Author: Ryan Timbrook
UW Net ID: timbrr
Project: Ling 473 Project 3
Date: August 23, 2016

Description:

This project implements a finite state machine that identifies syllables in Thai text. The main intent is to automatically identify syllable and word boundaries in the text in order to separate words with a single white space in order to tokenize the text.

Approach:

I chose to implement this project using the Python programming language version 3.4. In order to handle the encoding necessary to process the utf-8 input file I used the python codecs module which encodes and decodes files for input/output processing. To handle the final state break insertions of white space requirement I used a dictionary object which recorded the previous characters ending index marker along with the next characters starting index marker. These markers were used to slice a copy of each word of the Thai line from the input string into a new string which at completion of the FSM was returned to the main calling function. When structuring my finite state transducer function I ran across three challenges which required specific programming to handle the parsing of the Thai input text.

The first was with the state transitions to final states 7, 8 and 9 required that the for loop not advance prior to entering these states from the previous state otherwise the transition from these states to the next state incremented the character being evaluated passed the predefined state transition mapping specification causing the current state to raise an exception.

The second was that it was necessary, based on my implementation design of inserting space breaks into the Thai line, that I evaluate the dictionary object's starting index value compared to the total input length in order to capture the last word of a line if its length didn't naturally transition through the final states 7, 8 or 9.

The third was with the transitioning to states 7 and 8 during actions upon the final character of the input string. This transitioning to these states was causing a falsely implemented line break at the end of five of the input lines parsed. I found the solution to this issue in the class go post. It required evaluating the character index to the length of the string and if it was the last character of the string the state should break out of the loop not completing in final states 7 or 8.