Maintainer’s Manual

robertperry.py

co-written by Colin White and Will Longabaugh

Overview:

The robertperry module takes a word (argument ‘w’ in all method headers) and returns a list of candidates closely related to the word the user might have meant to type. If the user typed a valid word, at the front of the candidate list will be that word. If the user didn’t type a valid word, at the front of the candidate list will be the word the algorithm determines the user most likely meant to type.

Methods are called in reverse order of their listing in the source code (as Python code normally goes).

\_\_init\_\_(self, trie) :

trie – the trie the algorithm will be running on.

Initializes the candidates list and sets the maximum amount of candidates that will be returned.

complete(self, w, index, node, score):

w – the string the user entered

index – an index from which we will be attempting to tack on the rest of the word

node – the node we are currently at from the searchDown method

score – the score we’ve gotten from the searchDown method

This method takes the current state of an attempt at an autocorrected word from searchDown in the form of a node we’re at in the trie. If the word attempt is at an index less than the length of the string the user entered, it will attempt to tack on the rest of the string from nodes in the trie. If it can’t, the method returns. If it can, and the resulting node is the end of a valid word, it calculates the score of the word from the score provided by searchDown and the word frequency. It inserts this word-score tuple into a list and sorts it. If the list is bigger than the size set by the initialization method, it pops a candidate from the end of the list.

searchDown(self, w, index, node, neighbor, err, error, score):

w – the string the user entered

index – an index we’re currently at in an autocorrect attempt

node – the node we’re currently at (corresponds with the word we’re trying to build in an autocorrect attempt)

neighbor – broadly corresponds with whether the last correction made was to a neighboring letter of the letter the user entered on the keyboard

err – current number of errors on this autocorrect attempt

error – max number of errors allowed

score – score of current autocorrect attempt; calculated by types of errors made

This method does most of the work on making an autocorrect attempt. First it checks the amount of times searchDown has been called; if it has been called more than 15,000 times, it exits the method. If not, it recursively builds autocorrect attempts, stopping when the current number of errors reaches the maximum number of errors allowed, or when ‘index’ is equal to the length of the string the user entered. Types of recursions are performed in this order:

1. adding a letter
2. recurring without an error
3. replace letter with neighbor
4. replace letter with non-neighbor
5. replace letter with an apostrophe
6. transpose letters
7. delete neighboring letter
8. delete non-neighboring letter

Additions to the current score are based off of the type of error made:

+ 0 – recurring without error

+ .75 – apostrophe addition

+ 1 – neighbor replacement, transposition

+ 1.05 – letter addition, neighbor deletion

+ 1.15 – non-neighbor replacement

+ 1.2 – non-neighbor deletion

searchDownHead(self, w, node, neighbor, error):

w – the string the user entered

node – header node of the trie

neighbor – Boolean specifying whether this pass will start with a neighbor or not

error – max number of errors on this pass

This method handles base cases of autocorrect attempts sent to the trie. All error types are attempted as possibilities for the first letter of autocorrect attempts.

findCandidates(self, w):

w – the string the user entered

returned: self.candidates – the list of candidate words

This is the method called from the front end. First, the input is one letter, it checks to see if it is next to ‘a’ or ‘i’, and if so, adds those to the candidate list. Then it checks to see if valid words can be created by splitting the input in two. Then it calls searchDownHead on increasing distance lengths until we have found 4 candidates for an autocorrected word. It returns the candidates found.