Report of Project 4

(J1799d Fall 2016)

ID	16213677	Name	Zijie Chai
Phone	15626134697	Email	418779654@qq.com
ID	16213697	Name	Fan Wu
Phone	13051022052	Email	fanw1@andrew.cmu.edu
Started date	2016/10/20	Completed date	2016/11/17

1. Project Requirement

Problem 1:

Build a continuous-speech recognition system that can recognize telephone numbers. To do so, connect the digit models you have learned via segmental K-means into the following larger HMM. Note that we allow the speaker to say telephone digits numbers either as a 7-digit number or a 4-digit number. Furthermore, we are taking advantage of the fact that the first digit in a telephone number is never 0 or 1.

Record 25 valid telephone numbers randomly and report recognition accuracy. Accuracy is reported in terms of:

- Sentence accuracy: How many telephone numbers were recognized correctly.
- Word accuracy: How many digits were recognized correctly.

Do the above with and without pruning and report speed.

Problem 2:

Do the above using a backpointer table. Report accuracies as described above.

Problem 3:

Build a system to recognize unrestricted digit strings.

In this case it is useful to assign an "insertion penalty" to the loopback to ensure that large numbers of digits are not hypothesized randomly. Determine the optimal insertion penalty empirically (by evaluating recognition error as given below) and report both the optimal insertion penalty and the recognition performance as required below.

Record the following 10 digit strings in your voice and evaluate accuracy on them. Accuracy is to be measured in terms of:

- Sentence accuracy as before.
- Word error rate. For this perform DTW alignment between the recognition output and the recognized string. The number of errors is the edit distance between the true text and the recognized output. Divide the total number of word errors for all test data and divide by the number of words in the actual (true) strings to obtain error rate.

Problem 4:

In this problem, we will train HMMs for digits from continuous speech recordings.

Record each digit sequence as a continuous recording (without pauses between words). You will

now have 30 recordings of each of the digits. Train models for all ten digits from these continuous recordings. To do so, compose the model for each digit string by concatenating the models for the individual digits.
Include silence models on either side, e.g. to model "0 1 2 3 4 5 6 7 8 9", compose the HMM as "sil * 0 * 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * sil" (The "*" here indicates conctenation, and the digits represent their HMMs).
Initialize the HMMs for all digits by the models you learned for them from isolated recordings in previous assignments. Report accuracies as reported in previous problems.
Problem 5:
This problem aims to train models from a medium sized corpus of recordings of digit sequences. The recordings are continuous digit strings. You will use these data to train models for all digits. The transcriptions also have the "silence" marked, so you need not explicity add silences at the end of the digit strings.
The "test" directory contains 1000 test recordings. Each recording is a digit sequence. Report the recognition accuracy as you did for the previous problem.

2. My Program

Problem 1:

a. Data structure

(1) DTW column

Declared like this:

```
double oddColumn7[7][55], evenColumn7[7][55];
```

To update DTW matrix, we use two columns to synchronously update the value of every node of every digit, and the two column have the structure as below.

Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7

And for every digit there will be ten templates, for every template there are 5 states, and the inner structure of every digit is shown below.

Templ ate 0	Templ ate 1	Templ ate 2	Templ ate 3	Templ ate 4	Templ ate 5	Templ ate 6	Templ ate 7	Templ ate 8	Templ ate 9

(2) Backtrack Table

Declared like this:

```
vector<vector<vector<int > > > backtrackPtr;
```

It is a three dimension array, the first dimension is the frame number, the second dimension is the digit count, and the third dimension is the state count. For every digit there will be a path to backtrack.

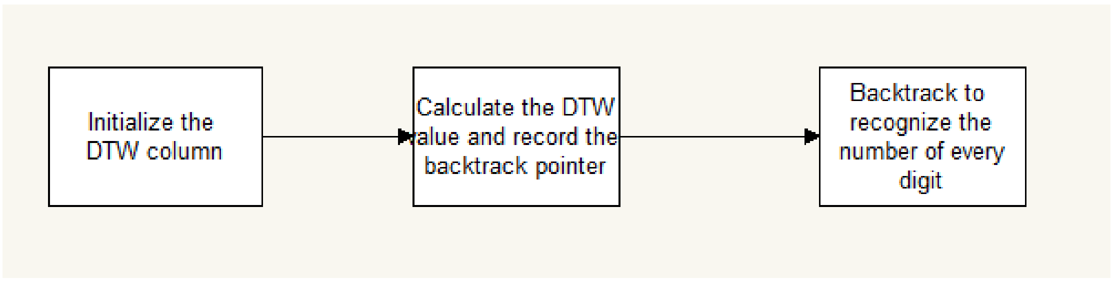
(3) Backward array

Declared like this:

```
vector<vector<int > > backwardPtr;
```

For every digit except for the first digit, the path of the first state of every template can from the previous node or can from the end of the last digit. If the first state source is from the end of last digit, the which template in the last digit is used as source will be store in this array corresponding to input frame.

b. Program flow chart


Initialize the DTW column
Calculate the DTW value and record the backtrack pointer
Backtrack to recognize the number of every digit

(1) Initialize the DTW column:

In this step, because the number of digit can be 7 and 4, we will update the every node of every state to INF, except for the first state in the every template of the first number and the third number. And for the first digit, the first two templates will be set to INF, because the first can neither be 1 nor 0.

(2) Calculate the DTW value and record the backtrack point:

In this step, we use two DTW columns to iteratively update the node and record the backtrack table. The thing that should be noticed is that, when updating the node distance of the first two templates, it should be ignored, because the first digit cannot be 0 or 1.

(3) Backtrack to recognize the number of every digit:

In this step, start from the node with the smallest cost in the last digit, then use the pointer in the backtrackTable to find the best way to the start. When it comes to the first state of one template, we will check if this node is come from the last node or come from the last digit, if it comes from last digit, then it means here is the start of one word. Then we can separate every word in the sentence.

Problem 2:

a. Data structure

(1) Two arrays used to update, each element in the array will record the source frame from which the current distance is calculated.

```
int oddBackPointer[7][55], evenBackPointer[7][55];
```

(2) When encounter a final state of one template, there will be a flag, one array to record the flag. The first dimension is the frame number, the second dimension is the digit number, and the third dimension is the flag of the frame.

```
vector<vector<vector<int > > > startFrame;
```

(3) When encounter the first state, we will choose the path between horizon and backward, use one array to record the path. The array will record the source of every first state. When backtrack, we will use this array to determine the backpointer.

```
vector<vector<int > > backwardPtr;
```

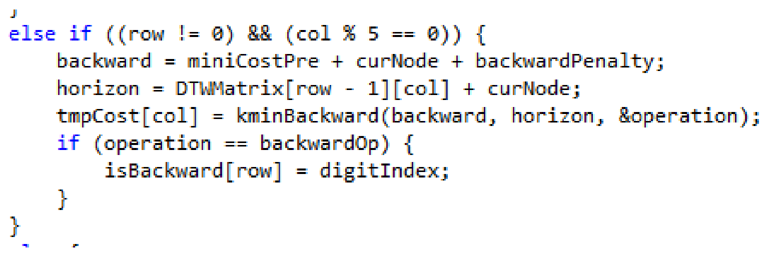
b. Program flow

```
For every frame:
    For every digit:
        For every template:
            If(is the first state) {
                If(is not backward){
```

```
                The source does not change;
            } else {
                The source is the state from which current state backward from
            }
        }
        If(is the last state) {
            Record the source frame where the last state comes and construct a flag
        }
    End
  End
End
```

Problem 3:

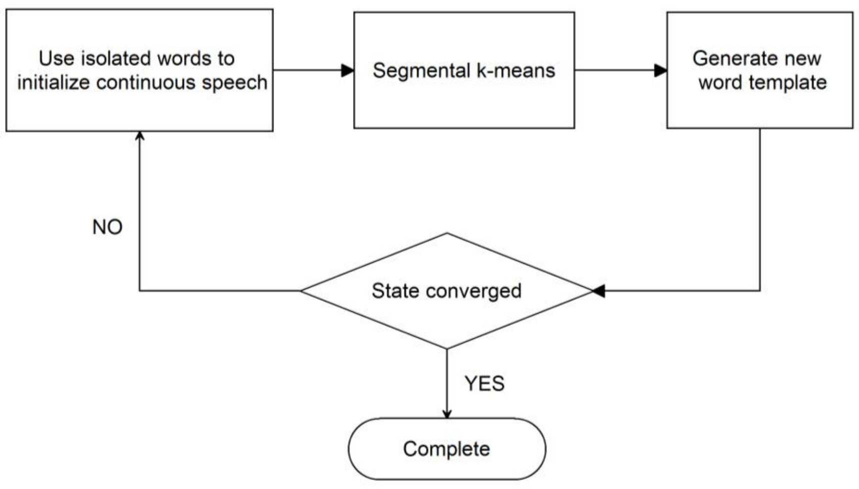
The recognition process is similar with problem 1. But because we do not know how many words are in the sentence, so we have to add the backward penalty to help recognize the gap between every word. So when comes to the first state of one digit, we should determine the distance of this node is from the horizon direction or the backward direction. The way to realize is shown as below:



```
else if ((row != 0) && (col % 5 == 0)) {
    backward = miniCostPre + curNode + backwardPenalty;
    horizon = DTWMatrix[row - 1][col] + curNode;
    tmpCost[col] = kminBackward(backward, horizon, &operation);
    if (operation == backwardOp) {
        isBackward[row] = digitIndex;
    }
}
```

Problem 4:

In this problem, we first use the pre-trained isolated words as template, to initialize every frame to each state. Then use all the words in every state to calculate means and covariances of each words, then by using segmental k-means to get new word templates. Then use the new templates to initialize the continuous speech and do the segmental k-means. Repeat the step until converge. The flow chart is shown below.


Use isolated words to initialize continuous speech
Segmental k-means
Generate new word template
NO
State converged
YES
Complete

Problem 5:

Use the same method as problem 4. The final process is shown below:

(1) Train the isolated word templates by using the isolated word in the training data.

(2) Use the isolated word as templates to initialize all speech frames in the training data to different states in different numbers by DTW.

(3) Use the initialized states to do segmental k-means to get the converged states. And get the new template of every new word.

(4) If iteration converges and we get the templates. If not, restart from step 1.

3. Program Running and Testing

Problem 1:

The right phone number	The recognized number
4444	4444
3456789	3456789
2345678	2345678
2212	2212
4679	4671
5678	5678
3567982	3567932
4032789	4032781
8346578	3343578
4320666	4320666
2579368	2579368
3579	3579
2468	2468
8642	8642
9753	9753
0034	0034
7364280	7364280
5470846	5470846
2333	2333
4555	4555
9999	9991
5111771	5111771
5678901	5678901
1356732	9353732
1111	1111

Word accuracy:

94.1%

Sentence accuracy:

76%

Problem 3:

The right phone number	The recognized phone number
9 1 1 3 8 5	111381
8 2 6 4 1 4 0 5 2 0 0 2	826414012000
8 2 1 2 1 7 6 3 4 2	8212176342
7 3 4 3 3 3 2 1 9 0 3 7 7	7313332110377
2 2 1 2	2212
1 2 3 4 5 6	123156
6 8 9 0 3 7 2 3 4 4	6810373347
7 2 1 8 4 3 4 7 9 2 4	72184337134
5 5 5 5 5	55555
3 7 2 7 4 9 2 1	37374131

Word accuracy: 83.5%

Sentence accuracy: 30%

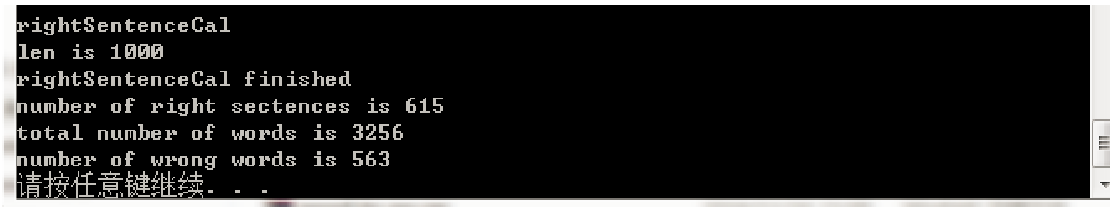
Problem 4:

The right phone number	The recognized phone number
9 1 1 3 8 5(6)	911385
8 2 6 4 1 4 0 5 2 0 0 2(12)	826414052002
8 2 1 2 1 7 6 3 4 2(10)	8212176342
7 3 4 3 3 3 2 1 9 0 3 7 7(13)	7343332190377
2 2 1 2(4)	2212
1 2 3 4 5 6(6)	123456
6 8 9 0 3 7 2 3 4 4(10)	66890372344
7 2 1 8 4 3 4 7 9 2 4(11)	72184347924
5 5 5 5 5(5)	655555
3 7 2 7 4 9 2 1(5)	37274921

Word accuracy: 97.6%

Sentence accuracy: 80%

Problem 5:



```
rightSentenceCal
len is 1000
rightSentenceCal finished
number of right sectences is 615
total number of words is 3256
number of wrong words is 563
请按任意键继续. . .
```

Word accuracy: 83%

Sentence accuracy: 61.5%

4. Project Conclusion

In this project, we learned to recognize continuous speech. In problem 1, we use isolated words as templates, to recognize fixed number of digit. That means, in one sentence, there will be 4 or 7 digits. Then we use isolated words as templates, to recognize continuous speech with multiple digits, whose result is not very good. However, for problem 4, we use continuous speech to train templates and get the HMM for every word. By using these words as templates, we get a relatively good result for recognizing continuous speech with multiple digits. At last, we use the same method, to train 8000 sentences and get the HMM for every number. By using these numbers as templates, we test 1000 speech and get a relatively good result.

5. Appendix

./Problem1,2,3
/audio/ FixedDigitRecognition.cpp
/audio/ FixedDigitRecognition.h
(for problem 1&2)

/audio/continuousRecogonition.cpp
/audio/continuousRecogonition.h

/Problem4,5(Project7)-Training (This project is used to train continuous speech to isolated word)
/match/ktrain.cpp
/match/ktrain.match

/Problem5(Projext7-2)-Testing(This project use the words trained from the last project to recognize continuous speech)
/Conti-speech/continuousRecogonitionloop.cpp(In this .cpp file, we can test 1000 speech at one time, and calculate the accuracy)
/Conti-speech/continuousRecogonitionloop.h