S'21 CSE323/EST323/ISE323 : Assignment 2
Virtual Keyboard Decoder
Due : 11:59 PM EST on March 30th, 2021

Aim: implement a simple statistical decoder using the dual Gaussian spatial model and a unigram language model.

Working folder:
1. An iPython notebook template: A2.ipynb.
2. A "Data" folder that contains data.txt, keyboard.csv, and unigram.dict (a unigram language model).
    a. data.txt: touchpoint data from user study where subjects typed short phrases on a smartwatch.
    b. keyboard.csv: the position of the top left corner of each key in millimeters. The key width is 3mm and the height is 4mm.

Procedure:
1. Review lecture 7/8/9 slides and recordings.
2. Read input files. Process data.txt to a list of touchpoint collections, each containing touchpoints $s_1, s_2, \ldots, s_n$ for some word.
3. Use step a-d to calculate the best word w* that maximizes $P(w|s_1, s_2, \ldots, s_n)$ for each touchpoint collection.

$$w^* = \underset{w}{argmax} P(w \mid s_1, s_2, \ldots, s_n)$$

$$P\left(w \middle| s_1, s_2, \ldots, s_n\right) = \frac{P\left(s_1, s_2, \ldots, s_n \middle| w\right) P(w)}{P(s_1, s_2, \ldots, s_n)}$$

    a. For each touchpoint collection, search the language model to get all possible words and their corresponding probabilities, P(w).
        i. Use the length of the correct word to filter possible words
        ii. You may also use the first and/or the last touchpoint to further narrow down possible words
    b. For each possible word, calculate $P(s_1, s_2, \ldots, s_n|w)$ using the dual Gaussian spatial model. Assume w consists of n letters: $c_1, c_2, \ldots, c_n$, then

$$P\left(s_1, s_2, \ldots, s_n \middle| c_1, c_2, \ldots, c_n\right) = P\left(s_1 \middle| c_1\right) P\left(s_2 \middle| c_2\right) \ldots P(s_n | c_n)$$

    Spatial model: σ^2 is a diagonal covariance matrix, where $\sigma_x^2 = a + b * W^2$,

    $\sigma_y^2 = c + d * H^2$ μ is the center of key c_i.

$$P(s_i|c_i) = P(s_i|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} exp\{-\frac{1}{2\sigma^2}(s-\mu)^2\}$$

    c. Calculate P(w) * $P(s_1, s_2, \ldots, s_n|w)$ for each possible word.
    d. Choose the word that maximizes P(w) * $P(s_1, s_2, \ldots, s_n|w)$ as the decoded word.
4. Compute the literal string for each touchpoint collection, which is the string determined by the key boundary without leveraging any language model.
5. Calculate success rate for both decoded words and literal strings as a metric to evaluate the decoder. To simplify the problem, we calculate the success rate as the number of correct words over the total number of words instead of the edit distance of the sentences.
6. In a file named results.txt, output the first line:

    success_rate (decoded_words), success_rate (literal_strings)

    Then, for each word, output a line to results.txt:

    correct_word, decoded_word, literal_string

Submission:
1. Assignment should be submitted on Blackboard as a single .zip file in the format: <First Name>_<Last Name>_<SBUID>_A2.zip. For example Yan_Ma_111122233.zip

2. The submitted zip file should contain only A2.ipynb (iPython notebook) with relevant code, results.txt, and an optional README.txt with anything you want us to know.

Grading Criteria:

1. We will provide an example dataset and the results of that dataset. You could compare your results with the provided results. You will get full points if the TA's results (success rates) - your results <= 3%.
2. We will deduct 5 points for not following file naming conventions.
3. If you submit assignments after the deadline, your maximum grade will reduce by 20 points per late day.