R Notebook

First, we import our necessary libraries.

```
library(corpora)
library(HMM)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

The main library we care about is from corpora and called "BrownBigrams." It is a data frame with 24167 rows and the following columns:

id: unique ID of the bigram entry

word1: the first word form in the bigram (character)

pos1: part-of-speech category of the first word (factor)

word2: the second word form in the bigram (character)

pos2: part-of-speech category of the second word (factor)

O11: co-occurrence frequency of the bigram (numeric)

O12: occurrences of the first word without the second (numeric)

O21: occurrences of the second word without the first (numeric)

O22: number of bigram tokens containing neither the first nor the second word (numeric)

It looks like this:

```
head(BrownBigrams,5)
```

```
##   id word1 pos1 word2 pos2 O11 O12   O21    O22
## 1  1     %    N     .    .   8  67 45465 864228
## 2  2     %    N    in    I   6  69 17981 891712
## 3  3     %    N    of    I  18  57 35481 874212
## 4  4     %    S     .    .   5  49 45468 864246
## 5  5     %    S    of    I  15  39 35484 874230
```

Our states are the possible parts of speech tags. We construct the list of states like this:

```
states = c("C", "D", "E", "F", "G", "I", "J", "L", "M", "N", "P", "R", "S", "T", "U", "V", "W", "Y", ".
states
```

```
##  [1] "C" "D" "E" "F" "G" "I" "J" "L" "M" "N" "P" "R" "S" "T" "U" "V" "W"
## [18] "Y" "."
```

Our Symbols are the words. We construct the list of symbols like this:

```
symbols = unique(BrownBigrams$word1)
head(symbols, 5)
```

```
## [1] "%"   "&"   "1.1" "1"   "10"
```

Next, we make our transition matrix. We start this by counting all of the different length-2 tag sequences that occur throughout our dataset. The following matrix has these counts where entry $X_{i,j}$

```
transition_matrix = matrix(0L, nrow = length(states), ncol = length(states), dimnames = list(states, sta

for(i in 1:24167){
  transition_matrix[BrownBigrams$pos1[i], BrownBigrams$pos2[i]] = transition_matrix[BrownBigrams$pos1[i]
}
transition_matrix
```

```
##      C   D   E  F G   I    J   L  M    N   P   R  S   T U   V  W   Y .
## C   43  67  47  3 0  28  103 136 19  322  30 111  4  17 0 198 24  71 0
## D   10  42  15  1 3  54   37 961 14 2850   9  58  2   5 0  41  6 368 0
## E    0   0   0  0 0   0    0   0  9    0   0   1  0   0 0   9  0   0 0
## F    1   0   0  0 6   0    0   0  0    0   0   0  0   0 0   0  0   0 0
## G   88  15 158  0 0  58  531   7  0   21  28  43  0 174 0   2  2   0 0
## I   22 163 345 11 0 124  136 328  2  808 263  97 15  21 0  13 48 296 0
## J  108 135  16  0 0   0  176  28  0  628   4   7  0  92 0   5  1   3 0
## L    7   0   0  0 0   0    0   0  0    0   0   0  0   0 0   0  0   0 0
## M    4   3   9  0 0   0    1   0  0    0  21  61  0   1 0 225  0   0 0
## N 1299 587  30  1 0  13 2143  14 36  311  48  43  7 284 0 435 95   5 0
## P   30  21  46  0 0   4  125   7 79    2  14 133  0  13 0 655 15   0 0
## R  118  50 125  3 0  54  299 149  6    5  62 174  2  56 0 175 17   0 0
## S    2   2   0  0 0   0    8   0  2   10   0   0  0   1 0   4  2   0 0
## T    2  21  16  0 0   1    6  26  0   38  13   5  1   0 0 518  5  33 0
## U    4   0   0  0 0   0    0   0  0    0   1   0  0   0 1   0  0   0 0
## V  111  34 667  4 0 722  599 198  0   55 296 430  5 205 0  25 43   7 0
## W    5   4  33  3 0   0   11   6 39   12  53  17  2   5 0 133  0   4 0
## Y  130  92   2  0 0   0   74   1  5   47   1   0  0   4 0  72  1 419 0
## .    0   0   0  0 0   0    0   0  0    0   0   0  0   0 0   0  0   0 0
```

Now that we have our counts, we divide each number in row i by the total count of dimension i.

```
tag_count = BrownBigrams %>% group_by(pos1) %>% tally()

for(i in 1:18){
 for(j in 1:19){
  transition_matrix[states[i],states[j]] = transition_matrix[states[i],states[j]] / tag_count$n[i]
  }
}
head(transition_matrix, 5)
```

```
##             C           D           E           F           G           I
## C 0.035159444 0.054783320 0.038430090 0.0024529845 0.0000000000 0.02289452
## D 0.002234138 0.009383378 0.003351206 0.0002234138 0.0006702413 0.01206434
## E 0.000000000 0.000000000 0.000000000 0.0000000000 0.0000000000 0.00000000
## F 0.142857143 0.000000000 0.000000000 0.0000000000 0.8571428571 0.00000000
## G 0.078083407 0.013309672 0.140195209 0.0000000000 0.0000000000 0.05146406
##             J           L           M           N           P           R
## C 0.084219133 0.11120196 0.015535568 0.26328700 0.024529845 0.09076043
## D 0.008266309 0.21470063 0.003127793 0.63672922 0.002010724 0.01295800
```

```
## E 0.000000000 0.00000000 0.473684211 0.00000000 0.000000000 0.05263158
## F 0.000000000 0.00000000 0.000000000 0.00000000 0.000000000 0.00000000
## G 0.471162378 0.00621118 0.000000000 0.01863354 0.024844720 0.03815439
##              S           T U           V           W          Y .
## C 0.0032706460 0.013900245 0 0.161896975 0.019623876 0.05805397 0
## D 0.0004468275 0.001117069 0 0.009159964 0.001340483 0.08221626 0
## E 0.0000000000 0.000000000 0 0.473684211 0.000000000 0.00000000 0
## F 0.0000000000 0.000000000 0 0.000000000 0.000000000 0.00000000 0
## G 0.0000000000 0.154392192 0 0.001774623 0.001774623 0.00000000 0
```

Finally, we make our emmission probabilities using the same process that we did for our transimissions.

```
emissions = matrix(0L, nrow = length(states), ncol = length(symbols), dimnames = list(states, symbols))

for(i in 1:24167){
  emissions[BrownBigrams$pos1[i], BrownBigrams$word1[i]] = emissions[BrownBigrams$pos1[i], BrownBigrams$
}

symbol_count = BrownBigrams %>% group_by(word1) %>% tally()
for(a in 1:length(states)){
 for(b in 1:length(symbols)){
  emissions[states[a],symbols[b]] = emissions[states[a],symbols[b]] / symbol_count$n[b]
  }
}
```

(We will not print out this matrix because even the first row is far too long).

Now, we can make our HMM and run the viterbi algorithm with some simple r commands!

```
model = initHMM(States=states, Symbols=symbols, startProbs=NULL, transProbs=transition_matrix, emissionM
viterbi(model, c("how", "are", "you"))
```

```
## [1] "W" "V" "P"
```

```
viterbi(model, c("i", "like", "this", "class"))
```

```
## [1] "P" "V" "D" "N"
```