EE627 – Term Project : Jul. 2013 Semester

August 12, 2013

Title : Build and demonstrate a real time continuous digit recognition system in English

Assigned to : Batch No. 9

TAs Assigned : Lalan. He is Available @ACES 203 MiPS Lab EE Department. email : lalank@iitk.ac.in

Objective :

The objective of this project is to build a continous digit recognition system. A real time digit recognition needs to be built using the MONC database. Experimental results for speech recognition in terms of word error rate (WER) also need to be provided.

Methodology To Be Followed :

The methodlolgy need to be followed for performing the recogniton experiments are as follows:

- The module 'Print_feature' writes features of each single sound file. This is useful to avoid recomputing features in the embedded training procedure.
- The module 'endpoint_feature' does the same as Print_feature but eliminates silences.
- The module 'Print_phon_feature' writes features of the required files where all the same phonemes of all the files are collected in one file, i.e. one output feature file for each phoneme. This is required for non-embedded training.
- The module 'initiali initializes' the HMM models. HMM model parameters are evaluated according to a clustering procedure.

- The module 'training' re-estimates HMM models phoneme per phoneme using the Baum–Welch algorithm. The bounds of each phoneme within the utterances are required, i.e. segmentation of all the training speech data.
- The module 'embedded' re-estimates HMM models per utterance using the Baum–Welch algorithm. Segmentation is not required.
- The module 'lessico' estimates language model parameters according to various algorithms.
- The module 'recog ' performs phoneme/word recognition.
- The module 'eval_rec' evaluates accuracy of word/phoneme recognition.

Database for Training

The database used for training or testing is MONC database. The MONC is derived from the Numbers Corpus release 1.0, prepared by the Center for Spoken Language Understanding at the Oregon Graduate Institute. The general strategy used to acquire the MONC was to playback utterances from the original Numbers corpus on one or more loudspeakers, and record the resulting sound field with lapel microphones, a single tabletop microphone, and a tabletop microphone array. The MONC database is available at MiPS Lab. Please contact the TA assigned for the same.

Tools To Be Used

The Tools to be used to implement this term project is the C++. The details of download, installation, and usage are available at the respective TA.

Deliverables/ Submissions

The deliverables or submission procedures for the term project are as follows:

- Presentations and Report : Two set of presentations is required for every batch in this term project. The first presentation will be scheduled before mid sem and the second presentation will be scheduled before end sem. The marks will be distributed separately for two presentation. Additionally a report needs to be submitted detailing the project.
- System Demo in real time : The demonstration of the project is needed to be carried out by each batch. The demonstration includes the real time presentation of the working model for recognition system.
- Code/script submissions : The code or script has to be submitted by each batch which will include complete details of the project.

Other Useful Links

The other useful links that might be helpful in preparation of the code or script are as follows

- C++ Code : available at respective TAs

Recognition Experimental System

1 INTRODUCTION

This CD contains the Recognition Experimental System (RES) software version 6.0 19/11/98. The software is distributed only with the book*: Speech Recognition: Theory and C++ Implementation*, written by Claudio Becchetti and Lucio Prina Ricotti, published by John Wiley & Sons, and contains the whole C++ code of the phoneme/word multi-speaker continuous speech recognition system (Initialization Training Recognition Evaluation). It can be also adapted to other tasks (for example, prediction of time series such as stock exchange movements). To install the software, please remember to read section 2.2 carefully.

2 CD ORGANIZATION

In the CD, there are four main directories: Source, Test_Me, Projects, and Sndfile. Their contents are:

- Source: contains C++ source code files organized into directories corresponding to projects (the same applies to Linux MS-DOS and Windows).
- Test_Me: contains two ready-to-use examples for MS-DOS/Windows and Linux. The first directory ("Phoneme") contains an example of phoneme recognition on TIMIT. Run "Start_Me.bat" for MS-DOS/Windows platforms and "Start_Me.lnx" for Linux.

The second directory “Word_Rec” concerns word recognition on ATIS. As in the previous case, run “ Start_Me.bat” for MS-DOS/Windows platforms and “ Start_Me.lnx ” for Linux.

- Projects: This directory contains the projects and make-files. It has three subdirectories: “projectMS”, “projectDjGpp” and “Proj_Lnx” each containing the projects of the programs for MS Visual 5.0, RHIDE and Dj Gpp on PC MS-DOS/Windows and RHIDE and Gpp on Linux. RHIDE is a graphic integrated development environment; see section 3 for further details. Make files are also included in the homonymous directories.
- Sndfile: This directory contains only some ATIS and TIMIT sound and label files that belong to the LCD Consortium. These files have been adapted to the Microsoft MS RIFF standard (i.e. that used by the common *.wav files) from the original NIST 1A standard to allow easy handling.

2.1 Projects

In each of the three subdirectories there are the same projects. There are two types of projects. The “test example projects” are used to test foundation libraries. These projects are also useful since they show practical applications of the foundation libraries. The other projects refer to programs concerning speech recognition. The directory where the project is located coincides with the directory where the main source files can be found.

The test example projects are (Chapter indicates where the subject is covered):

Project	**Directory**	**Chapter**	**Description**
baseclas_polymorf	baseclas	2	This project tests the class implementing polymorphism. This class is used to implement “drivers” that handle different databases (Chapter 2) or different DSP operations (Chapter 3).
baseclas_testbase	baseclas	1,2	This project tests the classes handling memory and strings. The class handling memory is the root class from which all the other classes are derived. The project also tests the diagnostics.
ioclass	ioclass	2	This project tests the class that retrieves data from speech databases.
feature	feature	3	This project tests the class that performs feature extraction. This class is designed to perform arbitrary sequences of digital signal processing on the input sequence according to the configuration file.
resconf	resconf	2	This project tests the class that handles configuration services.
utils	utils	1	This project shows a simple program that

			performs arbitrary sequences of operations on a list of files according to the configuration file. The implemented operations are utilities for conversion from MS-DOS to Unix.
vetclas	vetclas	4	This project shows and tests the mathematical operations over vectors, diagonal matrices and full matrices.

The projects related to programs specifically required for speech recognition are:

Project	**Chapter**	**Purpose**
Print_feature	3,8	This project writes features of each single sound file. This is useful to avoid recomputing features in the embedded training procedure.
endpoint_feature	3,8	This project does the same as Print_feature but eliminates silences.
Print_phon_feature	3,8	This project writes features of the required files where all the same phonemes of all the files are collected in one file, i.e. one output feature file for each phoneme. This is required for non-embedded training.
initiali	4	This project initializes the HMM models. HMM model parameters are evaluated according to a clustering procedure.
training	5	This project re-estimates HMM models phoneme per phoneme using the Baum–Welch algorithm. The bounds of each phoneme within the utterances are required, i.e. segmentation of all the training speech data.
embedded	5	This project re-estimates HMM models per utterance using the Baum–Welch algorithm. Segmentation is not required.
lessico	6	This project estimates language model parameters according to various algorithms.
recog	7	This project performs phoneme/word recognition.
segmen	8	This project performs phonetic segmentation.
eval_rec	8	This project evaluates accuracy of word/phoneme recognition.
eval_segm	8	This project evaluates accuracy of segmentation.

2.2 Installation

To install the software, copy the directories according to use, the compiler/platform, and the table below. Files are pointed using relative positions and therefore no problems should be encountered with the relative position of the directories. Moreover, since the CD-ROM files have attribute “read-only”, after copying files on a hard disk, remember to set files-permissions to read and write.

For example, to run the demos only, copy the whole directories "Test_me" and "Sndfile" in a directory say "RES", then set all the file attributes to Read and Write. Finally, execute in the directory "Test_me/Phoneme" the file Start_me.bat or Start_me.lnx for MS-Windows or Linux systems respectively. Some files will be created with the results and performance.

directory to copy-> use	Test_me	Source, SndFile	projects/ project_ms/ ms_make	projects/ project_ms	projects/ project_DjGpp	projects/ Proj_lnx
only demo	×					
command line compilation with MS compiler	×	×	×			
IDE (graphic) MS compiler	×	×		×		
command line compilation with DjGpp	×	×			use make-files located in the directories of the correspon-ding projects.	
RHIDE graphic environment with DjGpp	×	×			×	
RHIDE graphic environment with Linux	×	×				×

2.3 Compilation

Files have been compiled under Gpp versions 2.8.0 or later in the porting of Delorie for MS-DOS. PREVIOUS VERSIONS DO NOT WORK due to some non-standard implementation of templates. The integrated environment Rhide has also been used. Pay attention to DjGpp Rhide, Windows 95 and long file names. These work when proper installation of compilers is carried out and options have the right values LFN = Y in the djgpp.env file and execute "rhide –y". In any case you can disable long file names by setting LFN = N and executing rhide with "rhide –n"

Files have been also compiled with MS Visual C++ 5.0. Other platforms/compilers have been also used such as Linux Gpp version >2.8.0, Borland >=4.51, MS Visual C++ 4.0. Porting on different platform/compilers should not be critical taking into consideration the differences between file systems.

Finally remember that in Linux/Unix, lines of text files terminate with a simple line feed (\n == char (10)) while in MS-DOS lines terminate with a cursor return followed by a line feed (\r \n == chr(13) + chr(10)). Therefore, in text files produced by MS-DOS programs cursor return should be eliminated when the same files are to be used in Linux. To achieve this, the program util.exe or other similar programs such as DOS2UNIX can be used.

3 COPYRIGHT

4 FUTURE DEVELOPMENTS

We are currently testing the adaptation module to create models adapted to a specific speaker or a given database given the multi-speaker models (see Chapter 5, Maximum A Posteriori adaptation, for reference). This module will be distributed on the Web (see www.fub.it/res).

We are also testing a new recognition module based on a tree structure (see Chapters 7 and 8). These modules are finalized to create a real-time recognition module. The actual module is many times slower than real time. We hope also that independent users will help us in developing RES. In this case, we will be happy to include the new software on our Web (www.fub.it/res) server if required.

5 WHAT TO DO WITH RES SOFTWARE

Currently, RES is a laboratory tool useful for testing new research ideas or as a start-up for commercial applications. The latter use will be more attractive when the real-time recognition module is available. As a laboratory tool, RES is also used to compute new models suited for particular applications. Software solutions implemented in RES are also of interest. RES has been developed according to the conservative strategy that, in our experience, has allowed faster development of more robust software even for inexperienced programmers such as our students (see Chapter 1 for details). In such a method of programming, pointers are never used (as happens in Java), while efficiency remains comparable with respect to software using pointers. The main services generally required for any large mission-critical software are also covered. See, for example, Chapter 1 for memory diagnostic and configuration services; Chapter 2 for mathematics and Chapter 3 for digital signal processing issues.

6 FILE INDEX

The following are the source files and the directories contained in the Source directory. The source files are used for different platforms and compilers.

1) baseclas	baseclas.cpp
2) baseclas	Baseclas.h
3) baseclas	Baseclas.hpp
4) baseclas	Boolean.h
5) baseclas	Compatib.h
6) baseclas	Defopt.h
7) baseclas	Diagnost.cpp
8) baseclas	Diagnost.h
9) baseclas	Polymorf.cpp
10) baseclas	Polymorf.h
11) baseclas	Polytest.cpp
12) baseclas	Testbase.cpp
13) baseclas	Textclas.cpp
14) baseclas	Textclas.h
15) embedded	Emb_b_w.cpp
16) embedded	Emb_b_w.h
17) embedded	Emb_Train.cpp
18) eval_rec	evalopt.cpp
19) eval_rec	evalopt.h
20) eval_rec	Evaluate.cpp
21) eval_rec	Evaluate.h
22) eval_rec	eval_rec.cpp
23) eval_segm	eval.cpp
24) eval_segm	eval.h
25) eval_segm	main_eval.cpp
26) features	DSPPROC.CPP
27) features	endpoint.cpp
28) features	Feature.cpp
29) features	Feature.h
30) features	mean_feature.cpp
31) features	print_file_feat.cpp
32) features	print_ph_feat.cpp
33) features	Test_feature.cpp
34) Initiali	Iniopt.cpp
35) Initiali	Iniopt.h
36) Initiali	Initiali.cpp
37) Initiali	Initiali.h
38) Initiali	Proiniti.cpp
39) ioclass	labelcl.cpp
40) ioclass	labelcl.h
41) ioclass	Soundfil.cpp

42) ioclass	Soundfil.h
43) ioclass	Soundlab.cpp
44) ioclass	Soundlab.h
45) ioclass	TESTIONE.CPP
46) ioclass	Test_MsWav.cpp
47) lessico	lessico.cpp
48) lessico	lessico.h
49) lessico	lexopt.cpp
50) lessico	lexopt.h
51) lessico	main_lessico.cpp
52) recog	hypolist.cpp
53) recog	Hypolist.h
54) recog	Hypolist.hpp
55) recog	recog.cpp
56) recog	recopt.cpp
57) recog	recopt.h
58) resconf	resconf.cpp
59) resconf	Resconf.h
60) resconf	TESTCONF.CPP
61) segment	Hypolist.cpp
62) segment	Hypolist.h
63) segment	hypolist.hpp
64) segment	hypolistseg.cpp
65) segment	Segment.cpp
66) segment	Segopt.cpp
67) segment	Segopt.h
68) training	Baumwelc.cpp
69) training	Baumwelc.h
70) training	Protrain.cpp
71) tspecmod	testtspecbase.cpp
72) tspecmod	Tspecbas.cpp
73) tspecmod	Tspecbas.h
74) tspecmod	Tspecbas.hpp
75) utils	multifop.cpp
76) utils	multifop.h
77) vetclas	Arraycla.cpp
78) vetclas	Arraycla.h
79) vetclas	Arraycla.hpp
80) vetclas	Diagclas.cpp
81) vetclas	Diagclas.h
82) vetclas	Diagclas.hpp
83) vetclas	Testvet.cpp
84) vetclas	Vetclas.cpp
85) vetclas	Vetclas.h
86) vetclas	Vetclas.hpp