Muhammad Umar Salman
21010241

NLP 703 SPEECH RECOGNITION ASSIGNMENT 1

Problem 1

Transcribe into ARPAbet symbols each of the following (note: if there is more than one pronunciation of the word, give ARPAbet pronunciations for the multiple pronunciations):

(a) monosyllabic words: why, what, how

Answer:

Why: /W/ /AY/

What: /W/ /AH/ /T/

How: /HH/ /AW/

(b) bi-syllabic words: family (pronounced as a 2-syllable word), welcome

Family: /F/ /AE/ /M/ - /IH/ /L/ /IY/

Welcome: /W/ /EH/ /L/ - /K/ /AH/ /M/

(c) tri-syllabic word: family (pronounced as a 3-syllable word)

Family: /F/ /AE/ - /M/ /IH/ - /L/ /IY/

(d) sentence: My name is Fred.

My Name Is Fred

/M/ /AY/ . /N/ /EY/ /M/ . /IH/ /S/ . /F/ /R/ /EH/ /D/

Problem 2

(a) Write out the phonetic transcription for the following words: /she/, /eats/, /some/, /meat/

She Eats Some Meat

/SH/ /IY/ . /IY/ /T/ /S/ . /S/ /AH/ /M/ . /M/ /IY/ /T/

Muhammad Umar Salman
21010241

(b) What effect occurs when these four words are spoken in sequence in a sentence? What implication does this have for speech segmentation of a sentence into words?

The effect of these words is caused as the last phoneme sound of a word is the same as the phoneme sound of the next word (IY, S, M). Thus, when these words are spoken in a sequence since the ending of the previous and starting of the current sound the same, it is spoken in a flow without a break, as if all the words are connected to form a single word. This causes a lot of problems for speech segmentation as the speech recognizer would find great difficulty where one word begins and where the other finishes thus not being able to accurately predict which phonemes are part of which words leading to inaccuracy in recognition.

Problem 3

Write a Python command line application to syllabify your English pronunciations using ARPABET syllabification. It will take an ARPABET transcription in array or string form and return a list with the syllables chunked. The minimum is for 100 predefined words. You can also consult CMU dictionary as in: http://svn.code.sf.net/p/cmusphinx/code/trunk/cmudict/cmudict-0.7b Make sure to validate the command line input and maintain it to find the resulted word.

Example:

Input: /W/ /AY/

Output: Why

Answer)

Files Attached:

- parser.py
- words.json (150 words)

Command: (Example of unclean input, Output: “This Language is gold”)

- python parser.py “/DH /IH/ S . /L/ AE /NG - /G/ //W/ /AH/ JH .IH/__?Z/. G/// /OW/ {L D}"

Muhammad Umar Salman
21010241

```python
import sys
import re
import json


def readInput():
    N = len(sys.argv)
    NameOfScript = sys.argv[0]

    if N == 1:
        print("WARNING: No Input given.")
        return None
    elif N == 2:
        return sys.argv[1]

    else:
        print("WARNING: Too many inputs.")
        print("Script takes 1 string as input.")
        return None

class Parser:
    def __init__(self, inputSentence):
        self.inputSentence = inputSentence

        with open('words.json') as f:
            WORDS = json.load(f)
        self.WORDS = WORDS

        self.PHONEMES = {value:key for key, value in WORDS.items()}

    def parseInput(self,):
        x = self.inputSentence.upper().strip()
        wordList = x.split('.')

        formattedWords = []
        for x in wordList:
            x = re.sub(r'_', ' ', x)
            x = re.findall(r'\w+', x)
            x = ' '.join(x)
            formattedWords.append(x)
        return formattedWords

    def retrieveWord(self,):
        inputWordList = self.parseInput()

        output = []
        for w in inputWordList:
            try:
                output.append(self.PHONEMES[w])
            except:
                output.append('<UNK>')

        return ' '.join(output).upper()

    def run(self,):
        print(self.retrieveWord())



if __name__ == '__main__':
    inputSentence = readInput()
    # inputSentence = "/DH /IH/ S . /L/ AE   /NG - /G/ //W/ /AH/ JH .IH/__?Z/. G/// /OW/ {L D}"
    # inputSentence = "/DH /IH/ S . /L/ AE   /NG - /G/ //W/ /AH/ .IH/__?Z/. G/// /OW/ {L D}"
    p = Parser(inputSentence)
    p.run()
```