RECNET - the speech recognition system

Jiří Bittner
xbittner@sgi.felk.cvut.cz

TU Delft
Faculty of Technical Mathematics and Informatica
June 1996

Contents

Acknowledgments

In May and June 1996, I took a part on a short-term stay at the Department of Technical Informatica of the TU Delft. This was possible thanks to a cooperation between the Czech Technical University in Prague and the University of Delft. First of all, I would like to thank to Dr. Leon Rothkranz, for his great support of the exchange program and his help during my stay at the department. I acknowledge the great contribution of Tony Robinson from the Cambridge University to speech recognition, by making RECNET publicly available.
My deep gratitude goes to Jac Speelman and other members of "speech recognition group" for their continuous support. Finally, let me thank to my flat-mates from "Het Gros" for an excellent time I had there.

Jiří Bittner
Delft, June 1996

Preface

This paper presents some practical experiences with RECNET - continuous speech recognition system, as well as some theoretical background.

First chapter is a brief introduction to fundamentals of speech recognition and was added just for completeness.

Second chapter describes theoretically the RECNET system. It is based on articles of Tony Robinson - the author of RECNET.

Chapter 3 deals with the problem of adapting RECNET to training on various platforms, especially on nCUBE supercomputer. It contains description of parallel nCUBE implementation and presents some results of the training process.

The Appendix contains description of the extended version of RECNET v.1.3, called RECNET-E . Overview of file formats, executables and scripts, together with some examples of their usage is presented.

Chapter 1

Introduction to automatic speech recognition

1.1 Acoustic preprocessing

In the recognition process the classification by acoustic processing is preceded. In this preprocessing phase the speech signal is analyzed to extract spectra-temporal features from the speech signal.

To automatically recognize speech, the acoustic signal first must be picked up by microphone, and fed into a computer [2]. The microphone signal is converted from an analogue to a digital representation **(A/D conversion)**. This means that the signal is measured (sampled) at fixed time intervals. These intervals generally range from 8000 times per second (8 kHz) for speech of telephone quality, up to 44100 times per second for Hi-Fi quality (compact disc). The values of these samples are quantized generally using 8,12 or 16 bits. The samples of the speech signal are, in their raw form, not very suitable as input for classification algorithms. Therefore the speech samples have to be transformed to parameter vectors **(feature vectors)**, which describe the speech at a lower rate, e.g. 100 Hz. This transformation is referred to as acoustic preprocessing or simply **preprocessing**.

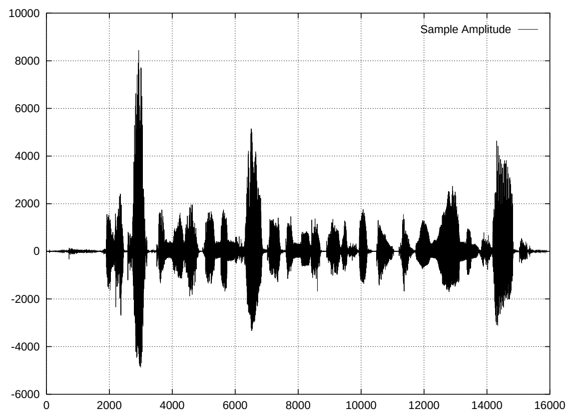

Figure 1.1: Acoustic wave of the sentence - "His captain was thin and haggard and his beautiful boots were worn and shabby."

In the preprocessing a **spectral analysis** is performed. The two main approaches to spectral analysis are the filter-bank analysis and the LPC analysis. The

filter-bank analysis estimates the energy of frequency bands [3]. Normally, filter-banks provide a frequency resolution of 7 up to 40 filter-bands in the frequency region from 0 to 5000 Hz. Examples of filter-banks are described in [4, 5, 3, 6, 7, 8].

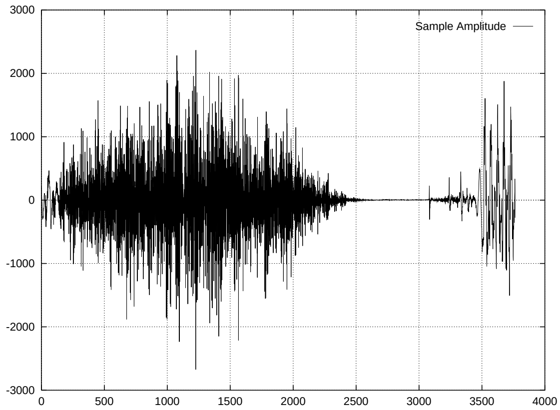

Figure 1.2: Word "beautiful".

The basic idea of **LPC** analysis is to create a model that predicts the new output (sample) from linear combinations of the present input and past output samples. Then, the speech signal can be described by the parameters of this model. For details about LPC reader is referred to [1].

Several comparisons of filter-bank and LPC analysis have been performed. In [9] resulting feature vectors were compared, while in method of comparison is a recognition experiment as shown in [10].

1.2 Classification algorithm

Research in automatic speech recognition in the early seventies resulted in the first practical recognition systems. Overviews about these developments can be found in [4].

Once the preprocessing has yielded an appropriate spectra-temporal representation of the speech signal, the classification algorithm can be applied. Two different approaches in classification are those based on explicit (speech) knowledge and those based on the description of (speech) patterns.

1.2.1 Knowledge based systems

Knowledge based systems try to obtain explicit (phonetic) knowledge about speech, and try to use it for the recognition [11]. Expansions to the system by introducing explicit phonological knowledge can be made [12]. Parameters that are used to describe the speech signal have therefor a phonological, phonetic, and/or acoustic origin. Some researchers argue that, although developing such system may be a long way, it will finally prove to be the best way to obtain high performance unconstrained recognition. Up to now this has not been proved to be true; actually, the alternative pattern-classification based recognizers have so far shown more progress towards the goal of unconstrained recognition. On the other hand, if one is interested in the thru nature of speech, the knowledge-based approach is the best way to obtain and to verify profound insight in linguistics and speech production.

Once we have obtained sufficient knowledge about speech, this approach to speech recognition will probably be superior, but as long as we do not have all knowledge, the recognition system will be confronted with events it cannot handle properly. One of the ways to obtain knowledge about speech, is by so-called **spectrogram reading**. A spectrogram is a time-frequency plot in which regions with high energy are highlighted. Frequency-selective energy concentrations present in a signal are visualized as a function of time. A human expert can locate phonetic events by looking at these spectrograms, and define rules for describing events. These rules must be formalized to become the knowledge representation in a recognition system.

1.2.2 Pattern classification based systems

The alternative approach to speech recognition is based on a description of speech in the form of patterns. The essence of this approach is that examples of speech events are collected, which are transformed into a reference set of patterns. The new speech is then compared to all reference patterns, and the best matching pattern is selected.

In designing a recognition system based on pattern classification, first the unit that is to be modeled has to be established. This unit can be a **word** or a **sub-word** (syllable, phoneme, acoustic event). When the recognition system only has to recognize a fixed number of words, whole words can be modeled. In case the vocabulary of the system is larger, and the vocabulary is likely to be extended, sub-word units are more efficient. In that case different words of the vocabulary can share the same sub-word units.

One of the most difficult problems in speech recognition is the variation in the **duration** of various speech units. By varying the duration of phones and words, a speaker emphasizes content of words, or shortens function words. The same phoneme can have a different duration depending on the syllable position in the word and on the speaking rate. For example, it appears that the longer the word is, the shorter the phonemes in that word are. Good references for research on duration are [13]. Another very important phenomenon in choosing and modeling an appropriate unit, concerns within- an between- word **coarticulation** [14]. Coarticulation can be defined as the influence of one speech segment upon its neighbors. This effect is most prominent with adjacent sounds (phones). One of the solutions to the coarticulation effect is to make the units context dependent.

Various algorithms have been developed to perform pattern classification. For all algorithms, realizations of the unit of modeling have to be collected. Then, a model is formed out of these examples. However, if the examples are too diverse, different models have to be made (multiple models or templates). A clustering technique can be used to select speech examples that are alike.

1.2.3 Commonly used pattern-classification algorithms

1.2.3.1 Dynamic time warping

Dynamic time warping is one of the most successful methods for pattern classification [15]. This non-probabilistic algorithm is able to perform a non-linear time alignment between two patterns. One of the signals is the reference pattern and the other is the speech pattern to be recognized. In most (commercial) whole-word recognition systems, that have been developed in the 70's and 80's the DTW technique was used.

1.2.3.2 Hidden Markov Modeling

Hidden Markov modeling is a technique that has become popular in speech recognition research from about 1975. Like DTW, HMM also allows to perform a non-linear time alignment. One of the important extra features of HMM is that in the alignment a probability is assigned to events. Therefor the HMM technique is more flexible.

1.2.3.3 Neural Networks

Artificial neural networks are the most recent approach to speech recognition [16]. In classification of isolated events, NN's already have proven to be a useful technique [17], but in modeling temporal variations (time alignment) HMM is still superior. One of the most recent modifications to the NN's, is the Time-Delay NN (TDNN), by means of which a better modeling of time variation can be achieved [18].

Some hybrid systems have been built, where NN's perform the preprocessing, and HMM's perform pattern classification. An example is the RECNET system, based on Recurrent NN and HMM [22, 21].

1.3 Issues in speech recognition systems

Up to now, no "ultimate" speech recognition system was built. A perfect recognizer should have capabilities similar or behind, those of a human listener. For many reasons such a perfect recognizer cannot be achieved yet, so any recognition system has to deal with the following issues, and must specify its limitations.

- **Style of speech**
  Different speaking styles are : free conversation, read text,isolated words or single words. Systems with highest level of artificial intelligence are called **speech understanding systems** (they "understand" the questions or commands of the speaker) and **spoken language systems** with natural dialog interaction between man and computer. A system capable of speech to text transcription of read text, is referred to as **continuous speech recognition system**. One of the issues in speech recognition is the segmentation of a utterance into words. Therefore certain systems require the speaker to speak with pauses between words. A recognizer designed for such kind of speech is called **isolated word recognition system**.

- **Speaker dependency**
  There are large differences between the speech of different speakers. whereas the speech of one speaker can vary between repetition of the same utterance, the variations amongst the speakers are even more significant. **Speaker dependent system** does not model inter-speaker variations, while if speech of many speakers is used to create the speech model, such system is then called **speaker independent**. Another way to deal with speaker variation is to train the models on a database with few speakers and adapt to new speakers (**speaker adaptive systems**).

- **Vocabulary**
  The vocabulary sizes are categorized to following groups :

-	small	up to 10 words
-	middle size	about 100 words
-	large	about 1000 words
-	very large	about 20000 words
-	unlimited	

- **Grammar**
  Recognition systems often use grammar to model syntax of the language defined for an application. The grammar model substantially reduces the difficulty of the recognition process. Given a recognized word (sequence of words), the grammar can specify the set of succeeding word or probability of their occurrence. An information measure for the average number of words that can follow any other word is the average branching factor or **grammar perplexity**. In most systems regular grammar, word-pair grammar or bi-grammar are being used.

Chapter 2

Theory of RECNET

2.1 Introduction

In the field of large vocabulary speech recognition it is acknowledged that sub-word units must be used as an intermediate state between the acoustic and lexical levels. A common choice for a suitable sub-word unit is the phone and the use of pronunciation dictionaries justifies this level of description. However, there is a great variation in the acoustic realization of a phone, both between speakers and for single speaker depending on context. Some method of modeling this variation is required in order to produce high accuracy speech recognition systems.

Therefor a recurrent error propagation network is being used for modeling the low level mapping of the context dependent acoustic information to a context independent phonetic form. This is followed by a simple Markov model to perform the decoding to a sequence of words. This method places the computational burden at a lower level than a standard triphone based HMM system.

This paper contains just elementary information about the idea of combining connectionist model and HMM for speech recognition task as formed by Tony Robinson and reader is referred to one of his papers.

2.2 The TIMIT database

Accurate comparison of different speech recognition systems is a difficult task. It is therefor important to evaluate recognizers on a standard database. The DARPA TIMIT Acoustic Phonetic Continuous Speech Database (referred to as TIMIT database) has been designed to be used for training recognizers at the phoneme level. However, with some difficulties, it is possible to use this database also for performing word recognition. Difficulties arise from the fact, that phoneme set used for labeling the training and test sets differs from one used in pronunciation dictionary.

TIMIT database contains training and test portions, recorded by 317 and 103 speakers, respectively. Eight sentences were used per speaker. It is possible to extract speaker identity from the path and filename on the TIMIT CD. Each sentence is stored as 16-bit .wav file, together with time aligned phoneme (.phn) and word (.wrd) tables. The text (.txt) file contains the speaker's prompt in time of recording the wav file of the same prefix. The database contains 61 TIMIT symbols (phonemes), while there's just 48 symbols used in pronunciation dictionary.

Table 2.1 illustrates some important facts regarding to the TIMIT corpus.

Set	sentences	distinct texts	distinct words	distinct phonemes
whole corpus	6300	2342	6099	45
train set	4620	1718	4891	45
complete test	1344	624	2371	45
core test	192	192	902	45

Table 2.1: TIMIT database contains

2.3 Preprocessor

The preprocessor used in RECNET was a result of a comparison of many preprocessors for this systems [22, 21]. Linear Predictive Coding (LPC), Fast Fourier Transform (FFT), filter-bank and auditory model techniques were compared by deriving a form of normalized power spectrum plus a power channel for each. In the case of LPC and FFT, this power spectrum was also represented as a cepstrum. The addition of other features , such as zero crossings and estimates of the pitch and formant positions and amplitudes were also investigated. In all cases a 32 ms Hamming window was used with a frame spacing of 16 ms. The conclusion was reached that most preprocessors, which were based around a power spectrum, gave similar performance. The simplest of these used the cube root of the powers in twenty channels derived from the FFT. For practical reasons, the reprocessed data is scaled to fit into 8 bits per channel. This is done by computing a histogram and scaling so that no more than one in 500 samples lies outside the central 15/16th of the range. Typically this means that one sample in 1000 is thresholded.

2.4 The recurrent net

2.4.1 Recall

A recurrent net can be considered as a sequence of error propagation networks where the input and output vectors are divided into external and internal portions. The default external input vector, $u_{0\ldots L-1}$, consists of 21 channels from the preprocessor. For TIMIT symbol set the external output vector, $y_{0\ldots M-1}$ has 61 dimensions (one per phoneme label), and is fed to the postprocessor. The internal output forms a state vector, $x_{0\ldots N-1}$, of dimension 192 and is fed to the same network in the next time period as shown in figure 2.1.

This network operates by concatenating the current external input and the last internal output vectors to give the complete input vector at time t:

$$o_i^{(t)} = \begin{cases} 1, & i = 0 \\ u_i^{(t)} - 1, & 1 \le i \le L \\ x_i^{(t)} - L - 1, & L+1 \le i \le N+L \end{cases} \tag{2.1}$$

which is then passed forwards through the network by performing a matrix multiplication by the weights, w_{ij}, followed by the application of a non-linear squashing function:

$$x_i^{(t+1)} = \frac{1}{1+\exp(-\sum_{j=0}^{L+N} w_{ij} o_j^{(t)})}, \quad 0 \le i \le N-1 \tag{2.2}$$

$$y_{i-N}^{(t+1)} = \frac{1}{1+\exp(-\sum_{j=0}^{L+N} w_{ij} o_j^{(t)})}, \quad N \le i \le N+M-1 \tag{2.3}$$

The resulting output is compared with the desired output vector, $d_{0\ldots M-1}$, according to a cost function. the outputs are treated as probabilities and this cross-

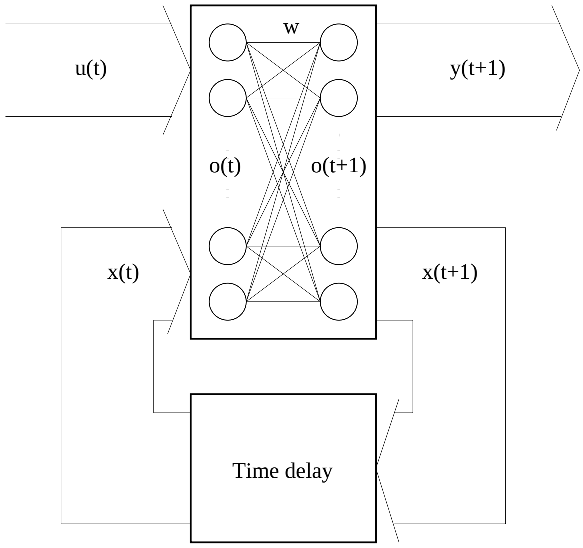

Figure 2.1: The Recurrent Error Propagation Network

entropy cost function is used:

$$\log(p^{(t)}) = \sum_{i=0}^{M-1} d_i^{(t)} log(y_i^{(t)}) + (1 - d_i^{(t)}) \log(1 - y_i^{(t)}) \tag{2.4}$$

2.4.2 Training

Training is performed by a forward parallel pass over a batch of 32 frames, the state unit being available from the previous time frame as in the recognition phase. The contribution to the gradient of the cost function from the frames in the window can be calculated directly by differentiation. The effect of future frames is disregarded, allowing the gradient at the state units at the end of the buffer to be set to zero. The derivates can then be "back-propagated" through the network, starting at the last frame in the buffer which gives the derivates for the state units used for the penultimate frame and so on back to the first frame. The assumption of disregarding future context is reasonable provided that there is sufficient evidence within the buffer for the context dependent features to be modeled. The start point of the buffer is varied on every iteration to average the effect of the boundary conditions. A local gradient is generated from the sum of gradients over 64 buffers, or about 0.2% of the complete training set.

The network used a novel algorithm for updating the weights. A positive step size for each weight is defined, and the weights are changed by this step size multiplied by the sign of estimated gradient. The estimated gradients are averaged with a first order filter, the "momentum term" [23], which started with a small time constant and increased over the first few passes through the training set until it is sufficient to smooth the estimated gradient over the whole of the training set. Initially all steps are equal , and the step is adapted by multiplying or dividing by a scaling factor, if the estimated gradient agrees or disagrees in sign with the smoothed gradient, respectively. The default scaling factor used in RECNET is 1.1. The step size are hard limited to be not greater than a certain factor (16) above or bellow the mean step size. This method has a theoretical disadvantage in that changes in the magnitude of step size can occur more rapidly than changes in the smoothed gradient. Thus it is possible to have a large smoothed gradient which consistently disagrees with the sign of estimated gradient and which may result in a rapid reduction of the step size to the lower threshold, so inhibiting further motion of that weight. In spite of this potentially disadvantage, this method was found to converge faster for this problem than any other method tried [Rob91].

The internal representation used to store context in the state units has no external constraints, and as such it is only dependent on the initial weights and their subsequent adaptation. Perhaps it is worthwhile to further study this internal representation. However, it is important that long term context can be stored in this structure.

2.5 Phoneme recognition

In a hand labeled data,a phoneme symbol is realized as a set of consecutive frames, each of which has the target output distribution for that symbol. A state transition model to generate the possible sequences of frame labels is used. The model is in one of the states which corresponds to the frame label at that time. The self loops allow for the label to be repeated and the other transitions allow for any of the possible phoneme labels to come next.

Any target sequence of frame labels can be compared with the actual output of the REPN by summing the frame-by-frame cost function of equation 2.4 for each frame. Thus it would be possible to generate every possible sequence of target frame labels, compare these sequences with the output of the network for an utterance, and pick the smallest distance to give the best match sequence of phonemes for that utterance. Alternatively, a more efficient method is that of dynamic programming [24], which may be used to find the best match. This method is the basis of speech recognition using Dynamic time warping [25], Hidden Markov Models employing a Viterbi search and is also used in most hybrid HMM/connectionist systems [26].

2.6 Word recognition

2.6.1 Markov model

The Markov model used was a simple context independent single pronunciation system with no provision for function word or cross word modeling. Scaled emission probabilities are derived from posterior probabilities computed by the network by dividing by the prior probability of the phone occurrence.

Previously, a variable rate approach to Viterbi recognition has been proposed for this system. A slight modification of this is to consider the probability that no phone transition occurs within a group of frames, or equivalently that the model remains in any one phone state for the whole segment. This technique shown in [Rob92], allows for a reduction in the frame rate by a factor of about three with no extra errors in the word-pair grammar case and a slight decrease in the number of errors in the case of no grammar.

It has been found that imposing state durations constraints yields a significant reduction in the number of errors. The minimum duration constraints are easily incorporated by rewriting every state as a sequence of states with tied emission probabilities. Only one of the states, then, has a non-zero self loop probability, so the computational overhead is minimal.

Chapter 3

Experimental results

3.1 Training the network

Training procedure was implemented in C as a sequential as well as parallel program.

As sequential it was compiled with gcc on Sun SPARC 5 machine running under Solaris-OS. The memory requirements of the program, using a large set of training data (whole TIMIT training set), are about 20MB.

Parallel implementation was tested on nCUBE 2 supercomputer, with 64 processing units. All communication advantages of nCUBE architecture were used while broadcasting and scattering data. Master/slave computational model was used, master being node 0 of the allocated sub-cube and performing also slave operations. This model has no computational overhead in compare to using a separate processing node for master, since there is always inherently sequential part of the process, while master node collects, averages and broadcasts received data. For broadcasting (sending improved weights to all the slave nodes) nCUBE's *nbroadcast* function were used, which is more efficient than simulating broadcast using *nwrite*. For all-to-one-scatter (collecting the gradient matrices), parallel sum algorithm were used, sending the matrix by *nwrite* function. Thus, communication flow is a binary tree rooted in node 0 (master), where edges on the same depth level correspond to disjunct communication links. Due to the fact, that communication occurs after certain number of local forward and backward passes of the network (default 32) and these passes take a lot more time than the communication it self, there is a small communication overhead in this algorithm.

Although some nodes have large memory capacity, there were limitations because of the units with just 4MB of memory. So, it was possible to run just one parallel program in time, because the process could not be "forked" on some of the nodes. Other difficulties arise from the different number formats of nCUBE and SUN machines. Swapping is performed when running on SUN machine when reading and writing binary file containing the training parameters [AppendixA].

The results of dry-stone benchmark test for SPARC 5 and single nCUBE node were about 150K and 7K, respectively, what did not seem to lead to significant improvement in the term of speed. However, comparing the actual speed of the training task, showed that the nCUBE implementation running on all 64 processors was about 5 times faster than the SPARC one. This speedup, which is better than expected running drystone test, is probably achieved because of the relatively faster floating point unit in compare to the integer unit on nCUBE and can be significantly improved by using faster parallel nodes, such as nCUBE 3.

The time needed for weights to stabilize, was varying depending on parameters of the training. The higher "momentum decay" [23], meaning longer time needed to

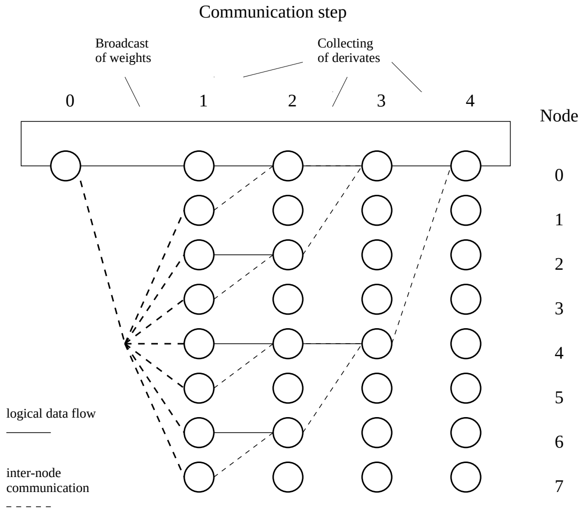

Figure 3.1: Example of the communication flow on 8 nCUBE nodes

smooth the gradient over whole training set, led to longer time necessary to stabilize the network, but generally achieved better results. An average time was about 40 hours.

Plots of some important indicators of the training procedure are shown on pictures 3.2 - 3.4, together with parameters used. The x-axis is a real number, corresponding to the number of passes through the training set. Time of one pass on the nCUBE was 53 minutes.

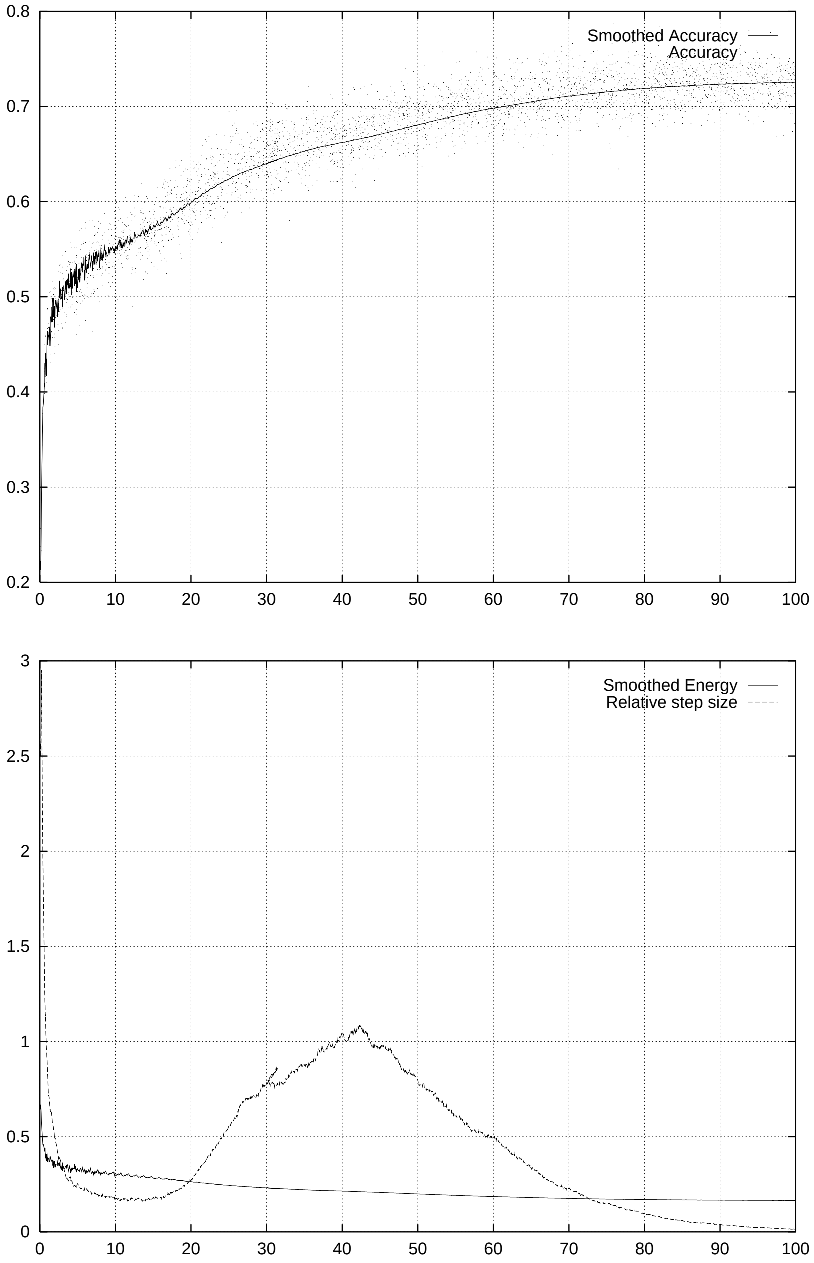

Figure 3.2: Training - α-decay=4.0, offsets=4

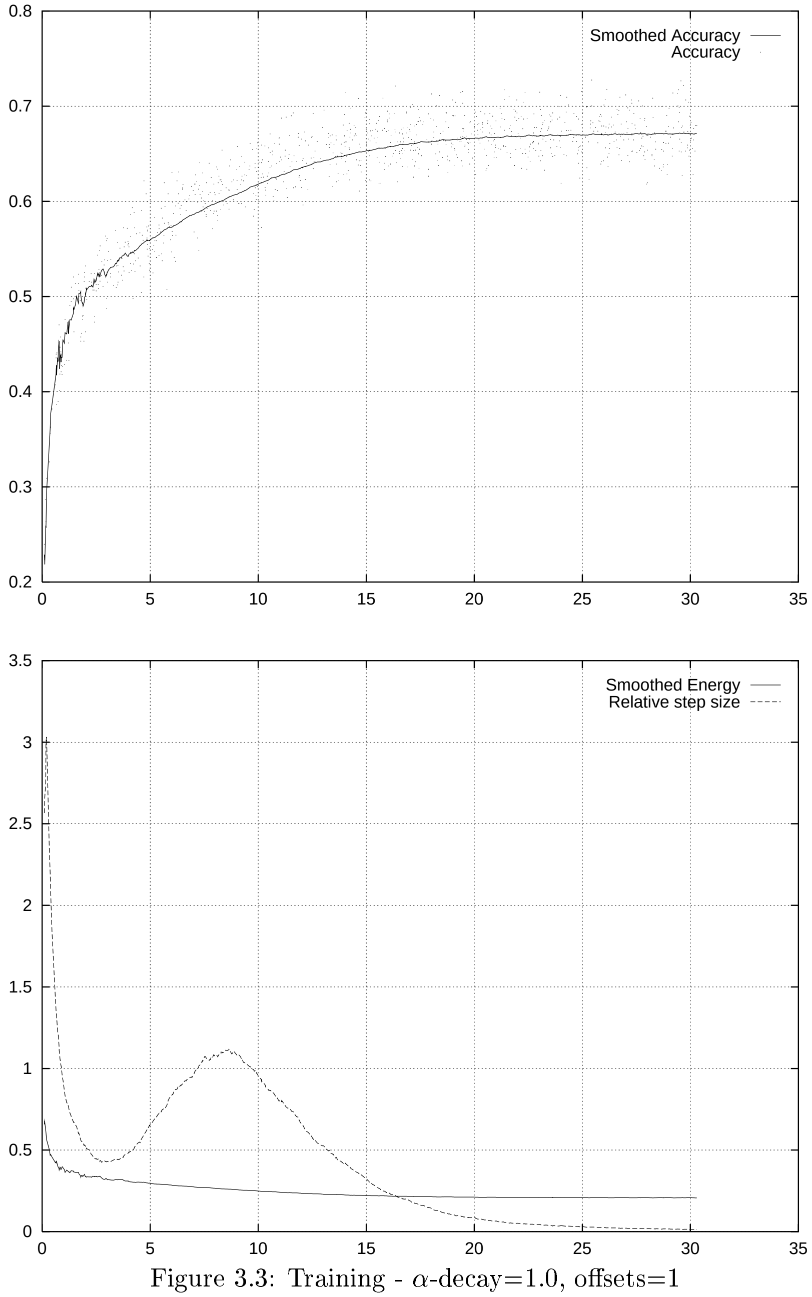

Figure 3.3: Training - α-decay=1.0, offsets=1

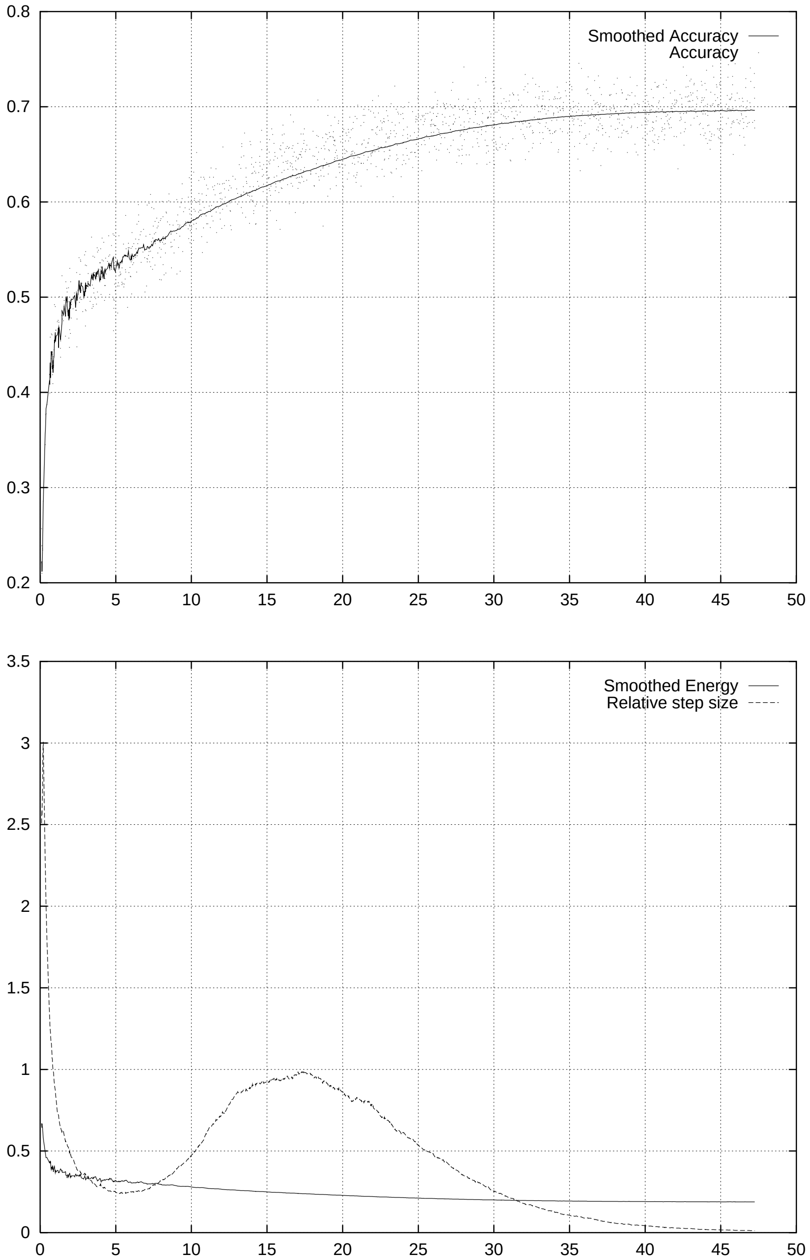

Figure 3.4: Training - α-decay=2.0, offsets=2

Weights	correct[%]	insert.[%]	subst.[%]	delet.[%]	error-rate[%]
original	73.5	3.4	20.2	6.3	29.9
new	73.0	4.2	20.3	6.7	31.1

Table 3.1: Results of phoneme recognition of first 100 sentences from the TIMIT test-set

3.2 Phoneme recognition

Accuracy of the phoneme recognition fully corresponds to tests performed by Tony Robinson as shown in [22]. In addition to the original RECNET system some scripts were created, so it is possible to perform the phoneme recognition passing just a name of a digital sound file (.wav) as a parameter of the script and obtain a time aligned phoneme string on the standard output [Appendix A].

Results of some tests are shown in 3.1.

3.3 Word recognition

The original RECNET system is made to perform the phoneme recognition task from the TIMIT database and the word recognition from the Resource Management database. However, it is possible to train the system on any of the speech corpuses, which meet certain criteria, and use it as a general speech recognizer independent of the corpus.

The script named `wav2txt` was created to perform such recognition. Since language model, some kind of dictionary and possibly other information is necessary to perform speaker independent speech recognition, more parameters to the script is needed. The "low-level" Language model is actually stored in file of the REPN weights (.wei) together with file containing information about phoneme durations (.dur) and phoneme table (.tbl). These parameters can be implicit to the script since they remain the same for a given language. A dictionary and some kind of grammar form task dependent parameters of the script. In RECNET the dictionary and pronunciation dictionary are not distinguished so the task dependent pronunciation dictionary (.dct) forms the task dictionary in the same time. It is possible to use word-pair grammar, stored as a binary file, generally of the same prefix as pronunciation dictionary. Thus only this prefix together with a name of the .wav file are necessary parameters to the word recognition script. Use of word-pair grammar is forced by a additional switch from command line.

To run a word recognition test for a set of sentences additional scripts were created, with possibility to play the sentence being recognized on a local machine.

Since the phoneme set used in pronunciation dictionary differs from the phonemes the network was trained on, a simple program was created (lna2lna), to combine phoneme-probability-estimations of phonemes witch are not used in pronunciation dictionary with most similar ones which are present. However, the results obtained using this method were even worse then simply using larger set of phonemes (training one) for performing word parsing. Because the phonemes in the pronunciation dictionary form a subset of the training one (after discarding the stress markers), the recognition was done without any modification to word parsing module. The proper solution to this problem is to train the network on the same set of phonemes used in pronunciation dictionary, thus for word parsing. Because of the lack of time this was not done yet and remains as a subject of possible future work.

Test	Sentences	Words	Perplexity	Word error rate [%]
complete	1344	2371	2.3	28.1
core	192	902	1.8	11.2

Table 3.2: Results of word recognition using word-pair grammar

Test	Sentences	Words	Perplexity	Word error rate [%]
complete	1344	2371	2371	69.8
core	192	902	902	63.5

Table 3.3: Results of word recognition without word-pair grammar

Results of tests on **core** and **test** sets of TIMIT database are shown in tables 3.2 and 3.3.

Example of Jac's scoring-algorithm output :

```
===================================

ref  : they make us conformists LOOK  GOOD
hyp  : they make us conformists LUNCH DIE
eval : C    C    C  C           S     S

correct words     :      4 (195)
substituted words :      2 (191)
inserted words    :      0 (32)
deleted words     :      0 (31)

word error rate   :   33.3% (60.9%)

total sentence errors (perc.) : 50 (100.0%)

===================================

ref  : we ALWAYS THOUGHT WE   WOULD die WITH OUR  boots on
hyp  : we ****** PLEASE  FELL EVEN  die **** VERY boots on
eval : C  D      S       S    S     C   D    S    C     C

correct words     :      4 (199)
substituted words :      4 (195)
inserted words    :      0 (32)
deleted words     :      2 (33)

word error rate   :   60.0% (60.9%)

total sentence errors (perc.) : 51 (100.0%)

===================================
```

3.4 Conclusion

The RECNET system was successfully compiled on SPARC-1+ and SPARC-10 running under Sun-OS and SPARC-5 running under Sun-Solaris operating system. For both systems gcc compiler was used together with Makefile, which is a part of RECNET distribution. Using make tool, phoneme recognition task from TIMIT database was performed, with expected results as mentioned in RECNET brief description (70% accuracy).

The goal of the project was to make RECNET adaptable to another language. To make this possible the REPN, which is the heart of the system, and together with phoneme duration module, the only language dependent parts, must be able to retrain. Thus, once be able to train the REPN there is no limitation in target language used for recognition.

The main problem is that system is not documented, there are practically no comments in the source code and the "data flow" is not very straight forward, since it is "hidden" in the Makefile dependencies, which were made to run just TIMIT or RM recognition tasks.

Although the system source code, includes the code for training on T-800 transputer array, the training procedure had been made unaccessible due to several reasons. Probably, two main reasons for that, were strong hardware dependency of code for training on T-800 and too great time necessary for training process.

First the original "not-used" code was modified to be able to run the training process sequentially on a single computer and to report the state of the training in form desired for visualization tools. Later, the training program was adapted to be possible to compile it and run it on the nCUBE. nCUBE version of the program was compiled by ncc compiler on the nCUBE's host machine, using modified original Makefile with an additional switch [see Chapter 3].

Since the program was running on the nCUBE, search for best parameters of training the started, to get approximately the same results as using the original weights for REPN. Finally the result of training was comparable with the one achieved by the author of the system (see table 3.1). While running the training on nCUBE, the attention was paid to create useful scripts for phoneme and word recognition to make the recognition process more straight forward and database independent. Besides this set of tools was created, to deal with preparing the dictionaries and grammar for word recognition, playing the sentences being recognized on Sun machines, handling the phoneme set differences, etc. An overview of the system implementation, flow, formats, tools created and so on is given in the next chapter.

Appendix A

The RECNET description

This chapter contains important information about actual implementation as well as some hints how the use different parts (modules) of RECNET for various purposes. This knowledge is based just on one month spent working with RECNET, while paying attention especially to implementation of the REPN. This chapter is only a basic introduction to the program and it is not meant to be reference manual. Moreover, the information comes without warranty, since it is hard to describe the system properly from uncommented source code not being an author of the system. However, this chapter could be useful for scientists with the aim of future extension of the system and those, who would like to implode parts of the RECNET into their own programs.

This chapter describes the enhanced version of RECNET v.1.3 referred to as RECNET-E .

A.1 Directory structure

The RECNET-E package contains a simple directory structure, to allow better understanding of system parts :

/	– C source codes of the main modules of the system (.c) – Various UNIX shell scripts – Main Makefile – Executables (compiled for SPARC and nCUBE)
/tools	– C sources of commonly used functions (.c) – Libraries of these functions for different platforms (.lib) – Makefile to create these libraries
/doc	– LaTeX source code of this report (.tex) – Postscript version of the report (.ps) – Other documentation and articles
/results	– Weights for REPN computed by training on nCUBE – Log files of the training – Results of phoneme and word recognition – Results of scoring algorithm
/backup	– Some files of original RECNET package, which were significantly modified in RECNET-E

A.2 Compiling

For compiling the system, use of make tool is preferred. Although the system is generally platform independent, there are some restrictions to this assumption.

1. For unknown target platform SWAP macro may be needed to define during compiling the system, since the integer format may vary from the default one [1]
2. 32-bit compiler is necessary
3. DOS users have to customize the Makefile to their compiler

A.2.0.1 Example of compilation of the main modules of the system

```
opel:~/speech/recnet-1.3> make compile

gcc -I tools -g -o myfon1 myfon1.c tools.a -lm
gcc -I tools -g -o wparse27 wparse27
```

A.2.0.2 Example of compilation of the library of tools

```
opel:~/speech/recnet-1.3> make tools.a
`tools.a' is up to date.
```

A.2.0.3 Example of compilation of nCUBE version of REPN module

This compilation must be performed on the nCUBE's host machine

```
dutisn:~/speech/recnet-1.3> make nc
ncc -Ncomm 2000000 -Nheap 1000000  -o nfon nmyfon1.o tools/ncubetools.a
-lm -lnpara
```

A.2.0.4 Examples of compilation of various tools using gcc

```
opel:~/speech/recnet-1.3> gcc -I tools -o swap swap.c tools.a

opel:~/speech/recnet-1.3> gcc -o wav2gnu wav2gnu.c
```

A.3 Data formats

Most of the files used in RECNET are text ones, what means that it is easy to get the idea about the file structure and format by viewing them on a terminal.

[1] The default format is big-endian. Wrong integer format leads to misunderstanding while reading binary files. In the case of RECNET-E these are the weights, word-pair grammar and duration file. Simple program called swap may perform necessary conversion.

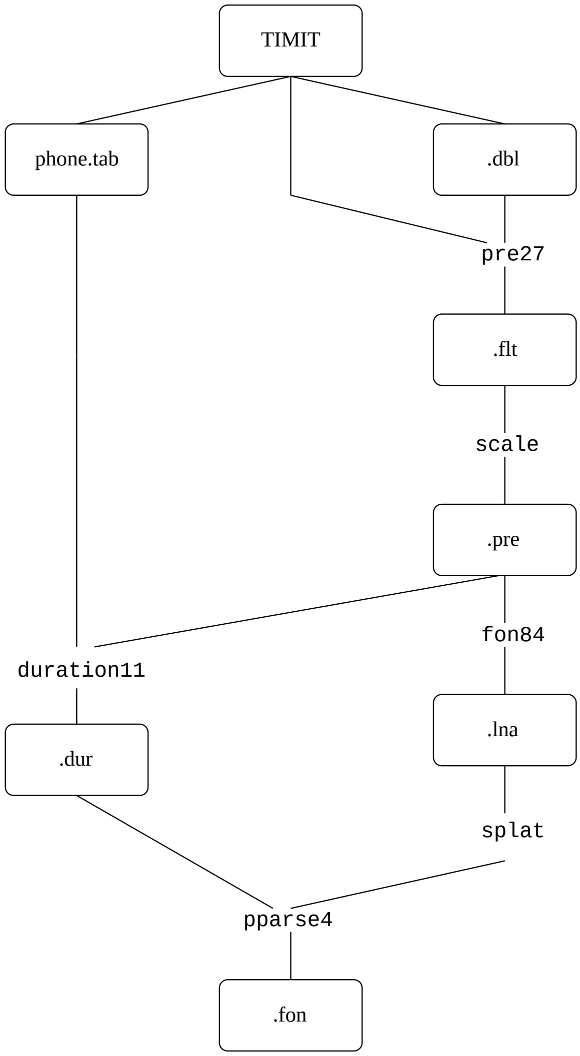

Figure A.1: Phoneme recognition data flow chart

A.3.1 Text files

.dbl Database of sentences.
Each row in the file corresponds to a path and a prefix of the sentence.[2]

```
test/dr1/faks0/si1573
test/dr1/faks0/si2203
test/dr1/faks0/si943
test/dr1/faks0/sx133
test/dr1/faks0/sx223
```

.tab Phoneme table.
Each line in the file contains one phoneme symbol, together forming set of phonemes for particular task. Example of set of 5 phonemes is shown.

[2]Prefix is used, because more than just one file corresponds to the sentence in the database (.wav,.wrd,.txt,.phn).

```
h#
pau
epi
eh
ih
```

.phn Time aligned phoneme string in the TIMIT database.
First column is the number of the first sample of the phoneme, second column contains the number of the last sample and third column is the phoneme symbol it self.[3]

```
0 8920 h#
8920 9530 hh
9530 10694 ih
10694 12281 z
12281 12930 kcl
12930 13710 k
13710 15707 ae
```

.fon As .phn, but produced by the RECNET system.

.wrd Time aligned word string in the TIMIT database.
Exactly the same format as .phn, except the third column contains the word instead of phoneme.

```
8920 12281 his
12281 20161 captain
20161 24187 was
24187 27710 thin
```

.wno As .wrd, but produced by the RECNET system, using no grammar.

.wwp As .wrd, but produced by the RECNET system, using word-pair grammar.

.dct The pronunciation dictionary.
Dictionary used for word recognition. First column is the word and all the remaining columns are phoneme transcription of this word. Phonemes must be separated by exactly one space symbol and new the line terminator must immediately follow the last phoneme in the string. Abstract words @INIT and @QUIT are desired by the parser. Dictionary should be lexicographically sorted.

```
@INIT h#
@QUIT h#
A ax
ABDOMEN ae b d ax m ax n
ABILITY ax b ih l ix t iy
ABOUT ax b aw t
ABRUPTLY ax b r ah p t l iy
ACCOMPLISHED ax k aa m p l ih sh t
```

.hyp Hypothesis file. Text recognised by RECNET.

.hno As .hyp, but created without using any grammar.

[3]This phoneme symbol must be present in the phoneme table file

.log Log file of the training process All important time dependent variables of the training are stored stored in the file, together with the training parameters. The file contains a header with command-line options. The rest of the file consists of several columns of real numbers, with a following meaning :

(a) Number of passes through training set
(b) α parameter of smoothing (see "Momentum term" [23])
(c) Relative step size
(d) Current energy of the network
(e) Smoothed energy
(f) Relative smoothed energy
(g) Percentage of correct frames in current batch of frame
(h) Smoothed value of correct-frames-percentage

A special character-prefix was used for text comments in the file, therefor it is ready to use by gnu-plot program. Some time stamps are present in the file, to record the information about training regarding to date and time.

A.3.2 Binary files

.wav The digital audio file.
First 1024 bytes is the file header, while the rest of the file are values of microphone signal quantization. These values are 16-bit signed words.

.flt The preprocesed file.
The file is a sequence of feature vectors, where each of them corresponds to a frame of the audio file. Each feature vector is stored as a sequence of $n+1$ floats, where n is the feature dimension selected in preprocessor. First float of this array is a phoneme index, when the phoneme time alignment is known [4]. Since more than once sentences can be preprocessed together, special symbol (128) is used in this place to separate them.

.pre The scalled preprocessed file.
The file is generally the same as .flt, just the elements of the feature vectors are bytes.

.lna The activation file.
The file contains estimations of probability for each phoneme in the phoneme table and corresponding frame. Thus, file consists of $p*n$ bytes, where p is the number of phonemes and n the number of frames in the preprocessed file.

.wei The weights.
The .wei file contains the weights for the REPN, state of the training together with additional information to allow to continue the training of the network [5].

.dur The duration file.
Structure of this file depends on the recognition task. In phoneme recognition this file contains bigram of phonemes and histogram of their durations. While in word recognition, the duration boundaries are stored together with self loop probability for each phoneme.

[4] Preprocessor can be forced to store this information

[5] The structure of this file is rather complicated and can be explored in source files tools/Write_machine.c and myfon1.c.

.wpg The word-pair grammar.

File consists of indices to the pronunciation dictionary and terminator symbol -1. Each entry in the file is 16-bit signed word. Each index in the file means, that the word of this index can follow the word with index equal to a abstract counter. This counter is incremented each time the terminator symbol (-1) is found. A text transcription of the file could like as follows :

```
3 10 34 -1 0 -1 -1 7 6 -1 9 6 1008 -1  98 87 -1 876 -1 5 -1
```

A.4 Executables and script tools

A.4.1 Preprocessing

A.4.1.1 The main module

Current preprocessor module is called pre27. All possible parameters are shown after launching the program without any arguments.

```
opel:~/speech/recnet-1.3> pre27
Syntax: pre27
        -verbose
        -norm
        -nophn
        -noswap
        -trimends
        -log
        -erb
        -mel
        -bark
        -basilarmm
        -nonlinear_exit
        -zc
        -f0
        -f0now
        -halfp
        -f0norm
        -voice
        -gender
        -peaksize
        -noise 0.000000
        -npower 1
        -nlpacf 0
        -nlpfilt 0
        -nlpchan 0
        -nlpceps 0
        -nlparea 0
        -nlpppos 0
        -nlppamp 0
        -nlpsgds 0
        -lporder 16
        -lpsmooth 1.000000
        -ftnonorm
        -nftchan 20
        -nftbscp 0
        -nftppos 0
        -nftpamp 0
        -nftstdd 0
        -minf 60.000000
        -maxf 8000.000000
        -samplef 16000.000000
        -frame 256
        -window 512
        -header 1024
        -offset 0
```

```
        -compress 0.333333
        -phntab /home/dsl1/ajr/lib/sriphone.tab
        -adc_prefix /home/dsl1/ajr/rm1/
        -phn_prefix /home/dsl1/ajr/DATA/rm1/
        -adc_suffix .wav
        -phn_suffix .phn
        <filelist file>
        <preprocessed file>
opel:~/speech/recnet-1.3>
```

Examples of preprocessor usage :

1. Phoneme time alignment is not known about .wav file (recognition)

   ```
   pre27 -nophn -f0 -voice -phntab tm-phone.tab -adc_prefix ./
   -phn_prefix ./ test.dbl test.flt
   ```

2. Time alignment of phonemes is stored in .phn files (preprocessing of the training set of the TIMIT database)

   ```
   pre27 -f0 -voice -phntab tm-phone.tab -adc_prefix /cdrom/cdrom0/timit
   -phn_prefix /cdrom/cdrom0/timit test.dbl test.flt
   ```

A.4.1.2 Scaling

Program called scale is used to map the floating point output to 8-bit values.

```
opel:~/speech/recnet-1.3> scale
Syntax: scale
        -byte
        -swap
        -linear
        -verbose
        -ninp 23
        -gap 8.000000
        <float train file>
        <-|float test file>
        <-|byte test file>
opel:~/speech/recnet-1.3>
```

Examples of usage :

1. Preprocessed training set is available

   ```
   scale train.flt test.flt  test.pre
   ```

2. Preprocessed training set is not used

   ```
   scale train.flt train.flt  test.pre
   ```

A.4.2 Duration modeling

Duration model is processed just once for current training set.

```
opel:~/speech/recnet-1.3> duration11
Syntax: duration11
        -pfp
        -half
        -nophn
        -graph
        -apriori
        -markov
        -verbose
        -self_loop
        -histogram
        -dictionary
        -ninp 23
        -lo_limit 0.093750
        -hi_limit 0.001953
        -nstateratio 0.500000
        -phntab /home/dsl1/ajr/lib/sriphone.tab
        <-|preprocessed file>
        <-|duration file>
opel:~/speech/recnet-1.3>
```

Example of usage :

1. Phoneme recognition

```
duration11 -apriori -lo_limit 0.0 -markov -ninp 23
    -phntab tm-phone.tab test.pre phn-train.dur
```

2. Word recognition

```
duration11 -apriori -self_loop -lo_limit 0.0 -ninp 23
           -phntab tm-phone.tab train.pre word-train.dur
```

A.4.3 Recurrent network

A.4.3.1 Single-processor version

```
opel:~/speech/recnet-1.3> myfon1
Syntax: myfon1
        -train
        -noerf
        -centre
        -state
        -debug
        -offline
        -linear
        -copyinit
        -nodiaginit
        -nophnsizenorm
        -inc 1.111111
        -dec 0.900000
        -step 0.001000
        -mmargin 1.040000
        -maxnpass 100.000000
        -maxweight 32.000000
```

```
        -nsum 2048
        -ninp 23
        -nsta 176
        -nout 61
        -seed 1
        -delay 4
        -savet 2
        -noffset 1
        -bias 1.000000
        -ainit 0.500000
        -adecay 1.000000
        -dump <no default>
        <-|preprocessor file>
        <weights file>
        <-|log file>
opel:~/speech/recnet-1.3>
```

Example of usage:

1. Recall

   ```
   myfon1 -dump test.lna test.pre timit.wei -
   ```

2. Training

   ```
   myfon1 -train -savet 10 train.pre new.wei new.log
   ```

A.4.3.2 Training on the nCUBE

All command-line options and parameters are exactly the same as for single processor version of the program. nCUBE's xnc loader is used to execute the program on more then one node.
Examples of usage :

1. `xnc -d6 -g nfon -savet 2 big.pre big.wei big.log`
2. `xnc -d6 -g nfon -savet 0 -adecay 2 -noffset 3 big.pre big.wei big.log`
3. `xnc -d5 nfon -savet 10  big.pre big.wei -`

Parameter "savet" is number of passes necessary to store the weights-file. "Adecay" is an α-decay of the "momentum term".

It is possible to save the current weights while training. To achieve this file called "save_now" must be created in the root directory of RECNET. So far the file is present, weights are being saved after each batch of forward-backward passes.

A.4.4 Postprocessing

A.4.4.1 Splitting of the activation file

Since its possible to join the frames of different sentences during preprocessing, some mechanism to allow their separation for post-processing must be present. This is done by program named splat, which splits the original file to desired directory structure. This makes the comparison of the results with the data in the database easier.

```
opel:~/speech/recnet-1.3> splat
Syntax: splat
        -verbose
        -nout 68
        -prefix /home/dsl1/ajr/DATA/rm1/
        -suffix .lna
        <filename file >
        <-|activation file>
opel:~/speech/recnet-1.3>
```

Usage:

```
splat -nout 61 -prefix cdrwm/ test.dbl test.lna
```

A.4.4.2 Phoneme parsing

```
opel:~/speech/recnet-1.3> pparse4
Syntax: pparse4
        -timing
        -frame 256
        -phntab /home/dsl1/ajr/lib/allophone.tab
        <activation file>
        <transition probability file>
opel:~/speech/recnet-1.3>
```

Example of usage:

```
pparse4 -phntab tm-phone.tab si1573.lna tm-test.dur
```

A.4.4.3 Word parsing

```
opel:~/speech/recnet-1.3> wparse27
Syntax: wparse27
        -print_speaker
        -phn
        -wrd
        -debug
        -bounds
        -forced
        -wpgram
        -verbose
        -conv2lna
        -print_silence
        -no_optional_silence
        -print_wplist_exit
        -print_duration_exit
        -print_dictionary_exit
        -frame 256
        -abias 1.000000
        -pbias 1.000000
        -wbias <12.000000|10.000000>
        -slp /home/dsl1/ajr/lib/rm1_3_32.dur
        -dict </home/dsl1/ajr/lib/pcdsril.dct|/home/dsl1/ajr/lib/pcdsril.dct>
        -wplist /home/dsl1/ajr/lib/rm1_wplist.wpg
        -phntab /home/dsl1/ajr/lib/sriphone.tab
        -act_prefix /home/dsl1/ajr/DATA/rm1/
```

```
        -phn_prefix /home/dsl1/ajr/DATA/rm1/
        -wrd_prefix /home/dsl1/ajr/DATA/rm1/
        -act_suffix .lna
        -phn_suffix .Phn
        -wrd_suffix <.wno|.wwp>
        -nist <nist output file>
        <-|list of files file>
opel:~/speech/recnet-1.3>
```

Example of usage :

```
echo si1573 | wparse27 -wpgram -nist result.hyp -slp word-train.dur
-dict core.dct -wplist core.wpg -phntab tm-phone.tab -act_prefix ./ -wrd_prefix ./  -
```

A.4.4.4 Scoring

For scoring the results, the dynamic programming match is used.

```
opel:~/speech/recnet-1.3> dpmatch2
Syntax: dpmatch2
        -verbose
        -confusion
        -ray
        -rprefix /home/dsl1/ajr/DATA/rm1/truedir/
        -rsuffix .wrd
        -uprefix /home/dsl1/ajr/DATA/rm1/
        -usuffix .wwp
        -sprefix /home/dsl1/ajr/DATA/rm1/truedir/
        -ssuffix .sym
        -cprefix /home/dsl1/ajr/DATA/rm1/
        -csuffix .cor
        -phntab /home/dsl1/ajr/lib/sriphone.tab
        <-|filename file>
```

Example:

```
dpmatch2 -verbose -rprefix /cdrom/cdrom0/ -rsuffix .phn
-uprefix cdrwm/ usuffix .fon test.dbl > results.res
```

A.4.5 Shell scripts and other tools

A.4.5.1 Phoneme recognition

To obtain phoneme string out of audio file a script call wav2fon is used. Prefix of the audio file (.wav) is passed to the script as the only parameter on the command-line. The result is a time aligned phoneme string and is printed to a standard output.

```
opel:~/speech/recnet-1.3> wav2fon si1573
0       8960    h#
8960    9472    hh
9472    10752   ih
10752   12288   z
12288   13056   kcl
13056   13824   k
13824   15872   ae
```

A.4.5.2 Word recognition

Several scripts were made to perform word recognition, using arbitrary .wav file as input. As described above for word recognition some informations about the language model are necessary. Two parameters -weights and duration model - are implicit to the script[6], while the prefix of the audio file and the prefix of the language model are passed on the command-line. The language model consists of the pronunciation dictionary (.dct) and the word-pair grammar. Usage of word-pair grammar is forced by a switch on the command-line.
Examples of usage:

1. Grammar free recognition

   ```
   wav2txt si1573 core
   ```

   (File core.dct must be present.)

2. Using a word-pair grammar

   ```
   wav2txt si1573 core -wpgram
   ```

   (Files core.dct and core.wpg must be present.)

A.4.5.3 Playing the sample

For test purposes playing of the wav files on Sun machines was implemented. This problem is not very straight forward, since Sun uses different audio format than the one of original data. Therefor some conversion is necessary. This is done by the program called `play` as well as an ability of playing the sound on a remote machine. For more information see the script.

A.4.5.4 Running the test

For running the recognition for a set of sentences, script called runtest was made. This program performs the word recognition and store its results in two files - .hyp and .hno. The .hyp file contains result of recognition for word-pair grammar case, while the .hno file is created by grammar free post-processing. Both files have the prefix of the argument given on the command-line.
Example of the core test:

```
opel:~/speech/recnet-1.3> runtest core
Processing timit/test/dr1/felc0/si1386 ...... ( no grammar / wp grammar )
IN WAY EACH NEGOTIATIONS THREE INDUSTRY BARGAINS HEALTHIER UNION WITH SINGLE
IN WAGE NEGOTIATIONS THE INDUSTRY BARGAINS AS A UNIT WITH A SINGLE UNION

Processing timit/test/dr1/felc0/si2016 ...... ( no grammar / wp grammar )
HEAVE OWN KNOWS ROPES THE BOAT'S COME UNSTUCK CAN
HEAVE ON THOSE ROPES THE BOAT'S COME UNSTUCK

Processing timit/test/dr1/felc0/si756 ...... ( no grammar / wp grammar )
MATERIALS US CERAMIC MOM IN CLAY WEB TWILIGHT OR THE LAUGH
MATERIALS CERAMIC MODELING CLAY RED WHITE OR BUFF

Processing timit/test/dr1/felc0/sx126 ...... ( no grammar / wp grammar )
ARTIFICIAL INTELLIGENCE HER WE OWN HER
ARTIFICIAL INTELLIGENCE IS FOR REAL
```

[6] This parameters can be modified in the script-header

A.4.5.5 Mapping the phonemes in .lna file

This programm was made to add probabilities of non-used phonemes in pronunciation dictionary to closest ones which are present. This was ment to be an easy way of handling the phoneme set difference in TIMIT database, what was not proved yet, partly because of the lack of time.
Text file describing the mapping contains two columns. First column contains phonemes which the network was train on, the second column are corresponding phonemes in pronunciation dictionary (see tm-61to48.tab).

```
opel:~/speech/recnet-1.3> lna2lna
Syntax: lna2lna
        -phntab tm-phone.tab
        -ophntab tm-48phone.tab
        -trans tm-61to48.tab
        <activation file>
        <output activation file>
opel:~/speech/recnet-1.3>
```

A.4.5.6 Swapping the integers

Because of the different number formats of some machines, program to swap all integers (floats) in a file was created.

```
opel:~/speech/recnet-1.3> swap
 Swapping ints in file :  convert file1 file2
opel:~/speech/recnet-1.3>
```

A.4.5.7 Makefile

The main Makefile in the system is not made just for compilation of program modules, but can be used for running phoneme recognition test on TIMIT database. The system of dependencies in Makefile describes data flow as shown in figure A.1. Some examples:

1. Running the TIMIT phoneme recognition test

   ```
   make TIMIT
   ```

2. Preprocessing of the TIMIT test set[7]

   ```
   make test.pre
   ```

[7] A subset of the whole test set can be processed - see NTEST in Makefile

Appendix B

Abbreviations

HMM	hidden markov model
NN	neural network
REPN	recurrent error propagation neural network
DTW	dynamic time warping
DP	dynamic programming
FFT	fast fourier transformation
LPC	linear predictive coding
TDNN	time delay neural network

List of Figures

Bibliography

[1] Makhoul, J. (1975). *"Linear prediction: a tutorial review"*, Proceedings of the IEEE, Vol. 63,561-580.

[2] Witten, I.H. (1982). *"Speech storage"*, in Principles of Computer speech, Academic Press,London,41-60.

[3] Dautrich, B.A.,Rabiner, L.R., Martin, T.B. (1983). *"On the effects of varying filter bank parameters on isolated word recognition"*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-31(4),793-806.

[4] Pols,L.C.W. (1971). *"Real-time recognition of spoken words"*, IEEE Transactions on Computers, Vol. C-20(9),972-978.

[5] Zwicker, E., Terhardt, E. (1979). *"Automatic speech recognition using psychoacoustic models"*, J. Acoust. SOc.Am., Vol. 65(2),487-498.

[6] Hataoka, N., Amano, A.,Yajima, S. (1986). *VCV segmentation and phoneme recognition in continuous speech"*, Proc. IEEE ICASSP 86, 2299-2302.

[7] Seneff, S. (1988). *"A joint synchrony/mean rate model of auditory speech processing"*, Journal of Phonetics, Vol. 16,55-76.

[8] Silverman, H.F. (1986). *"A high quality digital filterbank for speech recognition ...."*, IEEE Transactions on Acoustics, Speech and Signal Processing, VOl. ASSP-34(5), 1064-1073.

[9] Kay, S.M., Marple Jr, S.L. (1981). *Spectrum analysis - A modern perspective"*, Proceedings of the IEEE, Vol. 62(11),1380-1419.

[10] Davis, S.B., Mermelstein, P. (1980). *Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences."*, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-28(4),357-366.

[11] Zue, V.W. (1985). *"The use of speech knowledge in automatic speech recognition."*, Proceedings of the IEEE, Vol. 73(11),1602-1615.

[12] Church, K.W. (1987). *"Phonological parsing and lexical retrieval"*, Cognition, Vol. 25,53-69.

[13] Crystal, T.H., House, A.S. (1988). *"Segmental durations of fricatives in American English"*, J. Acoust. Soc.Am., Vol. 84(5),1932-1935.

[14] Schouten, M.E.H., Pols, L.C.W. (1979). *"Vowel segments in consonantal contexts : a spectral study of coarticulation - part 1"*, Journal of Phonetics, Vol. 7,1-23.

[15] Huang, X.D., Ariki, Y., Jack, M.A. (1990). *" Hidden Markov models for speech recognition"*, Computer speech and language, Vol.3 (3),239-251.

[16] Iso, K.I., Watanabe, T. (1991). *"Large vocabulary speech recognition using neural prediction model"*, Proc. IEEE ICASSP 91,57-60.

[17] Hinton, G.E., Sejnowski, T.J., Ackley, D.H. (1984). *"Boltzmann Machines: Constraint satisfaction networks that learn"*, Technical report, Vol. CMU-CS-84-119, 43 pages.

[18] Tebelskis, J,. Waibel, A., Petek, B., Schmidbauer, O. (1991). *Continuous speech recognition using linked predictive neural networks"*, Proc. IEEE ICASSP 91, 61-64.

[19] Hassoun, M. (1995). *"Fundamentals of Artificial Neural Networks"*, The MIT Press, London.

[20] Alphen, P. van (1992). *"HMM-based continuous speech recognition"*, PTT Research, Delft.

[21] Robinson, T. (1992). *"A Real-Time Recurrent Error Propagation Network Word Recognition System*, Proc. IEEE ICASSP-92.

[22] Robinson, T. (1991). *"A Reccurent Error Propagation Network Speech Recognition System*, Computer Speech and Laguage, Vol. 5.

[23] Rumelhart, D.E., Hinton, G.E., Williams R.J. (1986). *Learning internal representations by error propagation.* Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1, chapter 8. Bradford Books/MIT Press, Cambridge,MA.

[24] Aho, A.V., Hopcroft, J.E., Ullman,J.D. (1983). *"Data Structures and algorithms"*.Addison-Wesley.

[25] Ney, H. (1984). *"The use of a one-stage dynamic programming algorithm for connected word recognition"*, IEE Transactions on Acoustics, Speech and Signal Processing, 32(2):263-271.

[26] Morgan,N.,Bourland,H. (1990). *"Continuous speech recognition using multilayer perceptrons with hidden Markov models"*. In Proc. ICASSP, 413-416.