Jacob Bohac
ELEC 465
Homework #2

Program Output

```
Part (a) - Random 4 letter words

cvln qfkf fkvf qocp ymsz szzv aiti bcid zurr bdwg
pumf iqug cmiu mhso rlws phwr bpie shlh bzol rjst
xcpl jhca uavk hrbk ilpa scjv dxhu ibpi devo oxoi
ylsh cvsk jjmd nvzq vimd jbno fjct gseg eyoi wivf
rhig fjwa rifc jtqq eumk nstt qjbm rwrk gcql lonf
yuhj pyzt sngh fzaw iekb aclg gdtr tiwr dgas gbmy
puiw tkud ofer jrzp utin dhgg ngbt hpty jdvc opfe
ulvf cvuy qfnt otcb zdwg urhd xchl tmrq znvb ksbc
xovn hzqi emqy fxbc zipt whjv wgyg yzix pfkw faej
ovku slyu wops vaqr gozh qhef oret rkfg hrca cawy


Part (b) - 4 letter words from probability model

tdhp ppml ottk oliu nata hsri uahm mwgk coct ipia
nele rtdh uaie tcor eahs woeb hiwr bgso mgwg ebob
daha eatl aeey tlae uaya isov coeh rulu atze ttpu
aytu lyci acii vbha rnkl mvkm skso elng lkex kiil
msul ueno tadi yoyr arkh eapr lwqu ibiu ufis nalk
aota etsg oepv eipp hjor maok haev bpid ifdo dyyt
eorl alek utdl utue wdea toee uiua iuvo npbp efdc
dhna elgc omem djsa trde obte tyxa fcdk lslo fttx
ialn lhog btnr yjas iatp cadn tpfd nccy eonp aeac
bnua avuk vsca tgpp aatn dani pdar ibtl ornp pldo


Part (c) - single letter context

gerb onto loki stan adel dayt unki boli ador llfa
enka looo rtzz yvit memo dude uryn eell adoo rean
cama aiar ools ilal care idyl enux tana buda bass
oana kill oaba dyli neim humi rllt reng idgr pril
thea lona aung ryum nile ussh zeco arau ckun ckig
ecoz dono draf otho area ekul stax erck tuna myro
rtth peas umpe tsor popu hone pale yayr yser bela
redo duit lust rean here mock ledo mope aute peak
czet geep pabe rash ixyu oner iead idem ckel rofl
serk empo lope etyo ayon babe lire eura anke anan
```

Part (d) - two-letter context

dosh **move** orbo aage **ripe** **dunk** **last** bbes ungl waga
atea arsa leon alse atec vail irye adea undy essd
balk ully ormy ucho ankt **ramp** arpa **hope** **balk** lage
oulp xyiu eete **cyst** inka rame **limp** **alto** eros **envy**
nkml yogy otto ange **lode** **axes** eont **eden** **cyst** **bolt**
real **rime** **from** roax kitt tema eept balt heah urkm
diew neff **adam** odeu suct **exam** **chit** rpaw hown alea
illa iate waya zeub lial eldy eape hont aste **rope**
rmyt rrev milo **boss** teld whup **tong** jeam avet dest
wehr uhnl **cyst** vamo oyaw bita oule rtyp gaor basp

In part (a) the letters were randomly selected, and did not produce any words that make sense. Part (b) used a probability model to select the letters, and while it also did not produce any sensible words, it had a higher vowel to consonant ratio that gave some pronouncable 4 letter strings. Part (c) used a one letter context probability model which fared much better that the previous methods, producing 25 recognizable words. Part d improved on that by using a 2 letter context, and produced 27 words.

It is apparent that keeping additional context improves the efficiency of the probability model, but we also see that the amount of resources required for the straight forward implementation grows exponentially, which makes it unreasonable for large alphabets, or large contexts.

Appendix: Word generator program

```
#include <iostream>
#include <iomanip>
#include <stdio.h>
#include <stdlib.h>
#include <string>
#include <time.h>
#include <fstream>

using namespace std;

int main(){
        int i,j,k;
        string str;
        srand (time(NULL));
        string line;
        int weights[26];
        double probs[26];
        int total = 0;
        ifstream file("4letters.word");
        int singleContextWeights[26][26];
        int doubleContextWeights[26][26][26];

        for(i=0; i<26; i++){
                for(j=0; j<26; j++){
                        singleContextWeights[i][j] = 0;
                }
        }

        for(i=0; i<26; i++){
                for(j=0; j<26; j++){
                        for(k=0; k<26; k++){
                                doubleContextWeights[i][j][k] = 0;
                        }
                }
        }

//--------------------------- PART A------------------------------//
        cout << "\nPart (a) - Random 4 letter words\n" << endl;

        for(i=1; i<=100; i++){
                str = "";
                for(j=0; j<4; j++){
                        str += rand() % 26 + 97;
                }
                cout << str << " ";
                if(!(i%10)) cout << endl;
        }

//--------------------------- PART B ------------------------------//
        cout << "\n\nPart (b) - 4 letter words from probability model\n" << endl;

        // obtain probability model for the alphabet
        for(i=0; i<26; i++){
                probs[i] = 0.0;
                weights[i] = 0;
        }
```

```
// open file and populate 0,1,2 context probabilities
if(file.is_open()){
        while(getline (file, line) ){
                // send string to lower
                for(i=0; i<4; i++){
                        if(line.at(i) >= 65 && line.at(i) <= 90){
                                line.at(i) = line.at(i) + 32;
                        }
                }
                for(i=0; i<4; i++){
                        char c = line.at(i);
                        if(c >= 97 && c <= 122){
                                // no context
                                weights[line.at(i) - 97] ++;
                                // 1 letter context
                                if(i<3) singleContextWeights[line.at(i)-97]
                                                       [line.at(i+1)-97]++;
                                // 2 letter context
                                if(i<2) doubleContextWeights[line.at(i)-97]
                                      [line.at(i+1)-97][line.at(i+2)-97]++;
                        }
                }
                total += 4;
        }
}

file.close();
for(i=0; i<26; i++){
        probs[i] = (double) weights[i] / (double) total;
}

// create cdf for 0 context
double cdf[26];
cdf[0] = probs[0];
for(i=1; i<26; i++){
        cdf[i] = cdf[i-1] + probs[i];
}

for(i=1; i<=100; i++){
        str = "";
        for(j=0; j<4; j++){
                int k=0;
                double random = ((double) rand() / (RAND_MAX));
                while(random >= cdf[k]){
                        k++;
                }
                str += (char)(97 + k);
        }
        cout << str << " ";
        if(!(i%10)) cout << endl;
}
```

```
//--------------------------- PART C ------------------------------//
        cout << "\n\nPart (c) - single letter context\n" << endl;

        // Create cdf for each letter
        double singleContextCdf[26][26];
        for(i=0; i<26; i++){
                int colTotal = 0;
                for(j=0; j<26; j++){
                        colTotal += singleContextWeights[i][j];
                }
                singleContextCdf[i][0] = (double)singleContextWeights[i][0] / (
                                                           double) colTotal;
                for(j=1; j<26; j++){
                        singleContextCdf[i][j] = ((double)singleContextWeights[i]
                              [j] / (double) colTotal) + singleContextCdf[i][j-1];
                }
        }

        for(i=1; i<=100; i++){
                str = "";

                // get first letter from 0 context cdf
                int k=0;
                char firstLetterIndex;
                double random = ((double) rand() / (RAND_MAX));
                while(random >= cdf[k]){
                        k++;
                }
                firstLetterIndex = k;
                str += (char)(97 + k);

                // get letters 2,3,4
                char c = firstLetterIndex;
                for(j=1; j<4; j++){

                        double random = ((double) rand() / (RAND_MAX));
                        k=0;
                        while(random >= singleContextCdf[c][k]){
                                k++;
                        }
                        str += (char)(97+k);
                        c = k;
                }
                cout << str << " ";
                if(!(i%10)) cout << endl;
        }
```

```
//--------------------------- PART D ------------------------------//

        cout << "\nPart (d) - two-letter context\n" << endl;

        // create 2 letter context cdf
        double doubleContextCdf[26][26][26];
        for(i=0; i<26; i++){
                for(j=0; j<26; j++){
                        int colTotal = 0;

                        for(k=0; k<26; k++){
                                colTotal += doubleContextWeights[i][j][k];
                        }

                        if(colTotal == 0){
                                doubleContextCdf[i][j][0] = 0.0;

                        }
                        else{
                                doubleContextCdf[i][j][0] =
                         (double)doubleContextWeights[i][j][0] / (double) colTotal;
                                for(k=1; k<26; k++){
                                        double a = (double) doubleContextWeights[i]
                                                                            [j][k];
                                        double b = (double) colTotal;
                                        double c = doubleContextCdf[i][j][k-1];
                                        doubleContextCdf[i][j][k] = (a/b) + c;
                                }
                        }

                }

        }

        for(i=1; i<=100; i++){
                str = "";
                // get first letter from 0 context cdf
                k=0;
                char firstLetterIndex;
                double random = ((double) rand() / (RAND_MAX));

                while(random >= cdf[k]){
                        k++;
                }
                firstLetterIndex = k;
                str += (char)(97 + k);

                // get second letter from 1 context cdf
                random = ((double) rand() / (RAND_MAX));
                k=0;
                while(random >= singleContextCdf[firstLetterIndex][k]){
                        k++;
                }
                str += (char)(97+k);
                // get letters 3 and 4
                char a = str.at(0)-97;
                char b = str.at(1)-97;
                bool noContext = false;
```

```
                  for(j=2; j<4; j++){
                        double random = ((double) rand() / (RAND_MAX));
                        k=0;
                        while(random >= doubleContextCdf[a][b][k]){
                                k++;
                                if(k == 25){
                                        noContext = true;
                                        break;
                                }
                        }
                        if(noContext){
                                // going to use 0 context probabilty model
                                random = ((double) rand() / (RAND_MAX));
                                k=0;
                                while(random >= cdf[k]){
                                        k++;
                                }
                        }
                        str += (char)(97+k);
                        a = b;
                        b = k;
                }

                cout << str << " ";
                if(!(i%10)) cout << endl;
        }

}
```