Running authorship attribution via FCGR on the CCAT-10 data set

Initialization of the needed functions.

```
In[109]:= letterRules1 = {"p" | "d" → "b", "k" | "q" | "x" | "z" → "c", "h" | "j" → "g", "f" | "v" → "w",
             "y" → "i", "l" → "r", "\t" | "\n" | ";" | "," | "?" | "!" | ":" | "." | "(" | ")" | "-" |
               "+" | "[" | "]" | "(" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" → "0"};

          letterRules2 = {"g" → {0, 0}, "i" → {0, 1}, "t" → {0, 2}, "m" → {0, 3},
             "r" → {1, 0}, "a" → {1, 1}, " " → {1, 2}, "s" → {1, 3}, "e" → {2, 0}, "n" → {2, 1},
             "0" → {2, 2}, "u" → {2, 3}, "b" → {3, 0}, "w" → {3, 1}, "c" → {3, 2}, "o" → {3, 3}};

In[111]:= makePositionsC = Compile[{{shifts, _Integer, 2}, {k, _Integer}},
             Module[{posns},
              posns = FoldList[Mod[2 * #1 + #2, 2 ^ k] &, Reverse@shifts];
              Most[Reverse[Map[{2 ^ k, 1} + {-1, 1} * Reverse[#] &, posns]]]
            ], RuntimeOptions → "Speed", CompilationTarget → "C"];

          FCGR[chars_, k_] := Module[
            {posns, newposns},
            newposns = Round[makePositionsC[Cases[chars, {_Integer, _Integer}], k]];
            Normal[SparseArray[Apply[Rule, Tally[newposns], {1}], {2 ^ k, 2 ^ k}]]
           ]
```

```
In[115]:= processTrainInput[ilist_, keep_] :=
           Module[
            {idata = ilist, dcts, top,
              topvecs, uu, ww, vv, udotw, norms},
            topvecs = Map[Flatten, idata];
            {uu, ww, vv} =
              SingularValueDecomposition[topvecs, keep];
            udotw = uu.ww;
            norms = Map[Sqrt[#.#] &, udotw];
            udotw = udotw / norms;
            {udotw, vv}]

          processTestInput[ilist_, vv_] :=
           Module[
            {idata = ilist, dcts, top,
              topvecs, tdotv, norms},
            topvecs = Map[Flatten, idata];
            tdotv = topvecs.vv;
            norms = Map[Sqrt[#.#] &, tdotv];
            tdotv = tdotv / norms;
            tdotv]
```

Download the CCAT-10 corpus to home directory. From there the code is as below.

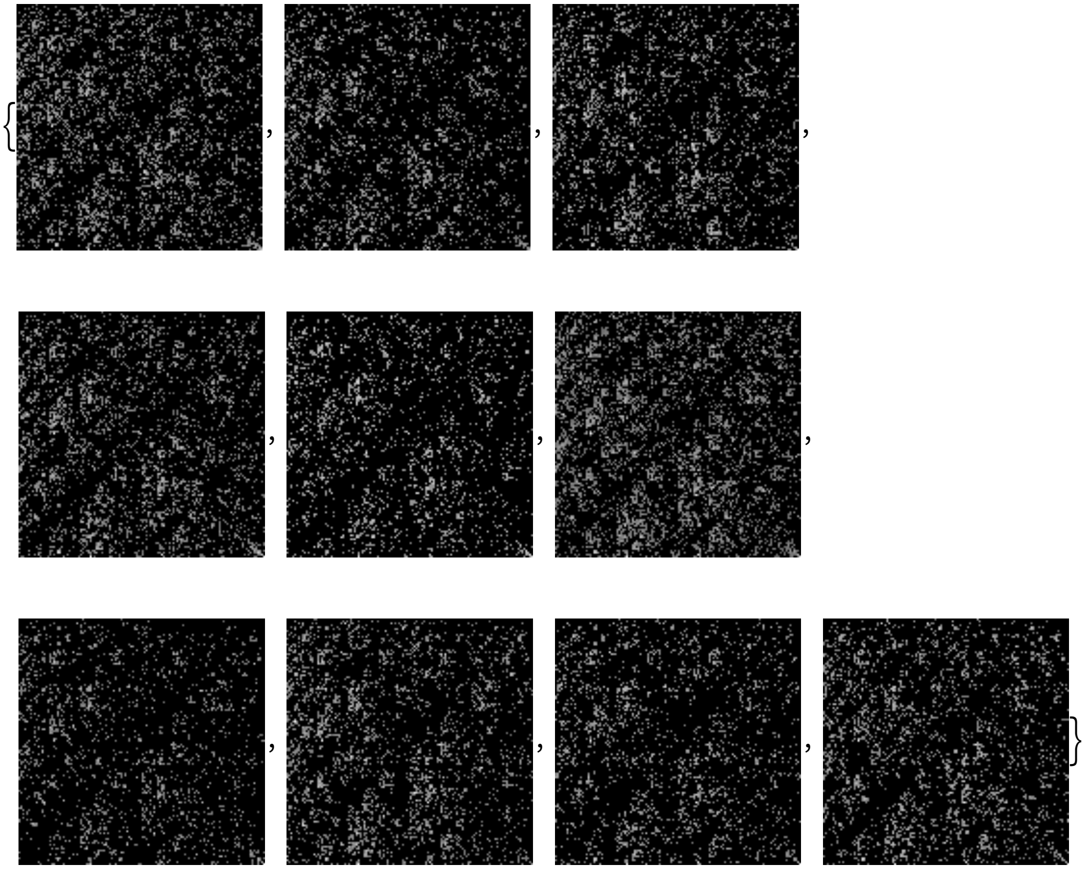
```
In[117]:= c10Train = FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10train", "*"}]];
          c10Test = FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10test", "*"}]];
          allC10TrainFiles =
            FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10train", "*", "*"}]];
          allC10TestFiles = FileNames[FileNameJoin[{$HomeDirectory, "c10", "C10test", "*", "*"}]];
          clip = 25;
          AbsoluteTiming[allTrainWritings =
              Map[StringTake[Import[#, "Text"], {clip, -clip}] &, allC10TrainFiles];]
          AbsoluteTiming[allTestWritings =
              Map[StringTake[Import[#, "Text"], {clip, -clip}] &, allC10TestFiles];]
          allTrainWritingsPartitioned = Partition[allTrainWritings, 50];
          allTestWritingsPartitioned = Partition[allTestWritings, 50];
          MaxMemoryUsed[]

Out[122]= {1.225298, Null}

Out[123]= {1.228019, Null}

Out[126]= 1 110 829 632
```

```
In[127]:= AbsoluteTiming[traintextLetters =
             Map[Characters[ToLowerCase[RemoveDiacritics[#]]] &, allTrainWritingsPartitioned, {2}];]
          AbsoluteTiming[traindigitseqs = Map[Developer`ToPackedArray[
                 (IntegerDigits[Flatten[# /. letterRules1 /. letterRules2], 2, 2] /.
                    IntegerDigits[__] :→ Nothing)] &, traintextLetters, {2}];]

          AbsoluteTiming[testtextLetters =
             Map[Characters[ToLowerCase[RemoveDiacritics[#]]] &, allTestWritingsPartitioned, {2}];]
          AbsoluteTiming[testdigitseqs = Map[Developer`ToPackedArray[
                 (IntegerDigits[Flatten[# /. letterRules1 /. letterRules2], 2, 2] /.
                    IntegerDigits[__] :→ Nothing)] &, testtextLetters, {2}];]

          trainauthors = Map[FileNameTake, c10Train];
          ltlen = Length[trainauthors];
          testauthors = Map[FileNameTake, c10Test];
          MaxMemoryUsed[]

Out[127]= {0.215752, Null}

Out[128]= {2.845503, Null}

Out[129]= {0.224612, Null}

Out[130]= {2.912574, Null}

Out[134]= 1 110 829 632

In[135]:= pixLevel = 7;
          AbsoluteTiming[trainimages1a = Table[
              FCGR[traindigitseqs[[j, k]], pixLevel], {j, ltlen}, {k, Length[traindigitseqs[[j]]]}];]
          AbsoluteTiming[testimages1a = Table[FCGR[testdigitseqs[[j, k]], pixLevel],
              {j, ltlen}, {k, Length[testdigitseqs[[j]]]}];]
          MaxMemoryUsed[]

Out[136]= {2.798862, Null}

Out[137]= {2.845943, Null}

Out[138]= 1 110 829 632
```

```
In[139]:= expon = 1 / 5;
          trainimages1 = Map[(# / N[Max[#]]) ^ expon &, trainimages1a, {2}];
          testimages1 = Map[(# / N[Max[#]]) ^ expon &, testimages1a, {2}];
          MaxMemoryUsed[]
          trainSetLabels =
            Flatten[Table[ConstantArray[trainauthors[[j]], Length[trainimages1[[j]]]], {j, ltlen}]];
          trainImages = Apply[Join, trainimages1];
          testSetLabels =
            Flatten[Table[ConstantArray[testauthors[[j]], Length[testimages1[[j]]]], {j, ltlen}]];
          testImages = Apply[Join, testimages1];
          Length[testImages]
          MaxMemoryUsed[]

Out[142]= 1 110 829 632

Out[147]= 500

Out[148]= 1 110 829 632
```

Here are images from each each author’s initial training text.

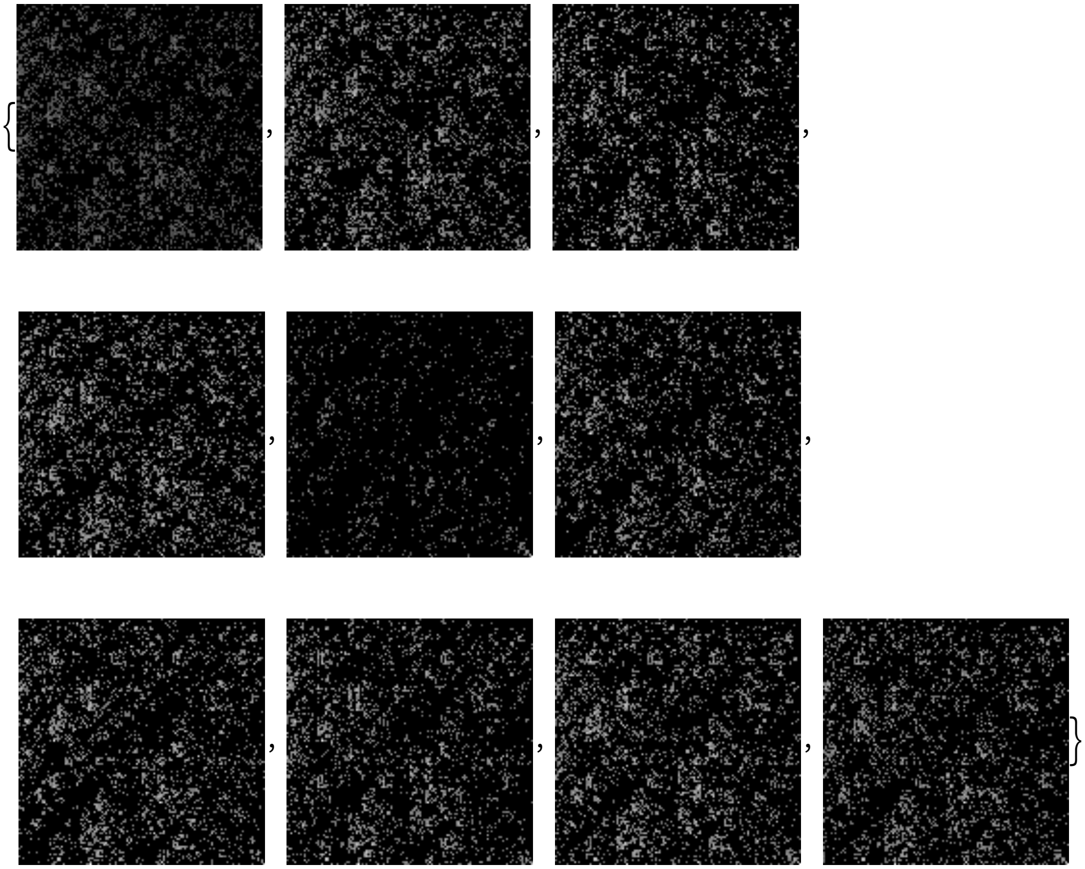
```
In[59]:= firstImages = Map[Image, trainImages[[1 ;; -1 ;; 50]]]

Out[59]=
```



Here are images from each each author’s final training text.

```
In[60]:= firstImages = Map[Image, trainImages[[50 ;; -1 ;; 50]]]
```

Out[60]=



Retain 500 largest singular values to reduce dimension of each FCGR image from 128^2 to 500.

```
In[149]:= keep = 500;
          {processedTrainImages, vvY} =
            processTrainInput[trainImages, keep];
          processedTestImages =
            processTestInput[testImages, vvY];
          MaxMemoryUsed[]
          trainData = processedTrainImages -> trainSetLabels;

Out[152]= 1 338 590 264
```

```
In[154]:= authors = Union[trainSetLabels];
          labelLen = Length[authors];
          replacements = Thread[authors → Range[labelLen]];
          newtrainSetLabels = trainSetLabels /. replacements;
          trainSet = Rule[processedTrainImages, newtrainSetLabels];
```

Create and train a neural net to recognize the dimension reduced vectors. We do this 10 times using some randomization, then sum probabilities on a per-testtext item over the 10 runs. The author with highest probability is deemd author of that text.

```
In[219]:= net = NetChain[{400, Tanh, labelLen, Tanh, SoftmaxLayer[]},
            "Input" → keep, "Output" → NetDecoder[{"Class", Range[labelLen]}]];
          subprobs = Table[
            Print[First[AbsoluteTiming[trained = NetTrain[net, RandomSample[Thread@trainSet],
                 MaxTrainingRounds → 300, RandomSeeding → Automatic,
                 LossFunction → CrossEntropyLossLayer["Index"], Method → {"ADAM", "Beta1" → .9}]]]];
            Print[First[AbsoluteTiming[results = Transpose[
                 {testSetLabels /. replacements, Map[trained, processedTestImages]}]]]];
            Print[{j, Length[Cases[results, {a_, a_}]],
              N[Length[Cases[results, {a_, a_}]] / Length[results]]}];
            {N[Length[Cases[results, {a_, a_}]] / Length[results]],
             Map[Normal[trained[#, None]] &, processedTestImages]}
            , {j, 1, 10}];
          probs = Total[Normal[subprobs[[All, 2]]]];
          maxposns = Map[First[Ordering[#, -1]] &, probs];
          results = Transpose[{testSetLabels /. replacements, maxposns}];
          tallied = Tally[results];
          nextposns = Map[First[Ordering[#, -2]] &, probs];
          nextresults = Transpose[{testSetLabels /. replacements, nextposns}];
          correct = Sort[Cases[tallied, {{a_, a_}, b_} :> {a, b}]];
          totals = Sort[Tally[maxposns]];
          wrongcounts = Normal[SparseArray[totals[[All, 1]] → totals[[All, 2]]], 50] -
             Normal[SparseArray[correct[[All, 1]] -> correct[[All, 2]]], 50];
          {Length[Cases[results, {a_, a_}]], N[Length[Cases[results, {a_, a_}]] / Length[results]],
           Length[Cases[nextresults, {a_, a_}]],
           N[Length[Cases[nextresults, {a_, a_}]] / Length[nextresults]], Mean[subprobs[[All, 1]]],
           Normal[SparseArray[correct[[All, 1]] -> correct[[All, 2]]], 50], wrongcounts}
          tallied
```

```
8.769335

0.05054

{1, 430, 0.86}

8.679846

0.05078

{2, 430, 0.86}

8.702146

0.049789

{3, 429, 0.858}

8.641024

0.068268

{4, 430, 0.86}

8.702668

0.054137

{5, 431, 0.862}

8.573347

0.052002

{6, 428, 0.856}

8.591369

0.050249

{7, 430, 0.86}

8.607939

0.051202

{8, 430, 0.86}

8.764854

0.052622

{9, 431, 0.862}

8.6696

0.052077

{10, 429, 0.858}

Out[229]= {430, 0.86, 52, 0.104, 0.8596,
           {35, 50, 38, 23, 50, 49, 49, 47, 50, 39}, {1, 3, 2, 8, 0, 15, 0, 13, 9, 19}}

Out[230]= {{{1, 1}, 35}, {{1, 6}, 15}, {{2, 2}, 50}, {{3, 3}, 38}, {{3, 8}, 12}, {{4, 10}, 19}, {{4, 4}, 23},
           {{4, 9}, 8}, {{5, 5}, 50}, {{6, 6}, 49}, {{6, 1}, 1}, {{7, 7}, 49}, {{7, 8}, 1}, {{8, 8}, 47},
           {{8, 3}, 2}, {{8, 2}, 1}, {{9, 9}, 50}, {{10, 10}, 39}, {{10, 4}, 8}, {{10, 9}, 1}, {{10, 2}, 2}}
```

Repeat to double check.

```
In[207]:= net = NetChain[{400, Tanh, labelLen, Tanh, SoftmaxLayer[]},
            "Input" → keep, "Output" → NetDecoder[{"Class", Range[labelLen]}]];
          subprobs = Table[
            Print[First[AbsoluteTiming[trained = NetTrain[net, RandomSample[Thread@trainSet],
                  MaxTrainingRounds → 300, RandomSeeding → Automatic,
                  LossFunction → CrossEntropyLossLayer["Index"], Method → {"ADAM", "Beta1" → .9}]]]];
            Print[First[AbsoluteTiming[results = Transpose[
                  {testSetLabels /. replacements, Map[trained, processedTestImages]}]]]];
            Print[{j, Length[Cases[results, {a_, a_}]],
              N[Length[Cases[results, {a_, a_}]] / Length[results]]}];
            {N[Length[Cases[results, {a_, a_}]] / Length[results]],
             Map[Normal[trained[#, None]] &, processedTestImages]}
            , {j, 1, 10}];
          probs = Total[Normal[subprobs[[All, 2]]]];
          maxposns = Map[First[Ordering[#, -1]] &, probs];
          results = Transpose[{testSetLabels /. replacements, maxposns}];
          tallied = Tally[results];
          nextposns = Map[First[Ordering[#, -2]] &, probs];
          nextresults = Transpose[{testSetLabels /. replacements, nextposns}];
          correct = Sort[Cases[tallied, {{a_, a_}, b_} :> {a, b}]];
          totals = Sort[Tally[maxposns]];
          wrongcounts = Normal[SparseArray[totals[[All, 1]] → totals[[All, 2]]], 50] -
             Normal[SparseArray[correct[[All, 1]] -> correct[[All, 2]]], 50];
          {Length[Cases[results, {a_, a_}]], N[Length[Cases[results, {a_, a_}]] / Length[results]],
           Length[Cases[nextresults, {a_, a_}]],
           N[Length[Cases[nextresults, {a_, a_}]] / Length[nextresults]], Mean[subprobs[[All, 1]]],
           Normal[SparseArray[correct[[All, 1]] -> correct[[All, 2]]], 50], wrongcounts}
          tallied
          8.723461

          0.060977

          {1, 433, 0.866}

          8.628176

          0.051208

          {2, 431, 0.862}

          8.58775

          0.051275

          {3, 432, 0.864}

          8.574647

          0.052109

          {4, 429, 0.858}

          8.58878

          0.050387

          {5, 431, 0.862}

          8.624452

          0.049145

          {6, 430, 0.86}

          8.582698

          0.050673

          {7, 431, 0.862}

          8.624664

          0.051035

          {8, 431, 0.862}

          8.61392

          0.051863

          {9, 431, 0.862}

          8.659046

          0.054873

          {10, 431, 0.862}

Out[217]= {431, 0.862, 51, 0.102, 0.862,
           {35, 50, 39, 23, 50, 49, 49, 47, 50, 39}, {1, 3, 2, 8, 0, 15, 0, 12, 9, 19}}

Out[218]= {{{1, 1}, 35}, {{1, 6}, 15}, {{2, 2}, 50}, {{3, 3}, 39}, {{3, 8}, 11}, {{4, 10}, 19}, {{4, 4}, 23},
           {{4, 9}, 8}, {{5, 5}, 50}, {{6, 6}, 49}, {{6, 1}, 1}, {{7, 7}, 49}, {{7, 8}, 1}, {{8, 8}, 47},
           {{8, 3}, 2}, {{8, 2}, 1}, {{9, 9}, 50}, {{10, 10}, 39}, {{10, 4}, 8}, {{10, 9}, 1}, {{10, 2}, 2}}
```