Project 5&6

(2015 Fall)

Course name: J1799d Instructor: Ming Li TA: Wenbo Liu

ID number	**15213796 / 15213782**	Name	**Luting Wang / Dajian Li**
Tel	**15622777988 / 13570300549**	Email	**364706267@qq.com / dajianl@andrew.cmu.edu**
Starting date	**2015.11.8**	Finished date	**2015.12.3**

1、 Project requirement

In this project, we are required to write a continuous speech recognition system to recognize telephone numbers. It includes a) continuous speech recognition using Hidden Markov Model (HMM) which are trained on isolated word data; b) continuous speech recognition using HMM models which are trained on small scale of continuous speech data. For the first one, since we have trained GMM model for digit 0-9 in previous assignments, we can just connect these digit models together to build a Finite State Gramma (FSG); for the second one, we use these models as initialization and train models for ten digits from continuous recording.

2、 Design your program

In project 5, there are two different FSG, as shown in Figure 1 and 2. For the first FSG, the length of the digit is fixed, it can only recognize 4 or 7 digits; while for the second FSG, the length of the digit is arbitrary, it can recognize any digit sequence, but to prevent larger number of digits, we should set optimal insertion penalty empirically.

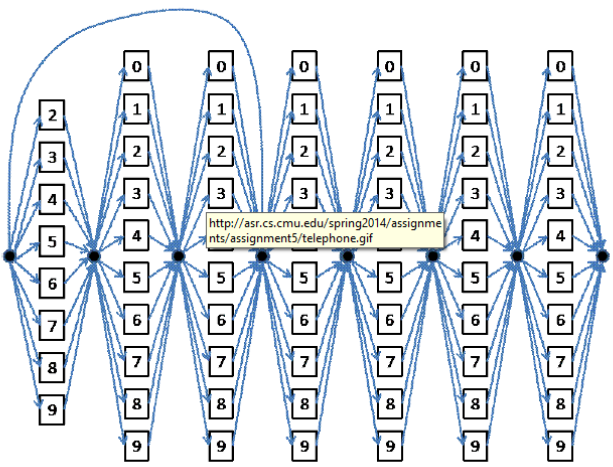

Figure 1

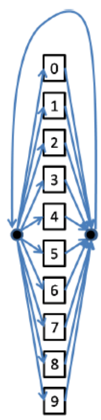

Figure 2

In project 6, the FSG is similar as Figure 1. To initialize HMM, we use the models we learned from isolated digit recordings. To train HMM models from continuous digit recordings, we concatenate the models together in specific order and do DTW alignment. Then with six different digit sequence (each with 5 recordings), we have 30 recordings of each of the digits. Then we train digit models and recognize with the continuous recordings for project 5 using these models.

3、 Program implementation and testing.

There are some important data structures and functions we implement, for details, please refer to the source code.

vector<HMMModel*> models; // the container for trained HMM models
vector<vector<TreeMap> > tree_Map // a map for language model non-emitting state
vector<vector<Trellis> > trellis; // search trellis
vector<int> lm_state_index; // index for each HMM model

void creat_Map(); // create a map for each non-emitting state, including digit, index and valid.
void search(vector<vector<float> > mfcc39) //applying DTW to search

4、 Experimental results and discussion

Project 5:

Table 1 and 2 is the result for project 5, problem 1 (red means matched). We record 25 valid telephone numbers randomly and write down the recognition accuracy including sentence accuracy and word accuracy, and compare the run-time speed with pruning and without pruning. Since we can only know the correct path until we enter the last state, so we can't actually prune a digit before that. So we apply pruning in each line of the search trellis to reduce computation time, but the speed increase is not very obvious. For 4 digits recognition, with pruning the computation time is 14s, and without pruning the time is 16s, about 2s speedup; while for 7 digits recognition, with pruning the computation time is 27s, and without pruning the time is 30s, about 3s speedup

Table 1 The result for project5, problem1

Input digit sequences	Recognition of digit sequences	with end-penalty (100)
2212	2212	2212
2344	2883484	2344
2345	7345	7345
4348	4348	4348
5555	5555	5555
6438	8643838	6438
9037	9007	9007
3770	7770	7770
8543	8703	8703
7676	7676	7676
3890	3890	3890
4235	4205	4205
0123	0103	0103
2345678	2045378	2045378

3727492	7727095	7727095
7343332	7343005	7343005
4572314	4857014	4857014
9022345	7054045	1005
6126690	6170690	6170690
8888999	8777999	8777999
5275561	5575571	5575571
9876543	8878543	8878543
8000123	8000103	8000103
2626111	7670111	7670111
7672212	7075217	7075217

Table 2 The accuracy statistics for project5, problem1

Item	Quantity
Total sequences	25
Total digits	136
Sentence match	7
Digits match	95
Sentence accuracy	28%
Word accuracy	70%

And for problem 2, after using a back-pointer table, the accuracy is the same as the previous one, as shown in Table 2.

For problem 3 to recognize unrestricted digit strings, we record 10 digit strings to evaluate recognition error as shown in Table 3. To ensure that large numbers of digits are not hypothesized randomly, we set an "insertion penalty" to the loopback as 150 and report their accuracy including sentence accuracy and edit distance.

Table 3

Table 3 The result for project5, problem3

Input	Recognition	Edit Distance	Deletion	Insertion	Substitution
911385					
826414052002					
8212176342					
7343332190377					
2212					
123456					
6890372344					
72184347924					
55555					
37274921					

Table 4 The accuracy statistics for project5, problem3

Item	Quantity
Total sequences	10

Sentence match	
Total digits	
Digits match	
Total Deletion	
Total Insertion	
Total Substitution	
Sentence accuracy	
Word accuracy	

Project 6:

Table 5 and 6 is the result for project 6, problem 1. We use the digit sequences recorded in project5, problem 3 to recognize and report their accuracy as we do in project5. As you can see, since we use continuous recordings to train digit models, the accuracy has been improved a lot.

Table 5 The result for project6, problem1

Input	Recognition	Edit Distance	Deletion	Insertion	Substitution
911385					
826414052002					
8212176342					
7343332190377					
2212					
123456					
6890372344					
72184347924					
55555					
37274921					

Table 6 The accuracy statistics for project6, problem1

Item	Quantity
Total sequences	10
Sentence match	
Total digits	
Digits match	
Total Deletion	
Total Insertion	
Total Substitution	
Sentence accuracy	
Word accuracy	

For problem 2, we train models from a medium sized corpus (8400 recordings) of recordings of digit sequences and use 1000 test recordings to recognize. Their accuracy is shown below.

Table 7 The accuracy statistics for project6, problem2

Item	Quantity
Total sequences	1000
Sentence match	
Total digits	
Digits match	
Total Deletion	
Total Insertion	
Total Substitution	
Sentence accuracy	
Word accuracy	