset.
- Always put [S] with the following syllable
- Any other consonants go into a coda

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

Once you have written your program, create an output file, check it using the 'evaluate.py' script and answer the following questions:

Questions

1) Give examples of two different kinds of errors in the output.
2) Why do these errors happen?
3) How could the syllabification program be improved to fix these errors?
4) **Extra Credit:** Try to improve the performance of your syllabification program. Be sure to explain what you did and how it helped!

PART 2

Overview

Now for the really fun part: you will use your syllabification program to analyze child speech data. Look at the provided files ending in '.actual' and '.target'. For each pair, the 'target' file shows the adult pronunciation of the word, while the 'actual' file shows how the child pronounced the word. These are real data from little kids' spontaneous productions at different ages. The format of the files is exactly the same as the test data. Provided are files at 4 different ages for two children, Georgia and Charlotte[1], whose ages (years;months.days) and corresponding files are given below:

(1) Georgia
   a. geo13 1;4.17
   b. geo20 1;7.23
   c. geo28 2;1.01
   d. geo43 2;10.11

(2) Charlotte
   a. cha13 1;4.11
   b. cha21 1;8.17
   c. cha35 2;2.22

[1] By the way, there is way more data for many children available – I just chose two kids who both had data over a range of ages, and for whom there were more than a few words at the earliest ages.

d. cha48 2;11.16

What to Do

Extend your program so that in addition to syllabifying the provided file, it prints out the relative frequencies (proportions) of syllable types in the file. Syllable types are strings C*VC* where C stands for any consonant (sonority classes 0-3) and V stands for any vowel (sonority class 4).

Hint: Once you have your syllabified word from Part 1, it should be easy to split that string by '+' to find the syllables. And then convert each of the phonemes in the syllable into either C or V, and keep track of how many times each has been seen using a dictionary.

When you run your program on geo43.cha.actual, you should get the following relative frequencies (these are printed rounded to two decimal points):

```
Frequency of CCV :       1.02
Frequency of CCVC :      2.39
Frequency of CV :       47.44
Frequency of CVC :      28.33
Frequency of CVCC :      3.41
Frequency of CVCCC :     0.34
Frequency of V :         7.00
Frequency of VC :        9.73
Frequency of VCC :       0.34
```

Some Background

By examining syllable structures across many languages, theoretical linguists have identified some Implicational Universals between syllable types that specify their relative markedness or complexity: for a pair of syllable types A, B: A ⇒ B (is more marked than B) if in all languages that have A, B is also found and there are some languages that have B but not A. Here are some of these universals:

a) CVC ⇒ CV
b) V ⇒ CV
c) CCV ⇒ CV
d) VC ⇒ V
e) CVCC ⇒ CVC

Researchers who study language acquisition have observed that the same universals that hold across languages seem to hold across stages of development: the more marked syllables tend to be acquired later by children. Such observations have supported the view that Universal Grammar encodes these universal implications and constrains the acquisition process in all languages.

Questions:

Once you've extended the program use it to examine the distribution of each of the child data files and answer the following questions.

1) What are the first syllable types the children attempt (syllable types in the earliest 'target' files)? What are the syllable types children attempt last (syllable types only attempted in later files)?
2) What are the first syllable types the children actually say (syllable types in the earliest 'actual' files)? What are the syllable types children start saying last? (syllable types appearing only in later 'actual' files)?
3) Comparing the proportions in pairs of actual vs. target files, which syllable types tend to be over-

represented in the children's pronunciations? Which tend to be under-represented?

4) What generalizations can you make about acquisition of syllable types? Which syllables do children seem to acquire earlier than others?
5) How do your generalizations from question (4) above relate to the Implicational Universals in (a)-(e) above? Are all the implications respected? Explain.

WHAT TO SUBMIT

- Submit (as an attachment) your program 'syllabify.py' with the extensions from Part 2, clearly commenting what each part of the code does and the extra credit, if applicable.
- Make sure to write your name on your program in a comment at the top of the code.
- Make sure to write in a comment at the top what version of python you are using, if not 2.7.
- Submit your answers to questions 1-3 from Part 1 and 1-5 from Part 2