Training data

Every machine learning algorithm needs sets of data, which is called training dataset. Training data sets and input data together we believe that construct predictive relationship between input data and output action. Training data should be as closer to the actual output as possible. A well pre-labeled and impartial data will help trained classifier to perform more accurate prediction. Amount of training data set depends on complexity of the concept of predictive model.

Our algorithm required training data sets of alphabets as we are predicting alphabets. Concept device designed to predict all alphabets only, so our training dataset would cover all alphabets with enough frequency. We designed training text dataset from pangram sentences which covers all alphabets in meaningful sentences. (available on)

"The five boxing wizards jump quickly"
"Sixty zippers were quickly picked from the woven jute bag"
"Sphinx of black quartz: judge my vow."

Layer Min_Max Matrix

As discussed in setup procedure, we divided computer keyboard alphabets in three layers : Upper layer (keys:'q','w','e','r','t','y','u','I','o','p'), middle layer (keys: 'a','s','d','f','g','h','j','k','l') and lower layer (keys: 'z','x','c','v','b','n','m'). This 3 X 16 matrix contain data of flex sensor input for each layer row wise. Left part of half matrix (3X8) contains minimum value of flex sensors while typing in particular layer by fingers. Remaining half part contains maximum value of flex sensors for the same process. By default we set all values to 2000 because our flex sensor only goes from 100 – 1200, that means after training if we find any two values 2000, we can conclude non used finger in particular layer.

	F_min_0	F_max_0	F_min_1	F_max_1	F_min_2	F_max_2	F_min_3	F_max_3
L_0	571	629	474	810	463	752	474	857
L_1	401	658	618	846	506	879	521	973
L_2	582	662	774	857	615	890	589	1078

	F_min_4	F_max_4	F_min_5	F_max_5	F_min_6	F_max_6	F_min_7	F_max_7
L_0	557	853	474	1295	155	984	387	600
L_1	701	987	557	1364	209	973	376	759
L_2	698	1157	1074	1327	2000	0	387	832

Training Phase:

Input Observations: Q1,Q2,Q3,Q4,Q5…Qn where Q1,Q2,…Qn are flex sensor inputs from Arduino board.

Minimum = Qmin = min(Q1,Q2,Q3,….Qn)

Maximum = Qmax = max(Q1,Q2,Q3,….Qn)

Output: $[Layer_x][Finger\_min_y] = min([Layer_x][Finger\_min_y] , Qmin)$

$[Layer_x][Finger\_max_y] = max([Layer_x][Finger\_max_y] , Qmax)$ where $x \in \{0,1,2\}$ that denotes the layer and $y \in \{0,1,2,3,4,5,6,7\}$ that denotes the fingers.

System In Use:

Input Observations: Q1,Q2,Q3,Q4,Q5…Qn where Q1,Q2,…Qn are flex sensor inputs from Arduino board.

F where F denotes finger input from Arduino board having values from 0 to 7. Values 0 to 7 denotes LL, LR, LM, LI, RI, RM, RR and RL respectively.

Qavg = avg(Q1, Q2, Q3 … Qn) where avg() gives arithmetic average.

Output: $\{I: [Layer_i][Finger\_min_y] <= Qavg <= [Layer_i][Finger\_max_y] \parallel i \in \{0,1,2\}, y = F\}$

Finger Vs Alphabets occurrence matrix

Our algorithm selects most probable alphabet for that finger in that layer. To achieve this goal, matrix is constructed by monitoring occurrence of alphabets by finger and updating it in format like alphabets in row and finger in columns. This matrix updates in both phase training phase and

when device is in actual use. It is 27X10 sized matrix with labeled finger name in first row and alphabets in first column. Second column filled with layer information in form of '0','1' or '2'. '0' represent Upper Layer(UL) , '1' represent Middle Layer(ML) and '2' represent Lower Layer(LL)

LETTER	LAYER	LL	LR	LM	LI	RI	RM	RR	RL	Total_Occurrences
q	0	0	7	0	0	0	1	0	0	8
w	0	0	23	0	1	1	1	0	1	27
e	0	1	0	61	0	0	2	2	3	69
r	0	0	0	2	28	1	1	0	0	32
t	0	0	0	0	46	1	0	0	1	48
y	0	0	0	0	2	12	6	0	1	21
u	0	0	1	0	0	11	17	1	0	30
i	0	0	0	0	0	1	34	5	3	43
o	0	0	0	0	2	2	22	43	0	69
p	0	0	0	0	1	0	2	11	0	14
a	1	36	10	0	0	1	0	0	3	50
s	1	1	31	3	0	1	0	0	2	38
d	1	0	0	8	10	0	0	0	1	19
f	1	0	0	0	14	0	0	0	1	15
g	1	0	0	1	10	2	0	0	1	14
h	1	1	0	1	0	33	1	0	0	36
j	1	0	0	0	0	9	0	0	0	9
k	1	0	0	0	0	0	14	4	0	18
l	1	0	0	1	2	0	2	27	0	32
z	2	5	3	0	0	0	0	0	0	8
x	2	0	2	6	0	0	0	0	0	8
c	2	0	0	8	10	0	0	0	1	19
v	2	0	0	0	14	0	0	0	0	14
b	2	0	0	0	0	15	1	0	0	16
n	2	0	0	0	2	34	4	0	2	42
m	2	0	0	0	0	21	2	0	1	24

Training Phase:

Input Observations: F0,F1,…F7 , where F0,F1,…F7 are inputs from Arduino board for fingers LL, LR, LM, LI, RI, RM, RR and RL respectively.

L0, L1 and L2 where L0, L1 and L2 are identified layer from Layer Min_Max Matrix.

	C0,C1,….Cn , where C0,C1,….Cn are single character input from training data.
Output:	FAM[Fx][Cy] = FAM[Fx][Cy] + 1 where x € {0,1,2,3,4,5,6,7} and y €{0,1,2,…n} and FAM is Finger Vs. Alphabet Matrix

In Use :

Input Observations:	F0,F1,…F7 , where F0,F1,…F7 are finger inputs from Arduino board L0, L1 and L3 where L0, L1 and L3 are Identified layer from Layer Min_Max Matrix
Output:	Set {C0,C1…Cn} = FAM[Lx][Fx] where set{} would be sets of alphabets.

Transition Matrix

To make prediction more powerful, we use probability of transition of alphabets in words. It is 26X26 sized matrix with considering labels alphabets from a-z in row and column both. This matrix contains occurrences of alphabet to the following alphabet. Purpose of building this matrix is to predict next probable alphabets based on current alphabet input and also remove unwanted zero probability alphabets.

This matrix is already constructed based on words from a dictionary, which will be used to predict the words. This matrix also gets updated when device is in actual use to get more accurate prediction. (Refer Image on Page#18)

Training Phase:

Input Observations:	C_{CUR} , C_{PRE} where C_{CUR} is current character and C_{PRE} is previous character typed by user.
Output:	$[C_{CUR}][C_{PRE}]$ = N+1 where N denotes occurrences of this combination of characters.

System In Use:

Input Observations:	C_{CUR} , C_{PRE} where C_{CUR} is current character and C_{PRE} is previous character typed by user.
Output:	$N = [C_{CUR}][C_{PRE}]$ where N denotes occurrences of this combination of characters.

Dictionary

One dictionary is maintained to predict final word based on input alphabets. This is pool of words collected from open source Shakespeare's work. Concept device only predict words from dictionary only.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	0.001892	0.025652	0.065601	0.046468	0.001472	0.00778	0.028595	0.003995	0.043524	0.000421	0.010303	0.129941	0.041211	0.143398	0.001051	0.03196	0.001682	0.135198	0.059294	0.158116	0.016511	0.014087	0.006098	0.002944	0.018923	0.003765
2	0.147996	0.013361	0.006166	0.002055	0.157246	0	0	0.001028	0.121274	0.008222	0	0.147996	0.00925	0.003083	0.113052	0.001028	0	0.112025	0.04111	0.006166	0.090442	0.002055	0.001028	0	0.014388	0.001028
3	0.127937	0	0.00238	0.001865	0.140619	0	0	0.112272	0.068631	0	0.049981	0.044759	0.000373	0.000746	0.20552	0	0.001492	0.045132	0.013955	0.112272	0.039537	0.000373	0	0	0.012682	0.000373
4	0.087802	0.003351	0.004692	0.020107	0.313673	0.002011	0.016756	0.002681	0.229223	0.005362	0	0.019437	0.009383	0.004692	0.065013	0.00134	0.00067	0.044236	0.069705	0.002011	0.051609	0.018097	0.007373	0	0.020777	0
5	0.057715	0.007938	0.051596	0.110633	0.025302	0.014387	0.014057	0.002811	0.006946	0.000331	0.002315	0.059368	0.035059	0.134116	0.00463	0.019348	0.005623	0.181908	0.152968	0.044154	0.0043	0.018852	0.011411	0.02679	0.006946	0.000496
6	0.116395	0	0	0	0.14393	0.093867	0.001252	0.001252	0.232791	0	0	0.068836	0	0	0.145181	0	0	0.073842	0.007509	0.033792	0.068836	0	0	0	0.012516	0
7	0.109019	0.002973	0	0.000991	0.258672	0.001982	0.019822	0.102081	0.096135	0	0.000991	0.02775	0.004955	0.048563	0.0555	0.000991	0	0.120912	0.052527	0.012884	0.061447	0	0	0	0.021804	0
8	0.195502	0.002595	0.000865	0.002595	0.24654	0	0	0	0.155674	0	0	0.012111	0.008651	0.013841	0.185986	0.002595	0.000865	0.034602	0.010381	0.056824	0.038927	0.000865	0.00346	0	0.024221	0
9	0.049222	0.016337	0.085409	0.030739	0.053113	0.01751	0.030156	0.000195	0.001362	0.000973	0.00428	0.04786	0.028599	0.251751	0.114981	0.017899	0.001556	0.030739	0.081323	0.083852	0.003696	0.035409	0.000195	0.003502	0	0.009144
10	0.183206	0	0.007634	0	0.267176	0	0	0	0.045802	0	0	0	0	0	0.244275	0	0	0	0	0	0.251908	0	0	0	0	0
11	0.050761	0.015228	0	0.002538	0.352792	0.010152	0.005076	0.005076	0.225888	0.005076	0	0.027919	0.007614	0.048223	0.027919	0.005076	0	0.010152	0.159898	0.007614	0.010152	0	0	0	0.022843	0
12	0.137247	0.004277	0.004277	0.02916	0.21112	0.006221	0.00311	0.000389	0.173017	0	0.005832	0.111198	0.005054	0.002333	0.103421	0.005443	0	0.001166	0.03888	0.031104	0.034215	0.008942	0.001166	0	0.082426	0
13	0.201893	0.037224	0.001893	0.000631	0.258675	0.001893	0.000631	0	0.148896	0	0	0.001893	0.04164	0.003785	0.116719	0.109779	0	0.000631	0.032808	0.001262	0.027129	0	0.000631	0	0.011987	0
14	0.06789	0.001311	0.065531	0.090957	0.104063	0.011533	0.180341	0.002097	0.068676	0.002621	0.011271	0.006029	0.003408	0.023591	0.032241	0.000786	0.001311	0.001835	0.114286	0.169332	0.014155	0.01363	0.001311	0.000262	0.010747	0.000786
15	0.013802	0.014583	0.029427	0.026042	0.003125	0.010417	0.021875	0.002604	0.009896	0.001563	0.014063	0.065146	0.067188	0.271094	0.034635	0.034896	0	0.153906	0.040625	0.045573	0.068229	0.024479	0.03125	0.004948	0.007552	0.002083
16	0.133643	0.001162	0.001162	0.003486	0.166182	0.000581	0.001743	0.051133	0.651011	0	0.000581	0.09878	0.004067	0.000581	0.124927	0.053457	0	0.174317	0.030796	0.048228	0.037188	0	0	0	0.006392	0.000581
17	0.009804	0	0	0	0	0	0	0	0.009804	0	0	0.019608	0	0	0	0	0	0	0.009804	0	0.95098	0	0	0	0	0
18	0.138101	0.007645	0.019236	0.031073	0.236745	0.005425	0.017263	0.00148	0.133169	0	0.014057	0.011837	0.024908	0.021948	0.096178	0.007398	0.000247	0.02762	0.077682	0.054994	0.021455	0.01233	0.002713	0	0.036252	0.000247
19	0.042516	0.003865	0.046732	0.003162	0.15636	0.003865	0.001405	0.066409	0.120169	0	0.011244	0.01546	0.011595	0.003162	0.060436	0.052354	0.002108	0.001757	0.087491	0.219255	0.070274	0.000351	0.008784	0	0.011244	0
20	0.094991	0.002076	0.008824	0.000779	0.200623	0.001038	0.000779	0.064625	0.253309	0	0.000026	0.017649	0.006488	0.003114	0.076045	0.001557	0	0.088762	0.06722	0.030625	0.041786	0.000519	0.004412	0.00026	0.033999	0.00026
21	0.055936	0.0371	0.047374	0.027968	0.048516	0.008562	0.032534	0	0.04395	0.000571	0.002854	0.076484	0.059932	0.122717	0.003425	0.037671	0.001142	0.166667	0.115297	0.099886	0.000571	0.001142	0.000571	0.002283	0.003995	0.002854
22	0.139535	0.001292	0	0.002584	0.528424	0	0.001292	0.001292	0.249354	0	0	0	0	0	0.065891	0	0	0.001292	0.003876	0	0.002584	0	0	0	0.002584	0
23	0.233533	0.005988	0.003992	0.003992	0.201597	0.005988	0	0.005988	0.195609	0	0	0.015968	0.001996	0.051896	0.111776	0.003992	0	0.031936	0.053892	0.007984	0	0	0.003992	0	0.005988	0
24	0.090426	0.005319	0.12766	0	0.12234	0.005319	0	0.031915	0.111702	0	0	0	0.010638	0	0.005319	0.271277	0	0	0	0.159574	0.042553	0	0	0	0.015957	0
25	0.047619	0.032967	0.047619	0.029304	0.157509	0	0.007326	0.003663	0.087912	0	0	0.036608	0.080586	0.047619	0.080586	0.054945	0	0.032967	0.150183	0.029304	0.014652	0	0.025641	0	0	0.010989
26	0.196262	0.009346	0	0.018692	0.401869	0	0	0	0.121495	0	0	0.018692	0	0	0.102804	0	0	0.009346	0.009346	0	0.028037	0	0	0	0.028037	0.056075

(http://www.rinkworks.com/words/pangrams.shtml (pagramword))