Lab Report

Speech Recognition

December 3, 2015

Contents

1 Introduction

Automatic speech recognition is the task of converting a speech signal into (a sequence of) words. The purpose of this lab is to build a digit recognizer and to explore mechanisms that will extend and improve its functionalities. The initial aim will not be processing loose speech into *sequences* of words, rather single uttered words within a fixed vocabulary.
After an introduction to the tools used for the experiments, an explanation of the various steps to build a simple recognizer, and the theory behind each of them is supplied. Ultimately, various extensions to the initial model are implemented and analyzed and the correspondent results are provided.

2 Methodology

2.1 Background

A digit recognizer is a system capable of recognizing uttered digits. The system can be modeled in various ways; in this report only a particular approach is analyzed. The approach is based on statistical modeling and the use of Hidden Markov Models for classification. The recognizer is implemented with the 3rd version of the Hidden Markov Model Toolkit (HTK) which provides modules for speech analysis, HMM training, testing and result analysis.
This approach uses supervised machine learning to classify utterances, where relations are learned from *labeled* data, and classifications made accordingly.

2.2 Speaker dependent digit recognizer

A speaker dependent digit recognizer is a system modeled to classify individual uttered digits from the single speaker it was both trained and tested on. The process to build such a system can be roughly divided into five steps, the details of which are hereby reported:

Gathering and labeling data

The first step to build a simple digit recognizer, is gathering the data to train and test the model. The training data is recorded from a single speaker which utters digits from zero to nine in a random order (so to avoid capturing eventual propagation effects). The

data is recorded with a single microphone, and is subsequently segmented and labeled with either the uttered digit(e.g. *one*, *two* ...) or *junk* for silences and noises. The digit labels correspond to the classes the utterance instances can belong to. The test data is gathered similarly, however, the audio file is split in multiple files each containing an uttered digit and the label for each utterance is attached separately in a *mlf* file (the labels are needed to compute the accuracy of the system after prediction). All audio files are saved in a *wav* format.

MFCC conversion

The second step corresponds to converting and parameterizing the sound signals to Mel frequency ceptral coefficients (MFCC), a convenient way of separating the source signal and the filter for the utterance (Jurafsky and Martin, 2008, section 9.3). The conversion of a sound wave to MFCCs mainly follows five phases.

The first phase, called preemphasis, consists in incrementing the energy level of the sound signal at low frequencies to better capture the information there available.

Following the preemphasis, the signal is windowed at a specific sampling rate (16kHz for this task, the majority of the information in human speech is in frequencies below 8kHz). The extracted windows, or *frames*, are meant to be small enough to approximate speech invariant properties.

At this point, the spectral information of each frame is extracted by applying a Discrete Fourier Transform (DFT) through the FFT algorithm, obtaining what is known as the *magnitude spectrum*.

To improve recognition accuracy it is now important to embed domain knowledge in the obtained signal; this, is achieved by passing the magnitude spectrum through a Mel filter-bank which non linearly modifies the frequency scale to capture increasingly less variations as we go from low to higher frequencies. Moreover, F0 is smoothed away given the widths of the harmonics in the Mel-scale effectively fail to capture its variations (King, 2015, slide 50).

Finally, after applying a log scale, the resulting waveform is broken down to its wave components through the Discrete Cosine Transform (DCT) which captures the spectral envelope of the signal. The coefficients for the waves at different frequencies are then

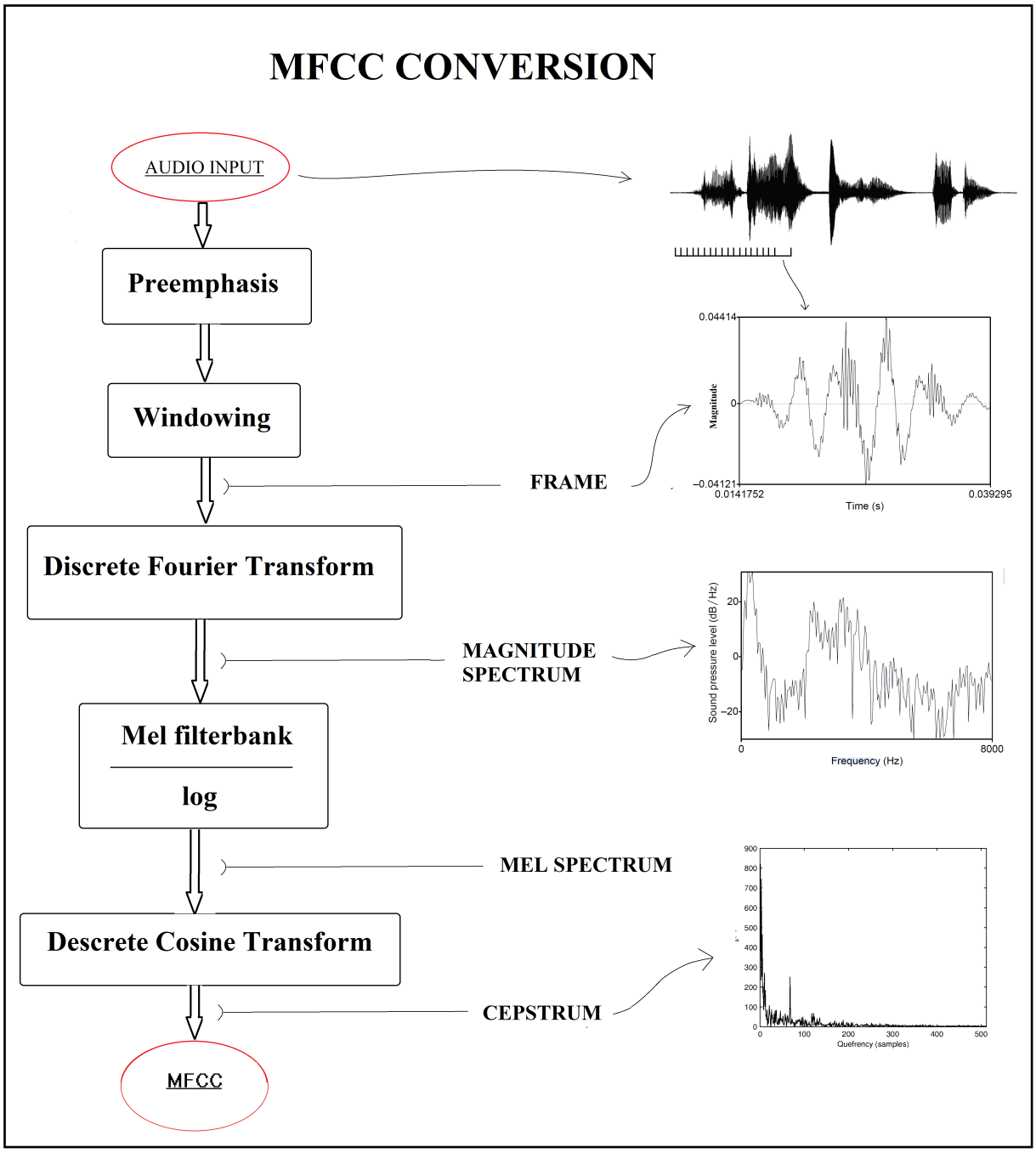


Figure 1: Steps to convert an audio input to MFCCs.

extracted and used as features in the frame's vector. The coefficients plotted in the time domain form the cepstrum (see Figure 1), an important representation useful in identifying the source filter, which can typically be fairly captured by the first 13 coefficients.

In the process, logarithmic scaling has two fundamental properties: transform the mul-

tiplicative relation between filter and source signal to an additive relation, and compress the energy values at different frequencies to a more comparable scale.

After MFCC conversion, the cepstral coefficients are uncorrelated, an important feature that allows probabilistic models to dismiss correlations and so limit space complexity (if correlations were present, the covariance matrices of the Gaussian models for the feature vectors would increase quadratically with respect to the dimension of the data, see HMM Training section)

In the experiment, bash scripts combine Edinburgh Speech Tools and the HTK command *HCopy* to conveniently convert the *wav* files.

HMM training

Following the MFCC conversions, Hidden Markov Models need to be trained. HMMs are statistical generative models in which states are assigned *transition probabilities*. The reason behind HMMs as models for speech recognition follows from their characteristics; states in HMMs are independent, that is, an observation at a particular state S is independent of all other states. Transitional probabilities, instead, can depend on one or more past steps. For the HMMs here built, the transitional probability to land on a particular state at time t only depends on $t-1$ (first order HMM).

The HMM implemented can be thought of as a state machine in which states are fit Gaussian distributions and the probability to pass from one state to the other is the relative *transitional* probability of the Markov Model (see Figure 2).

The Gaussian distributions inside the states are *generative* as they can generate examples by repetitively sampling from their probability distribution.

The likelihood of observing a particular vector at time t can be computed directly from the Gaussian probability density function with :

$p(o|\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^n|\Sigma|}} exp(-\frac{1}{2}(o-\mu)^T\Sigma^{-1}(o-\mu))$,

where o is the MFCC observed vector (a frame), μ is the mean vector, $\Sigma = \begin{bmatrix} \sigma_1^2 & & \sigma_{1n} \\ & \ddots & \\ \sigma_{n1} & & \sigma_n^2 \end{bmatrix}$ is the covariance matrix for the model and Σ^{-1} is the inverse of the covariance matrix.

The diagonal entries in Σ correspond to the variances for the various dimensions of o. The remaining entries can instead be zero, thanks to the independence of MFCC features;

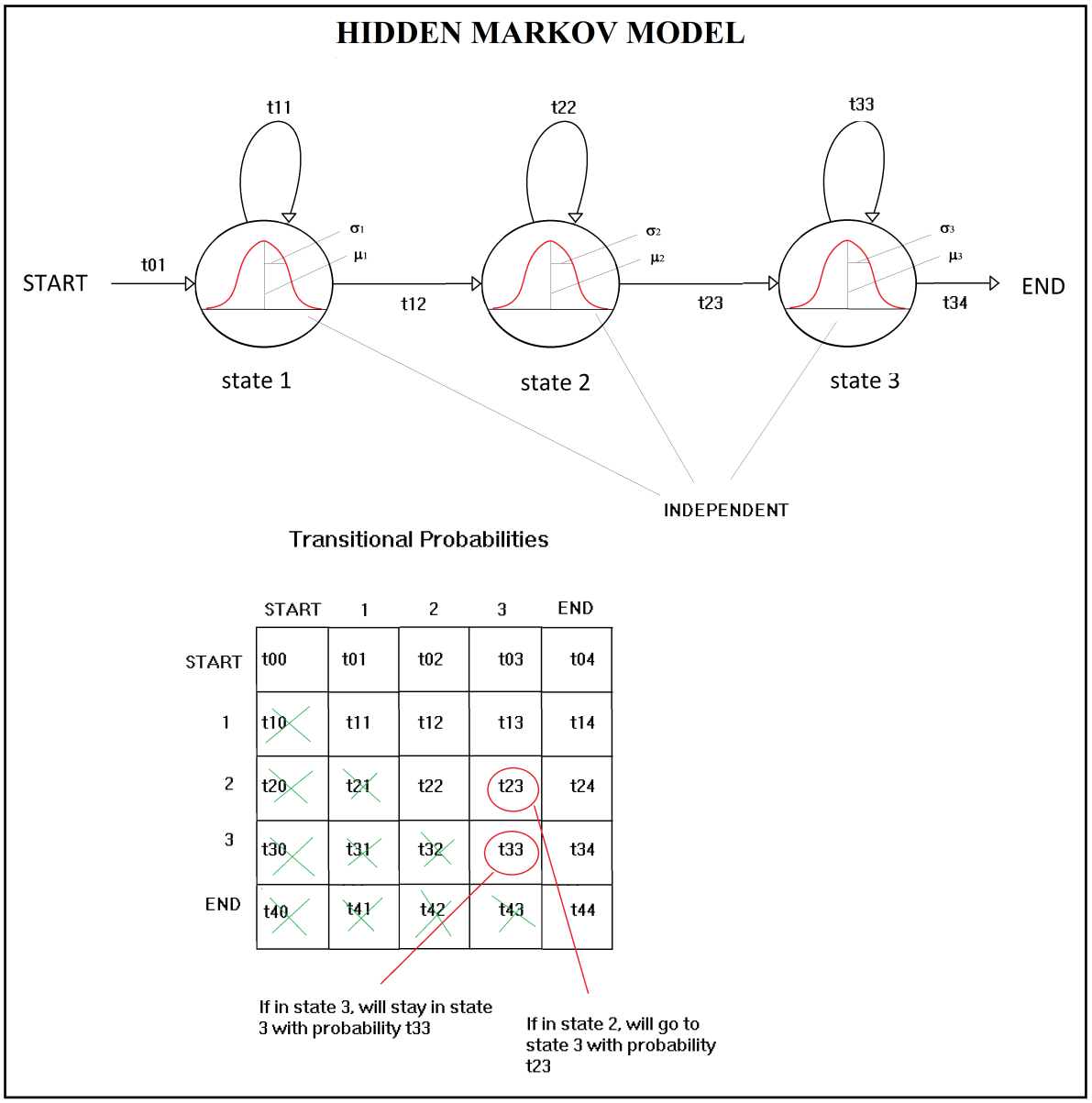


Figure 2: Hidden Markov Model as a state machine whose transition probabilities are recorded in a matrix. The states are not allowed backward transitions (like one would expect in speech), so about half entries in the matrix can be dismissed. Three of the states are fit Gaussian distributions.

the matrix can then be effectively reduced to a vector. Moreover, if the dimensions were dependent, a much bigger training set would be needed for the covariance values to be significant.

At each generation, the HMM state machine can either remain on the same state with

probability t or *transit* to another with probability $1 - t$. Transitional probabilities can then be used as duration models. When the state machine reaches the end state, a sequence of vectors has been generated (thus the generative nature of HMMs).

Training HMMs is generally a hard problem given their *hidden* properties (it is unknown which states generated which observations when a sequence is observed at training time). In the experiments each word model is initialized as an HMM with 5 states, three of which are Gaussians with $\mu = 0$ and $var = 1$.

The *HInit* command then uses the Viterbi algorithm (see System testing section) to find the most likely sequence for each observation and iteratively updates the model's parameters (μ and var for each Gaussian) according to which state is most likely to have generated which sequence, until the likelihood of the data stops increasing (see Figure 5).

After a first estimation, a more accurate algorithm is used to make the parameter values converge to better solutions by taking into considerations *all* possible sequences that could have generated O, weighting them by their likelihood and computing a weighted sum of the relevant observations to update the parameters in each state. The EM algorithm used is called the Baum-Welch algorithm and is here applied through the *HRest* command.

Grammar

To build a recognizer it is necessary to build a grammar defining the possible classes and how they can be combined to form sequences of words. For simple digit recognition, the grammar is trivial and corresponds to categorizing each instance as belonging to a class (see Figure 3). The language model defines the probability $P(W)$ of a word in the model;

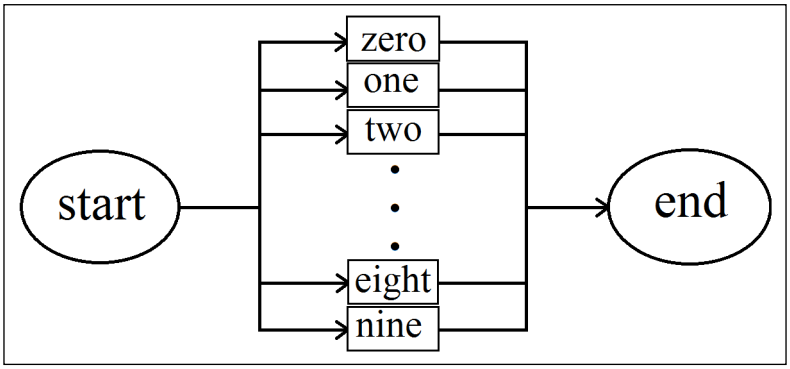


Figure 3: Model of grammar for a simple digit recognizer.

for this particular case, all word prior probabilities are equal.

System testing

To test the system, it is first necessary to classify each test utterance into one of the possible classes; for this, the probability of a word given a sequence of MFCC vector observations $P(W|O)$ needs to be computed. The generative nature of HMM allows the system to compute the probability of a vector sequence given a word $P(O|W)$. The computation, however, is too expensive to be done at recognition time, since $P(O|W)$ requires to sum over all possible ways the model for W could generate O. The Viterbi algorithm is here used to compute an approximation to the value of $P(O|W)$ as the single most likely path through the model that generates O; the HTK version of the Viterbi algorithm implements token passing, where tokens are partial paths, copied and passed through each possible sequence in the HMM state machine and pruned as they meet on the way (pruning is possible thanks to the independence assumption in HMMs which allows dismissing all but the highest probability token at any meeting point through the state machine). Once the approximated $P(O|W)$ is known, it is possible to retrieve $P(W|O)$ by Bayes rule, i.e. $P(W|O) = \frac{P(O|W)*P(W)}{P(O)}$, where O is a sequence of observations and W is a word (see Figure 4). In real computations $P(O)$ can be omitted since it is irrelevant in the choice of W to maximize the equation (it also can not be computed). The command *HVite* uses token passing through the language model (Figure 3) to predict the word where $P(W|O)$ is maximum.

Once the words for the different test utterances are predicted, the previously annotated *mlf* file containing the correct class for each test is used in the *HResults* command to built a confusion matrix from which we retrieve the prediction accuracy of the system.

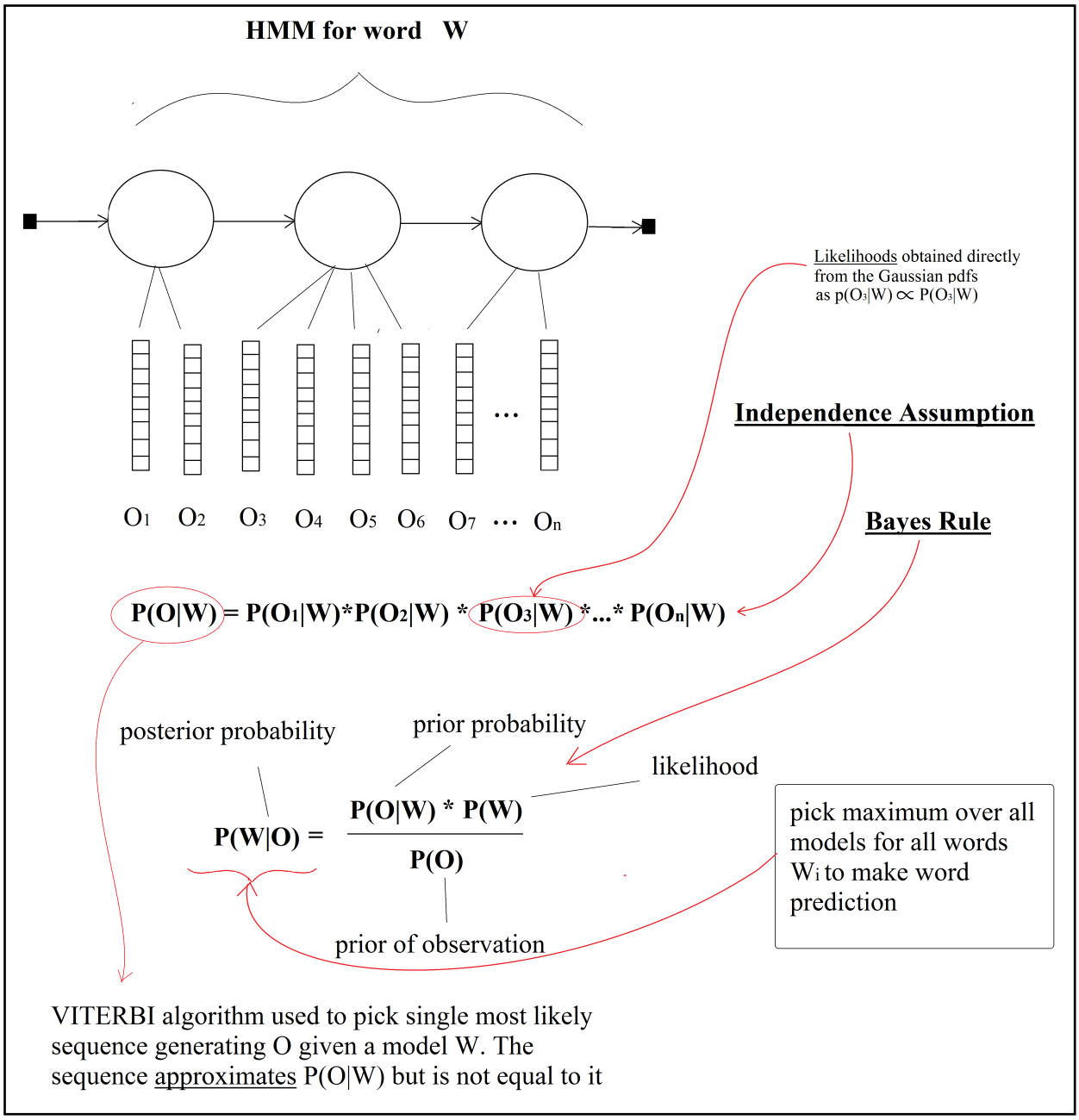


Figure 4: Example of system for word prediction. Bayes rule is used to pass from prior to posterior probabilities. The HMM which maximizes the posterior probability given a sequence defines the predicted word for the specific sequence.

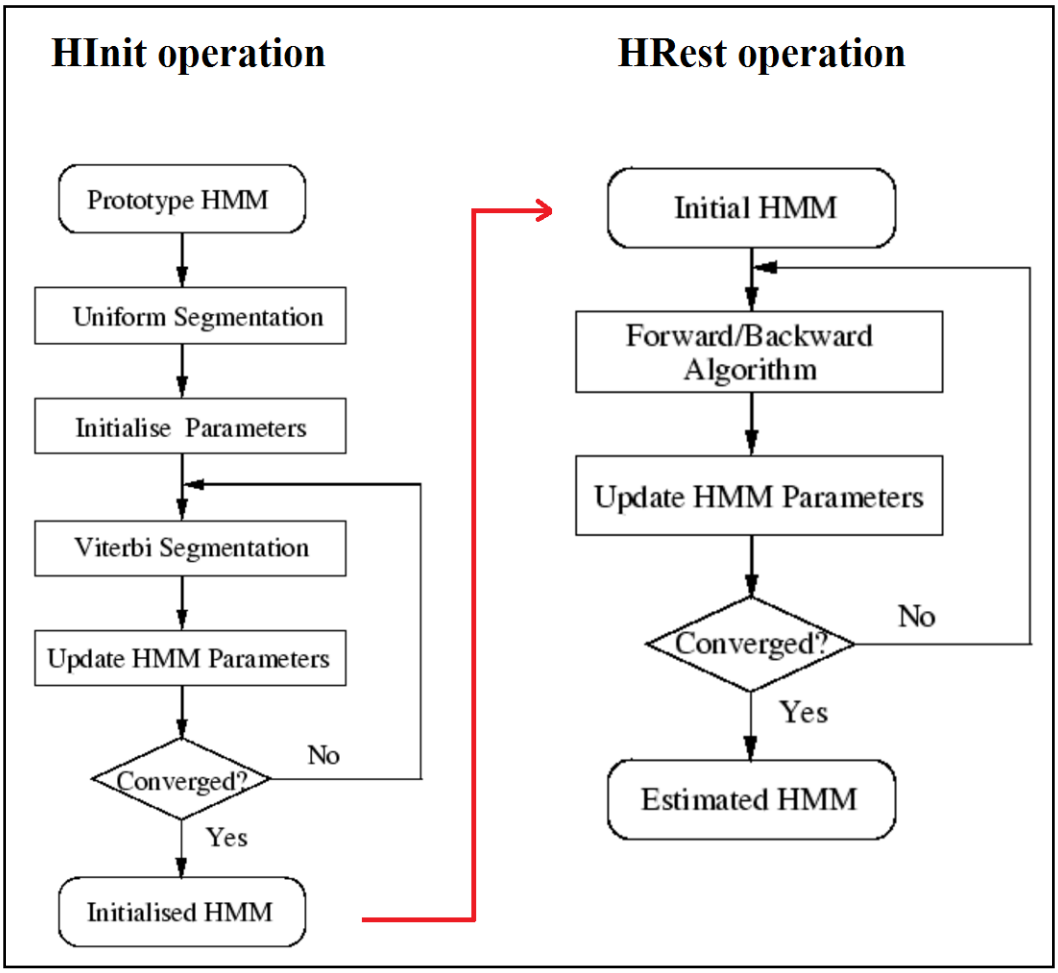


Figure 5: Flow charts of The HInit and HRest algorithms in HTK. The charts can be found in the HTK manual.

3 Extensions and results

3.1 Test on single speaker

A speaker dependent recognizer is built following the steps above described. The system is trained from a *wav* file containing seven utterances for each digit from zero to nine in random order. Once a 5 state HMM is built for each class (digit), the system is tested on a set of thirty uttered digits from the same speaker, three for each class. The predicted classes for each test utterance are matched against the actual labels and a confusion matrix is used to capture the accuracy of the classification task. The results are shown in Figure 6. The system manages to correctly classify all test instances and thus achieves 100% accuracy. A relatively small test set,

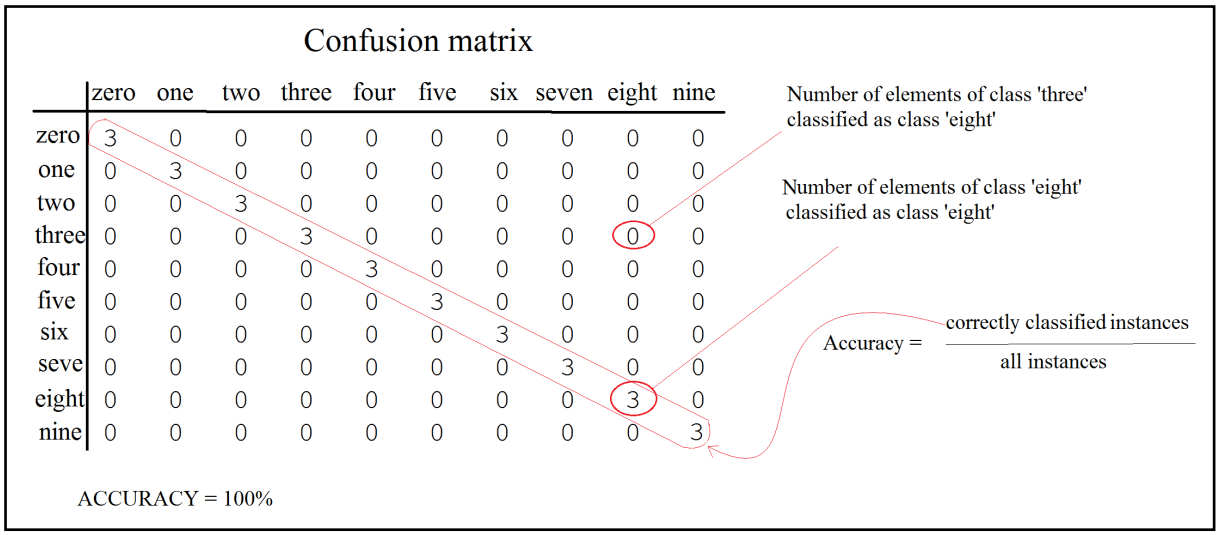


Figure 6: Confusion matrix for the speaker dependent recognizer trained and tested on a single speaker.

like the one used for the experiment, holds enough information to capture the different variations of speech for a single speaker, which utters a small number of short words like digits on a single microphone.

3.2 Test on different speakers

To take a further step into understanding the extent to which the so far built recognizer can be generalized to other speakers, the system is now tested against other speakers and the results reported in Figure 7.

The drop in performance over all speakers is non trivial (>50%). The reason behind the low accuracies lies in system training. The models *over-fit* the characteristics of the single speaker they were trained on and fail to capture more general speech properties for uttered digits. The speakers are chosen in sets of two varying only one parameter. However, from the results it is premature to assert any single parameter (e.g. gender, accent) worsens classification accuracy more than any other, given the model is only trained on a single speaker (see Speaker independent section for further experiments).

TRAINING SET			
gender	**native speaker**	**accent**	**headset**
Male	yes	Native American	logitech headset

TEST SETS				PREDICTION ACCURACY
gender	**native speaker**	**accent**	**headset**	
Male	yes	Native American	logitech headset	38.46%
Male	yes	Native American	logitech headset	27.50%
Male	no	N/A	logitech headset	36,67%
Male	no	N/A	logitech headset	56.67%
Female	yes	Native American	logitech headset	40.00%
Female	yes	Native American	logitech headset	23.33%

Figure 7: Accuracy results for the Test on different speaker example on various speakers.

3.3 Multiple speakers

To make the system more generally capture the way digits are uttered, a set of different speakers are used for both training and testing. The sets are now roughly ten times bigger than in previous experiments (See Table 1 for details). The speakers all use the same microphone to record their speech, namely 'logitech_headset', moreover, the same number of males and females are chosen. It is then reasonable to assume that the two parameters do not in any way affect the system accuracy.

The training and test set are loaded in both the *HInit* and *HRest* command through the *-S scriptfile* option where *scriptfile* is a script file containing the locations of the *MFCC* files for the different speakers. After classification, the results are matched against the *mlf* files for each speaker, containing the correct labels for the test utterances, and a confusion matrix is created and used to extract the accuracy of the system. Both the accuracy and the correctly classified percentage of words in the results amount to 95.72%. The trained HMMs manage to successfully model the different ways digits are uttered from different individuals in a relatively small set of speakers.

Table 1: Training and test set for multiple speaker extension.

Gender	Origin	Accent Type	headset
m	Northern Ireland	UK	logitech_headset
m	Southern England	UK	logitech_headset
m	Scotland	SC	logitech_headset
m	England	UK	logitech_headset
m	China	NN	logitech_headset
f	Scotland	SC	logitech_headset
f	Denmark	NN	logitech_headset
f	United States	NA	logitech_headset
f	France	NN	logitech_headset
f	Spain	NN	logitech_headset

LEGEND

UK :	*British English*	NA :	*North American*
SC :	*Scottish English*	NN :	*Non Native*
AZ :	*Australian/New Zeland*		

3.4 Speaker independent

The previously obtained results can be optimistic in accounting for the extent to which the system manages to classify digits from random users. To make testing closer to real world applications the training and test set are now obtained from different speakers (i.e. no speaker in one set is used for the other). The details of the speakers used in the experiments are tabulated in Table 2. The training set contains speech from the same speakers as in the multiple speaker extension. The test set contains speech from 5 males and 5 females all using the same microphone, namely 'logitech_headset'. The parameters are fixed to make similar assumptions as in the *multiple speakers* sections; moreover, the number of speakers in the training and test set is equal to make the results comparable to previous experiments. In the scripts, *HInit* and *HRest* do not need to be modified as opposed to *HVite* and *HRest*, which need to load the instances for the changed test set and *HResults*, the correspondent *mlf* files. The system achieves 85.53% recognition accuracy. The number, if smaller than in the multiple speaker example, results relatively high considering classification is done on word utterances from speakers never seen before.

Table 2: Training and test set for speaker independent extension.

	Gender	Origin	Accent Type	headset
TRAINING SET	m	Northern Ireland	UK	logitech_headset
	m	Southern England	UK	logitech_headset
	m	Scotland	SC	logitech_headset
	m	England	UK	logitech_headset
	m	China	NN	logitech_headset
	f	Scotland	SC	logitech_headset
	f	Denmark	NN	logitech_headset
	f	France	NN	logitech_headset
	f	Spain	NN	logitech_headset
	f	United Stataes	NA	logitech_headset
TEST SET	m	England	UK	logitech_headset
	m	/	NN	logitech_headset
	m	Southern England	UK	logitech_headset
	m	Portugal	NN	logitech_headset
	m	/	NN	logitech_headset
	f	England	UK	logitech_headset
	f	England	UK	logitech_headset
	f	Australia	AZ	logitech_headset
	f	Greece	NN	logitech_headset
	f	India	NN	logitech_headset

LEGEND

UK :	*British English*	NA :	*North American*
SC :	*Scottish English*	NN :	*Non Native*
AZ :	*Australian/New Zeland*		

A second experiment in the category is performed with the same trained system but now tested on a series of sets from different speakers, picked in pairs of two, and differing of only one parameter. The results are shown in Figure 8.

From the results it is clear how gender has a higher impact on speech recognition with respect

TEST SETS				PREDICTION ACCURACY
gender	native speaker	accent	headset	
Male	yes	Native American	logitech headset	90.00%
Male	yes	Native American	logitech headset	66.67%
Male	no	N/A	logitech headset	43.33%
Male	no	N/A	logitech headset	86.67%
Female	yes	Native American	logitech headset	50.00%
Female	yes	Native American	logitech headset	53.33%

Figure 8: Accuracy results of a system tested on various speakers. The system was trained on the training set in the *speaker independent* section.

to the accent/nationality of the speaker. The reason behind the drop in performance can be explained by the difference in pitch range (female pitch is generally higher) and the difference in the vocal tract length (usually longer in males), which translates in changes of the filter extracted with MFCCs (Simpson, 2009).

3.5 Microphone independent

Although speaker independent, both training and test set have so far been recorded with only one microphone. In real world applications, a model fitting a set of speakers recorded in a lab might be used to recognize speech recorded by a smart phone, laptop or any device with a microphone capable of recording speech. For this reason, we train the model similarly to the speaker independent speech recognizer, but this time, we test on recordings from both different speakers and microphones to the training set. We expect a drop in performance for the system as in the previous example.

The training set used is unaltered from the *speaker independent* recognizer. The test set, on the other hand, is changed. The speakers' details are tabulated in Table 3.

Similarly to the *speaker independent* experiment, to make results comparable and the assumption of gender independence hold, the test and training set are from an equal number of speakers equally divided between females and males. The models are trained like previously with the '*-S*' option in *HInit*, *HRest* and *HVite* which loads the training and test script files respectively and the *HResults* command with the specified test set *mlf*s.

The recognition accuracy of the system is 54.86%. The drop in performance for classifica-

Table 3: Training and test set for microphone independent recognizer extension.

	Gender	Origin	Accent Type	headset
TRAINING SET	m	Northern Ireland	UK	logitech_headset
	m	Southern England	UK	logitech_headset
	m	Scotland	SC	logitech_headset
	m	England	UK	logitech_headset
	m	China	NN	logitech_headset
	f	Scotland	SC	logitech_headset
	f	Denmark	NN	logitech_headset
	f	France	NN	logitech_headset
	f	Spain	NN	logitech_headset
	f	United Stataes	NA	logitech_headset
TEST SET	m	England	UK	sennheiser_headset
	m	/	NN	sony_pulse_headset
	m	China	NN	sennheiser_headset
	m	Greece	NN	macbookpro_builtin
	m	England	UK	lenovo_laptop
	f	/	NA	microsoft_headset
	f	/	NA	sennheiser_headset
	f	China	NN	rode_k2
	f	Southern England	UK	rode_k2
	f	Denmark	NN	CAD_U37_USB

LEGEND

UK :	*British English*	NA :	*North American*
SC :	*Scottish English*	NN :	*Non Native*
AZ :	*Australian/New Zeland*		

tions on words uttered from unknown speakers over different microphones is significant. The microphone parameter has a higher impact on accuracy than most other parameters so far explored. The reason behind the low performance lies in the speech quality captured by different microphones and, in particular, the microphone used in training. The 'logitek_headset' is a

medium quality microphone whose recorded speech might not be detailed enough to build models that successfully generalize digit utterances. Moreover, part of the speech in the test set is recorded with built in microphones, subject to capture room refractions and other noises from the speaker's surroundings; others, like the headset microphones, are instead much closer to the source.

Another reason for the generic low accuracy, may be due to the amount of training data supplied to the system.

To support the claims, a system is trained on a set of utterances from 20 speakers, 10 males and 10 females, twice as big as in previous examples, all using a stand-mounted Sennheiser microphone with much better recording quality. The system is tested on the same test set as in the previous experiment. The system accuracy after testing is of 77.14%. The results support the hypothesis.

3.6 Different states

To test the impact in speech recognition of the number of HMM states for the different word models, five systems are trained on a set of utterances from 20 speakers, as described in the *microphone independent* section, and tested on the test set tabulated in Table 3. The systems increasingly change the number of HMM states to model the different words, the results are shown in Figure 9.

For a digit recognizer it is typically the case to want each phoneme modeled by an HMM state. Some phonemes, in fact, are more *stretchy* then others, and their length in an utterance can vary more; the lengths can be modeled by the transitional probabilities which can loop through a state for longer or transit to another sooner.

As the number of states changes, keeping all other parameters in the system fixed, the accuracy increases to a maximum before lowering again. The result suggests how accuracy can be improved when digits are modeled at a sub phonemic level, where the invariant properties of the utterances hold. When the model is fine grained enough to capture the invariant properties, increasing the number of states can have the opposite effect, as the model over-fits the data and the accuracy increasingly lower.

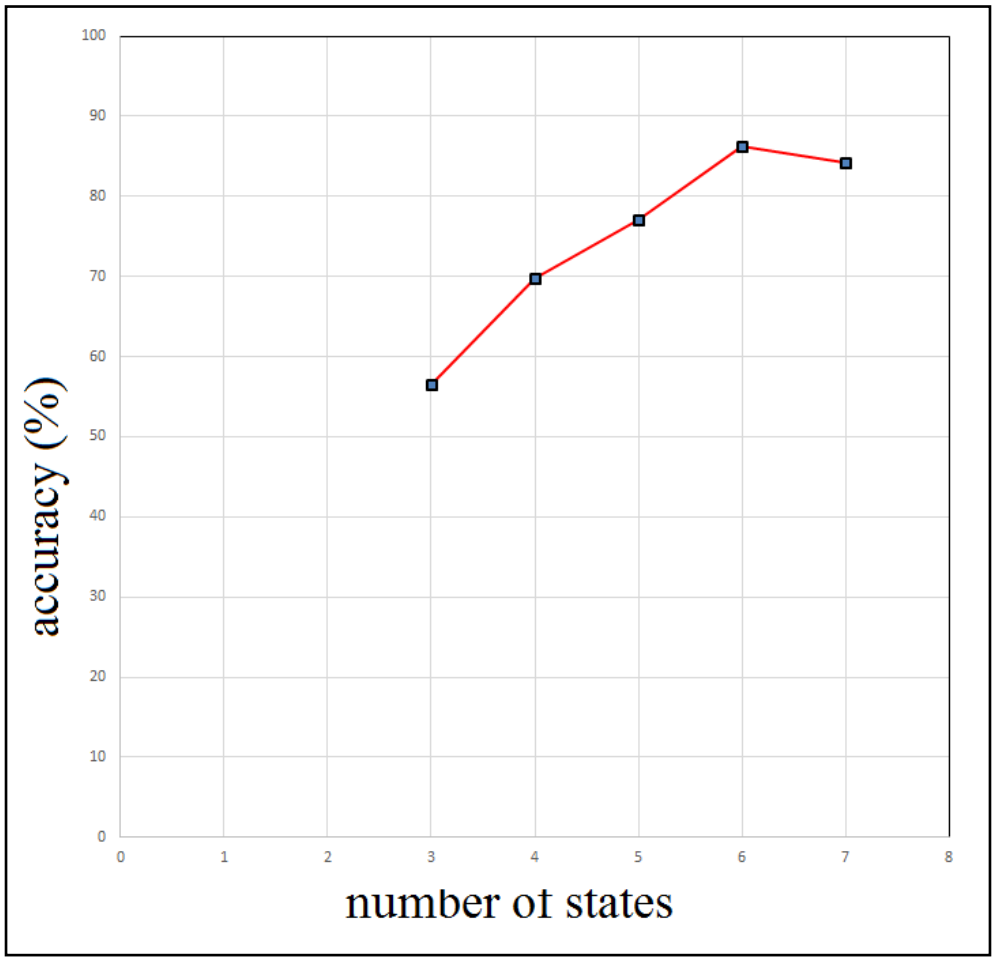


Figure 9: Plot of the number of states in the HMM models of a speech recognizer against the percentage accuracy of the system when tested. The number of states in the HMM account for two *dummy* states, a "start" sate and an "end" state (e.g. the number of states with Gaussian distributions in a 5 state HMM is 3).

3.7 Digit sequence recognizer

To build a digit sequence recognizer the laid out process, if similar, needs to be modified. First, it is necessary for the models to include *junk* as a class, so to recognize silences between digits. This, is done by changing both the grammar and the dictionary file used by HTK to include the new class. The new grammar, rather than being simplistic, includes observed knowledge about digit sequences to improve recognition accuracy and is built in a human readable syntax (see Figure 10). Later, grammar conversion is done with the *HParse grammar parsed_grammar* command which builds a network ready for HTK to process.

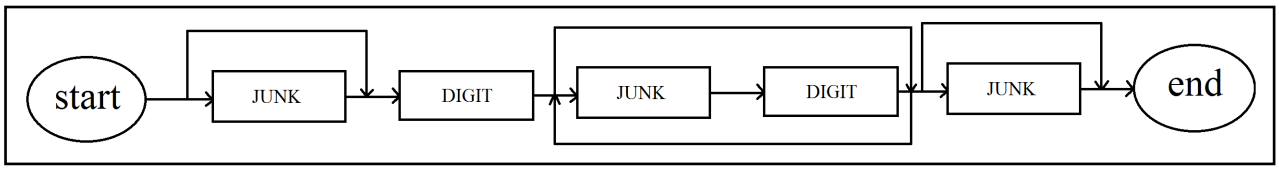
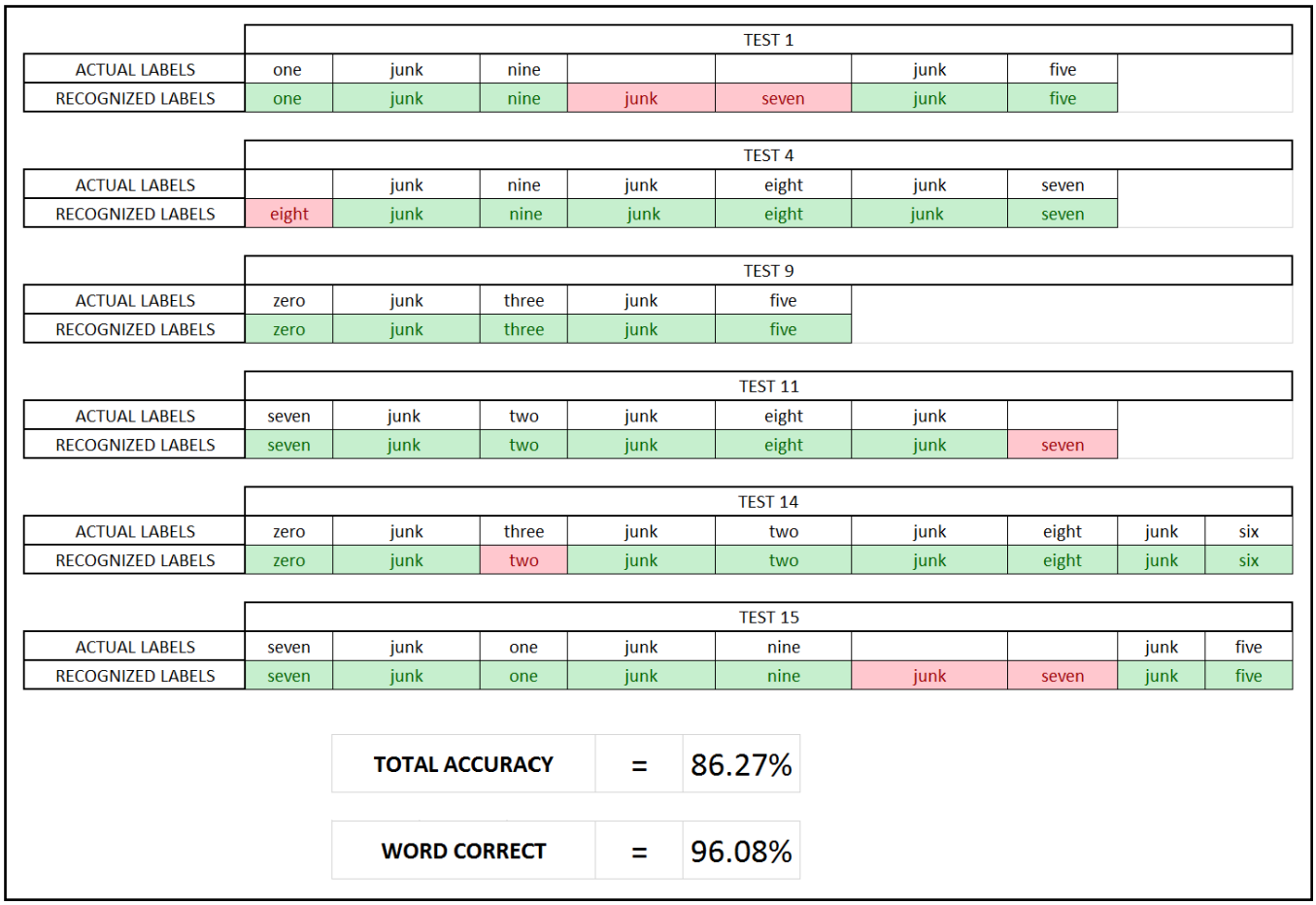


Figure 10: Grammar for the digit sequence recognizer. Part of the embedded domain knowledge features digits appearing within *junk*s and starting or ending with silences.

The training of the model is done with *HInit* and *HRest*, like previously discussed, on the speech of a single user. For testing, however, a set of 16 test instances is created. The instances contain speech from the same speaker as in testing which utters digits in a random order; in each instance a variable number of digits is uttered.
The test files are converted to MFCCs with the *HCopy* command and later used in *HVite* for testing the system.
A common problem with recognizing sequential speech are insertions and deletions. The system, in fact, could consider infinitely long sequences of digits and eventually find a matching substring for the uttered test. This is regulated with the *-p* option of the *HVite* command which allows to penalize insertions. Figure 11 shows some of the test results when the system is tested against the test set.
The recognizer performs overall well, problems with the models include the insertion of both a *junk* and a *digit* label when silences stretch for too long (e.g. TEST 1 and TEST 15; see Figure 11 and Figure 10).

4 Summary and Conclusion

A simple speaker dependent digit recognizer is built with the Hidden Markov Model Toolkit, the various implemented steps and theory behind them is explored. The extent to which the system generalized digit recognition is tested and discussed. After rebuilding the system and training it on multiple speakers with different fixed parameters (e.g. male/female ratio, microphone used), various tests show how gender and microphone type (low quality at training time) cause the



	TEST 1								
ACTUAL LABELS	one	junk	nine			junk	five		
RECOGNIZED LABELS	one	junk	nine	junk	seven	junk	five		

	TEST 4						
ACTUAL LABELS		junk	nine	junk	eight	junk	seven
RECOGNIZED LABELS	eight	junk	nine	junk	eight	junk	seven

	TEST 9				
ACTUAL LABELS	zero	junk	three	junk	five
RECOGNIZED LABELS	zero	junk	three	junk	five

	TEST 11						
ACTUAL LABELS	seven	junk	two	junk	eight	junk	
RECOGNIZED LABELS	seven	junk	two	junk	eight	junk	seven

	TEST 14								
ACTUAL LABELS	zero	junk	three	junk	two	junk	eight	junk	six
RECOGNIZED LABELS	zero	junk	two	junk	two	junk	eight	junk	six

	TEST 15								
ACTUAL LABELS	seven	junk	one	junk	nine			junk	five
RECOGNIZED LABELS	seven	junk	one	junk	nine	junk	seven	junk	five

TOTAL ACCURACY	=	86.27%
WORD CORRECT	=	96.08%

Figure 11: Subset of results for the digit sequence recognizer.

most drop in recognition performance. Accent, if to a different scale, also affects recognition. Ultimately, a digit sequence recognizer is built, tested against a set of 16 digit sequence utterances and shown to be capable of dealing with variable sequence lengths. Some parameters manipulation is needed to deal with problems like insertions and deletions.

References

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2 edition, May 2008.

Simon King. Pack 3 slides: Speech recognition, 2015.

Adrian P. Simpson. Phonetic differences between male and female speech. In *Language and Linguistics Compass*, volume 3, pages 621–640. Blackwell Publishing Ltd, March 2009.

John N Holmes and Wendy Holmes. *Speech Synthesis and Recognition*. CRC Press, 2 edition, 2001.

M.A.Anusuya and S.K.Kati. Speech recognition by machine: A review. *International Journal of Computer Science and Information Security*, 6(3), 2009.

word count: 3992