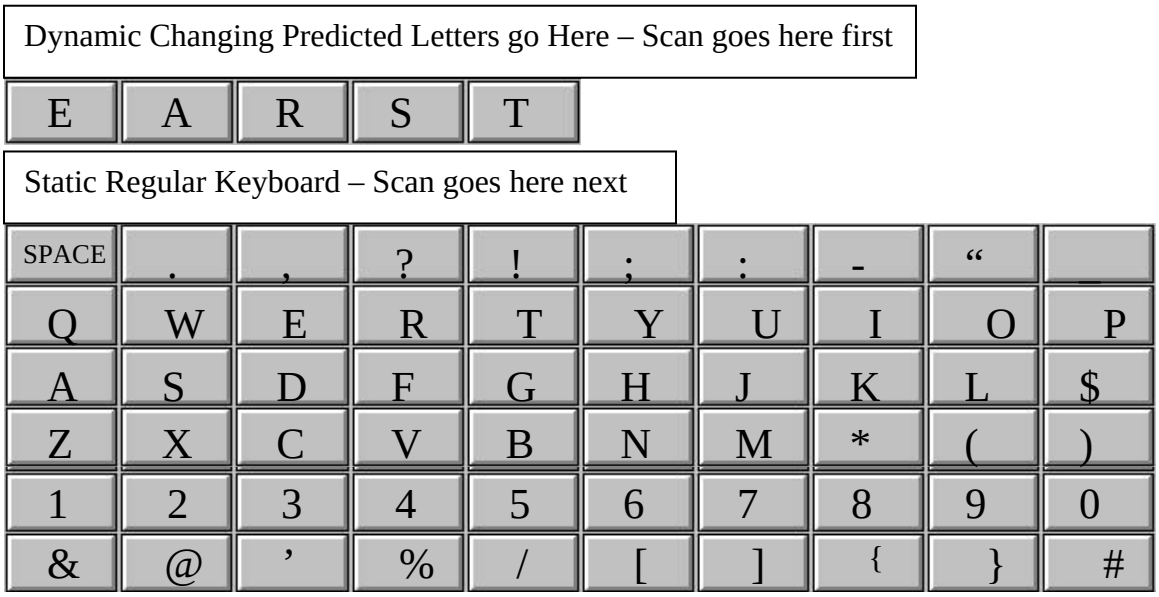
Letter Prediction in a Scanning-based AAC System
CISC882 Fall 2012
Due October 10, 2012, 11:55pm,
Competition October 11th!

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.



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.

Your job is to use n-gram modeling (at the character level) to do letter prediction in order to increase communication rate for someone who uses such a scanning-based AAC

System. You get to make all of the decisions – what corpus to use, how large of a corpus to use, what n to use, how to smooth the n-gram model (if necessary) how to clean up the corpus for training purposes, what exactly to train on, etc…

I think an interesting thing to try would be to try to do word prediction, and then predict letters according to the probabilities of different words.

We want to test your models against each other. To accomplish this I provide a zip file that has in it a perl script that calculates the amount of time it takes to type a text using the scanning keyboard. You will need to provide this script with a file containing the row of predictions your program calculates. There is an example there and the comments in the script are quite good. Many many thanks to Amy Siu who took this class several years ago and wrote the evaluation software!!!!

This we will use in a competition – you want to use whatever means you can to try to win the competition!

Testing

We will want to test the predictors you create against each other. At the same time, we would like to see how the training corpus affects the outcome. To this end, once you have selected your training corpus, separate out a contiguous 400 words to be used in the common testing set. You may not use this portion of your corpus in anything else you do – it is simply for testing purposes.

Close to the due date of the assignment, we will take all of the training corpora, and, for each corpus, each member should produce a corpus report

- time it took to type with regular keyboard
- time on your predicting keyboard
- plot the % time savings (loss) for each.

Also, indicate which testing corpus was your own.

Prior to that final exercise, there are a couple of tests you should run on your own corpus. First, you should test your predictor using 10-fold cross validation. But, there is a question of how many letters should be in the prediction window. Intuitively, as the prediction window is longer, it is more likely to contain the letter you wish to predict. On the other hand, if the window is too big, there is likely to be a degradation of performance because it would be actually faster to choose a letter low on the prediction list from its actual position on the regular keyboard instead. To see this:

- perform 10-fold cross validation on your corpus
- use a window size of 1, 2, 3, 4, 5, and 6 in this testing.
- Compare each against the speed of typing on a regular scanning keyboard.

What to turn in

You will need to turn in a copy of your program, along with the test results we will need for the competition comparison. In addition, make sure you turn in a write-up that helps us understand what you did. This write up should definitely include anything that I have no way of telling from your code. E.g.,

- What corpus did you use
- How large was your corpus

Some other kinds of things you should make sure you include are interesting facts about the way you approached the problem. E.g.,

- What n-gram
- How did you fill the prediction window
- How did you smooth

Your write-up should also include your own analysis of how well your program did and why. Let me know about things that you tried but discarded because they did not seem to work as well. Generally when you make decisions about how the program should run, let us know about what led you to make the choices that you did.

There are many different corpora available that you might want to use for this project. The nlp lab has some stored on the ee-cis machine named blizzard. From the ee/cis network it can be reached from /blizzard/corpora/

Take a look around that directory to see what is available – there are various readme files spread around. Also, feel free to use any other corpora you may find anywhere!