Letter Prediction for a Scanning-based AAC System
CISC106 Summer 2013
Due July 29, 2013, 11:55pm,

People who use AAC (Augmentative or Alternative Communication) Devices to speak often communicate at an extremely slow rate. This is even more so the case for someone who must select their desired input via scanning.

Consider the following set-up for row-column scanning in an AAC device. Regular row column scanning goes through each row – when the cursor gets to the desired row, the user hits a switch and the scan starts across each column and the desired letter can be selected by hitting the switch when the scan gets to that letter. In the set-up shown here, before the scan goes into the regular keyboard, it first cycles through a set of frequent letters which can be selected rather quickly. (This row is shown above the others so it is easy to see that it is different.

Dynamic Changing Predicted Letters go Here – Scan goes here first

E	A	R	S	T

Static Regular Keyboard – Scan goes here next

SPACE	.	,	?	!	;	:	-	“	_
Q	W	E	R	T	Y	U	I	O	P
A	S	D	F	G	H	J	K	L	$
Z	X	C	V	B	N	M	*	(	)
1	2	3	4	5	6	7	8	9	0
&	@	’	%	/	[	]	{	}	#

Suppose that SCAN_TIME is the time that it takes the scan to go from one place to the next. Then the time it takes to select the X is (5 * SCAN_TIME) + (2 * SCAN_TIME) because it is 5 steps down and 2 steps over. Similarly, the ! can be selected in the same amount of time but it would be (2 * SCAN_TIME) + (5 * SCAN_TIME) – 2 steps down and 5 steps over. So you can see that having the dynamic keyboard to have prediction letters my help the user to decrease the overall scan time for each word. (You do not need to implement or calculate anything regarding the scan time. This is just to illustrate and to show you how it works in an AAC device).

The project

Your job is to develop a program to do letter prediction in order to increase communication rate for someone who uses such a scanning-based AAC System. You will be provided with a set of files that contain most of the common nouns in the English vocabulary. These files will be used to search for the words that start with the letter chosen so far by the user. The prediction should be based on word frequency, for example, if the first letter the user types is "t" your program should search within the set of all words that start with "t" and get the 5 most common letters that come after "t". After the program finds those letters, it should display them on the screen (which would be shown in the dynamic keyboard of an AAC device) so the user can – or not – pick one of the letter your program predicted.

In order to check how good the prediction is doing, the program should score points if a letter is predicted correctly (if the letter the user picks is within the 5 ones suggested) and the points should be higher the more a letter is on the left (since it would be faster for an AAC user to get to it).

Vocabulary Files

The subset of words from the English vocabulary will be provided to you. Each letter starting words are within a separate file (e.g. a.txt, b.txt, c.txt…z.txt). A file named readFile.py contains a method called **read_file(**file_name**)** which receives a string representing the letter (e.g. "d") and returns a list containing all the words that start with the letter "d". You can use this method in order to load the set of words your program will be using to predict letters for each time we run it. Please watch out for the fact that all the letters in the set of files are lowercase.

Program Tasks

You need to make sure that your program is functional. For this reason, you should write small functions that will take care of each task of the program. Please attempt to the fact that the functions described here are the minimum modularization your program should do. You are more than welcome to break them into smaller functions to have an even more functional program.

1) Write a main function that should be called when the program is run. This function is responsible for requesting input from the user and calling other functions in order to execute specific tasks.
2) Write a function that will calculate the 5 most probable letters given the substring provided (the subset of characters that the user has provided so far.)
3) Write a function that checks if what the user typed is indeed a letter from the vocabulary. If the user types anything different (a number or another character) the function needs to notify that a letter is expected, disregard it and ask for input again (even hyphen, we will not be dealing with compound words in this project).
4) Write a function that shows the 5 letters that the system will suggest to the user. This function should print the letters on the screen.

5) Write a function that calculates the points the program has made so far and the total points it accumulated when the user finishes the word. The points should be calculated as follows:
    a. If the letter the user has chosen is suggested on the left most position, the program scores 50 points. If it appears in the second position from left to right the program scores 40 points, and so forth until the 5th position, where the program score 10 points. If the letter the user types was not suggested, the program does not score points for that letter.
    b. The points scored for each letter need to be accumulated. In the end of the word typing the program should show the total points scored.
6) Write a function that checks, after the user types all the letters and finishes using the letter prediction system, if the word (s)he typed exists in the vocabulary. If not, show a message saying that the word is not in the list.
7) Write a function that calculates the system's accuracy on predicting the letter of the word. After the user finishes using the system for a word, then a message should be shown telling the accuracy of the system.

P.S.: Remember that Python is case-sensitive and that all the words in the set of files are case-sensitive, so if the user types "P", your program will not find the file that contains all the words starting with "p" and it should, so make sure address this issue.

What to turn in

You will need to turn in a copy of your program via Sakai. In addition, make sure you turn in a write-up explaining with words what every function that you wrote does (**15 points**). This is called a README file (and you should actually name it README.txt) and it is a good practice that all programmers should perform. Although you will be providing the design recipe for each function, a set of instructions on how to run your code and the explanation of what each function performs in a high-level language helps to document your code.

Presentation

We will possibly have the opportunity to present the project in class. This will be defined based on the schedule. If we do have the chance, partial credit of the assignment will be given based on the presentation evaluation.