PRONUNCIATION SUPPORT
FOR ARABIC LEARNERS

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2015

By
Majed Alsabaan
School of Computer Science
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>12</td>
</tr>
<tr>
<td>Declaration</td>
<td>14</td>
</tr>
<tr>
<td>Copyright</td>
<td>15</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>16</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>18</td>
</tr>
<tr>
<td>1.1 Research Overview</td>
<td>19</td>
</tr>
<tr>
<td>1.2 Organisation of the Thesis</td>
<td>20</td>
</tr>
<tr>
<td>1.3 Research Questions and Research Tasks</td>
<td>22</td>
</tr>
<tr>
<td>1.4 Research Methodology</td>
<td>24</td>
</tr>
<tr>
<td>1.5 Research Contribution</td>
<td>25</td>
</tr>
<tr>
<td>1.6 Summary</td>
<td>25</td>
</tr>
<tr>
<td>2 Background and Literature Review</td>
<td>26</td>
</tr>
<tr>
<td>2.1 Automatic Speech Recognition (ASR)</td>
<td>26</td>
</tr>
<tr>
<td>2.1.1 Introduction</td>
<td>26</td>
</tr>
<tr>
<td>2.1.2 Hidden Markov Model (HMM)</td>
<td>28</td>
</tr>
<tr>
<td>2.2 The Arabic Language</td>
<td>31</td>
</tr>
<tr>
<td>2.2.1 Introduction</td>
<td>31</td>
</tr>
<tr>
<td>2.2.2 Arabic ASR</td>
<td>32</td>
</tr>
<tr>
<td>2.3 Pedagogy of Learning</td>
<td>34</td>
</tr>
<tr>
<td>2.3.1 Aptitude and Motivation</td>
<td>34</td>
</tr>
<tr>
<td>2.3.2 Thinking Tools</td>
<td>35</td>
</tr>
<tr>
<td>2.3.3 Conditions for Learning Language</td>
<td>35</td>
</tr>
<tr>
<td>2.3.4 Assessment in Learning</td>
<td>36</td>
</tr>
</tbody>
</table>
2.4 CALL tools in Language Learning ........................................ 38
  2.4.1 History of CALL ...................................................... 38
  2.4.2 Importance of CALL ................................................ 40
  2.4.3 Detailed Example of a CALL tool - Baldi ...................... 43
    2.4.3.1 How does Baldi work? ....................................... 45
  2.4.4 Graphical Simulation of the Vocal Tract ....................... 46
  2.5 Speech Synthesis ..................................................... 48
  2.6 Conclusion .................................................................. 50

3 Architecture of the CALL tool ............................................. 51
  3.1 System Architecture ...................................................... 51
  3.2 System Integration ........................................................ 51
    3.2.1 Identifying Mispronunciation by the HTK .................. 53
    3.2.2 Animating the Vocal Tract ...................................... 53
    3.2.3 Synthesising Utterances ........................................ 53
    3.2.4 Instruction Feature ............................................... 53
  3.3 Language Lab Version ................................................... 56
  3.4 Conclusion .................................................................. 56

4 HTK Experiments ............................................................... 57
  4.1 Using the HTK .............................................................. 59
    4.1.1 The HTK training steps ......................................... 59
      4.1.1.1 Data Preparation ........................................... 60
      4.1.1.2 Training the HTK ........................................... 65
      4.1.1.3 Recogniser Evaluation .................................... 67
    4.2 Experiments ............................................................. 69
      4.2.1 Arabic Unique Consonants .................................... 70
      4.2.2 Training Data and Platform .................................. 73
      4.2.3 Designed System ................................................ 75
      4.2.4 Experiment 1: Word Level vs. Phoneme Level .......... 76
        4.2.4.1 Experimental Design .................................... 76
        4.2.4.2 Results ...................................................... 78
        4.2.4.3 Conclusion ................................................ 79
      4.2.5 Experiment 2: The effect of Gender ....................... 82
        4.2.5.1 Experimental Design .................................... 82
        4.2.5.2 Results ...................................................... 83
B Python Script for the HTK training

C Dynamic Time Warping

D Morphing Code

E Top-Level System Integration

F Buckwalter Arabic transliteration

G Modified Buckwalter Arabic transliteration

H IPA for Arabic

I Modified IPA for Arabic

J Arabic Minimal Pairs

K Behavioural Traces
   K.1 User behaviours of language lab version .................... 161
   K.2 User behaviours of animation version ....................... 165
   K.3 User behaviours of synthesis version ....................... 168
   K.4 User behaviours of instruction version ..................... 171
   K.5 User behaviours of full version ........................... 174

Word Count: 43,777
List of Tables

4.1 Emphatic and non-emphatic sounds . . . . . . . . . . . . . . . . . 71
4.2 Arabic minimal pairs . . . . . . . . . . . . . . . . . . . . . . . . 73
4.3 Confusion matrix for emphatic sounds . . . . . . . . . . . . . . . 79
4.4 Experiments' summary . . . . . . . . . . . . . . . . . . . . . . . . 86

7.1 Classroom results . . . . . . . . . . . . . . . . . . . . . . . . . . 109
7.2 Individual student improvement . . . . . . . . . . . . . . . . . . 110
7.3 p-value for each version vs. language lab . . . . . . . . . . . . . . 110

F.1 Buckwalter Arabic transliteration table . . . . . . . . . . . . . . . 153
G.1 Modified Buckwalter Arabic transliteration table . . . . . . . . . . 155
H.1 IPA symbols for Arabic . . . . . . . . . . . . . . . . . . . . . . . . 157
I.1 Modified IPA symbols for Arabic . . . . . . . . . . . . . . . . . . 159
J.1 Words of Arabic Minimal Pairs . . . . . . . . . . . . . . . . . . . . 160
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of HMM</td>
</tr>
<tr>
<td>2.2</td>
<td>Wave signal of the word ‘MIN’</td>
</tr>
<tr>
<td>2.3</td>
<td>Baldi’s wire frame model</td>
</tr>
<tr>
<td>2.4</td>
<td>Inside view of Baldi</td>
</tr>
<tr>
<td>2.5</td>
<td>Sagittal view</td>
</tr>
<tr>
<td>2.6</td>
<td>The vocal tract</td>
</tr>
<tr>
<td>3.1</td>
<td>System architecture</td>
</tr>
<tr>
<td>3.2</td>
<td>Articulatory advice</td>
</tr>
<tr>
<td>4.1</td>
<td>HTK experiments</td>
</tr>
<tr>
<td>4.2</td>
<td>The HTK training steps</td>
</tr>
<tr>
<td>4.3</td>
<td>Step 1: Data preparation</td>
</tr>
<tr>
<td>4.4</td>
<td>Prompts file</td>
</tr>
<tr>
<td>4.5</td>
<td>TestPrompts file</td>
</tr>
<tr>
<td>4.6</td>
<td>SpeechRecord tool</td>
</tr>
<tr>
<td>4.7</td>
<td>The contents of codetr.scp</td>
</tr>
<tr>
<td>4.8</td>
<td>Grammar file</td>
</tr>
<tr>
<td>4.9</td>
<td>Pronunciation dictionary file</td>
</tr>
<tr>
<td>4.10</td>
<td>Using HDMan to generate the dictionary file</td>
</tr>
<tr>
<td>4.11</td>
<td>Step 2: Running the HTK training script</td>
</tr>
<tr>
<td>4.12</td>
<td>Tongue position during the articulation of the emphatic (C&lt;sup&gt;3&lt;/sup&gt;) and non-emphatic consonants (C)</td>
</tr>
<tr>
<td>4.13</td>
<td>Spectrogram representation of the word ‘d&lt;sup&gt;3&lt;/sup&gt;arb’</td>
</tr>
<tr>
<td>4.14</td>
<td>Spectrogram representation of the word ‘darb’</td>
</tr>
<tr>
<td>4.15</td>
<td>Grammar file - word level</td>
</tr>
<tr>
<td>4.16</td>
<td>Pronunciation dictionary file - word level</td>
</tr>
<tr>
<td>4.17</td>
<td>Grammar file - phoneme level</td>
</tr>
</tbody>
</table>
K.2 User behaviour in the language lab version ........................................ 162
K.3 User behaviour in the language lab version ........................................ 162
K.4 User behaviour in the language lab version ........................................ 162
K.5 User behaviour in the language lab version ........................................ 163
K.6 User behaviour in the language lab version ........................................ 163
K.7 User behaviour in the language lab version ........................................ 163
K.8 User behaviour in the language lab version ........................................ 164
K.9 User behaviour in the animation version ............................................. 165
K.10 User behaviour in the animation version ............................................ 165
K.11 User behaviour in the animation version ............................................ 165
K.12 User behaviour in the animation version ............................................ 166
K.13 User behaviour in the animation version ............................................ 166
K.14 User behaviour in the animation version ............................................ 166
K.15 User behaviour in the animation version ............................................ 167
K.16 User behaviour in the animation version ............................................ 167
K.17 User behaviour in the synthesis version ............................................. 168
K.18 User behaviour in the synthesis version ............................................. 168
K.19 User behaviour in the synthesis version ............................................. 168
K.20 User behaviour in the synthesis version ............................................. 169
K.21 User behaviour in the synthesis version ............................................. 169
K.22 User behaviour in the synthesis version ............................................. 169
K.23 User behaviour in the synthesis version ............................................. 170
K.24 User behaviour in the synthesis version ............................................. 170
K.25 User behaviour in the instruction version ......................................... 171
K.26 User behaviour in the instruction version ......................................... 171
K.27 User behaviour in the instruction version ......................................... 172
K.28 User behaviour in the instruction version ......................................... 172
K.29 User behaviour in the instruction version ......................................... 172
K.30 User behaviour in the instruction version ......................................... 173
K.31 User behaviour in the instruction version ......................................... 173
K.32 User behaviour in the instruction version ......................................... 173
K.33 User behaviour in the full version .................................................... 174
K.34 User behaviour in the full version .................................................... 174
K.35 User behaviour in the full version .................................................... 175
K.36 User behaviour in the full version .................................................... 175
List of Abbreviations

AI - Artificial Intelligence
ASR - Automatic Speech Recognition
CALI - Computer Assisted Language Instruction
CALL - Computer-Assisted Language Learning
DSP - Digital Signal Processing
DTW - Dynamic Time Warping
EM - Expectation Maximisation
HMM - Hidden Markov Model
HTK - Hidden Markov model ToolKit
ICT - Information and Communication Technology
IPA - International Phonetic Alphabet
L2 - Second Language
MALL - Mobile Assisted Learning Language
MALU - Mobile Assisted Learning Use
MFCCs - Mel-Frequency Cepstral Coefficients
MLAT - Modern Languages Aptitude Test
MOO - Multi-user domain, Object-Oriented
MSA - Modern Standard Arabic
NLP - Natural Language Processing
PCs - Personal Computers
PDAs - Personal Digital Assistants
SAMPA - Speech Assessment Methods Phonetic Alphabet
SR - Speech Recognition
TELL - Technology Enhanced Language Learning
TESOL - Teaching English to Speakers of Other Languages
TTS - Text-To-Speech
Abstract

PRONUNCIATION SUPPORT
FOR ARABIC LEARNERS
Majed Alsabaan
A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy, 2015

The aim of the thesis is to find out whether providing feedback to Arabic language learners will help them improve their pronunciation, particularly of words involving sounds that are not distinguished in their native languages. In addition, it aims to find out, if possible, what type of feedback will be most helpful. In order to achieve this aim, we developed a computational tool with a number of component sub tools. These tools involve the implementation of several substantial pieces of software.

The first task was to ensure the system we were building could distinguish between the more challenging sounds when they were produced by a native speaker, since without that it will not be possible to classify learners’ attempts at these sounds. To this end, a number of experiments were carried out with the hidden Markov model toolkit (the HTK), a well known speech recognition toolkit, in order to ensure that it can distinguish between the confusable sounds, i.e. the ones that people have difficulty with.

The developed computational tool analyses the differences between the user’s pronunciation and that of a native speaker by using grammar of minimal pairs, where each utterance is treated as coming from a family of similar words. This provides the ability to categorise learners’ errors - if someone is trying to say cat and the recogniser thinks they have said cad then it is likely that they are voicing the final consonant when it should be unvoiced. Extensive testing shows that the system can reliably distinguish such minimal pairs when they are produced by a native speaker, and that this approach does provide effective diagnostic information about errors.

The tool provides feedback in three different sub-tools: as an animation of
the vocal tract, as a synthesised version of the target utterance, and as a set of written instructions. The tool was evaluated by placing it in a classroom setting and asking 50 Arabic students to use the different versions of the tool. Each student had a thirty minute session with the tool, working their way through a set of pronunciation exercises at their own pace. The results of this group showed that their pronunciation does improve over the course of the session, though it was not possible to determine whether the improvement is sustained over an extended period. The evaluation was done from three points of view: quantitative analysis, qualitative analysis, and using a questionnaire. Firstly, the quantitative analysis gives raw numbers telling whether a learner had improved their pronunciation or not. Secondly, the qualitative analysis shows a behaviour pattern of what a learner did and how they used the tool. Thirdly, the questionnaire gives feedback from learners and their comments about the tool.

We found that providing feedback does appear to help Arabic language learners, but we did not have enough data to see which form of feedback is most helpful. However, we provided an informative analysis of behaviour patterns to see how Arabic students used the tool and interacted with it, which could be useful for more data analysis.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocInfo.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University’s policy on presentation of Theses.
Acknowledgements

After thanking the Almighty God for his endless blessings in this effort all the way to its completion, I would also like to express my heartfelt thanks to a number of people who provided me with guidance and support throughout this journey.

First and foremost, I would like to express my sincere thanks to my supervisor Prof. Allan Ramsay. I appreciate all his contributions of generous time, ideas, and scholarly advice to make my PhD experience productive and stimulating. His continuous guidance and invaluable suggestions have shaped the main structure of this work. I am also thankful for his continuous help to overcome the challenges that arose on this journey.

I would also like to thank Dr. Faisal Almeshary who is my line manager in the National Center for Assessment in Higher Education (Qiyas) in Saudi Arabia for his continuous support and consistent encouragement. Also, I wish to thank the staff of the University of Manchester for their help in providing me with an academic friendly environment. I appreciate their academic support and the facilities provided to carry out the research work at the university.

I want to say ‘Thank you’ to the members of my research group, especially Iman, Sardar, and Maytham who generously shared with me their knowledge and experience. I acknowledge their personal and scholarly interactions, and their suggestions at various points of my research.

The results reported in this thesis would not have been possible without the cooperation of those who participated in the data collection and classroom experiments. To all of them, I would like to extend my sincere gratitude for their time spent helping me.

I owe a lot to my parents, who encouraged and helped me at every stage of my personal and academic life, and wished to see this achievement come true. I want to say ‘Thank you’ for always being supportive and praying for me. I
would like also to extend my gratitude to my brothers and sisters for their encouragement, especially my brother Mazen, who completed his PhD in computer engineering from Waterloo University last year, for sharing his academic experience in Canada.

Last but not least, I would like to say ‘Thank you’ coming from the bottom of my heart to my beloved wife Maymunah. Thank you for being supportive and understanding of my self-imposed isolation. I really appreciate all the sacrifices she made to follow me all the way during this PhD pursuit. And to my three lovely children, Abdulrahman, Norah and Albaraa, thank you for always making me smile and for putting up with me when I had to work on my thesis instead of playing with you. Thank you.
Chapter 1

Introduction

Nowadays, learning a foreign language is an essential activity for many people. People may be interested in learning a foreign language for their business (i.e., the increasing globalisation of world trade) and for receiving and dissemination of information without borders. The requirements for understanding languages and different cultures are becoming more and more important due to the world-wide communication [NOC+09].

Proficiency in a foreign language is based on four different skills, namely reading, listening, writing, and speaking. There are a number of aspects to speaking a language well, one of which is pronouncing words correctly. Thàn has suggested that this aspect (i.e., the pronunciation aspect) is the most important one [Thà11]. We are particularly interested in pronunciation skill of Arabic as a foreign language. The increasing demand for learning Arabic is due to the highly mobile society. Therefore, something must be done to help these people, and our work is aimed at teaching non-native Arabic speakers how to improve their pronunciation.

During childhood, the pronunciation patterns of a mother language become firmly established in both a physiological sense and a psychological sense [Pen99]. After a certain point of childhood development, it becomes difficult to change one’s pronunciation patterns. Adult language learners, who are our target in this research on learning Arabic, generally reach a point at which they will hardly be able to improve their pronunciation without explicit instruction [Pen98]. This explicit instruction can come from either a human teacher or a machine through a
CALL (computer-assisted language learning\textsuperscript{[1]} tool. So, with explicit instruction which can be provided by a CALL tool, the sound pattern of one’s native language can be altered and the pronunciation of learners can be improved.

We believe that Speech Recognition (SR) technology can be used to help people who are learning to pronounce words in a foreign language such as Arabic. We used this technology to distinguish between confusable sounds in Arabic by conducting a number of experiments with the hidden Markov model toolkit (HTK \cite{YEG06}). In the experiments, we used isolated words for the training samples. These samples were chosen by using the minimal pair technique. This technique helps learners distinguish between similar and problematic sounds in the target language (i.e., Arabic) through listening discrimination and spoken practice. Extensive testing shows that the SR system can reliably distinguish such minimal pairs when they are produced by a native speaker, and that this approach does provide effective diagnostic information about errors.

The above facts have motivated us to work on developing a CALL tool in order to support the pronunciation of the growing number of non-native Arabic learners. Our CALL tool can be seen as computer-aided pronunciation which offers a medium for adult Arabic learners to access their own and native pronunciation in order to improve theirs.

\subsection*{1.1 Research Overview}

This research is aimed at teaching non-native Arabic speakers how to make the more challenging sounds that make up Arabic (i.e., help them to sound more like native speakers) by giving them three forms of feedback:

\begin{itemize}
\item An animation of the vocal tract. The learner is given a graphical representation of both the way the sounds are articulated, and the way the sound is produced correctly.
\item Synthesised speech. This is a second source of feedback to learners. They can play their voice, listen to a synthesis version of what they said, and listen to a correctly synthesis version.
\end{itemize}

\textsuperscript{1}Also known as computer-assisted instruction (CAI), computer-aided instruction (CAI), or computer-aided language learning (CALL)
• A written instruction. This is the third source of feedback to the learner. A written description of how the learner can pronounce the intended phoneme is displayed.

This project used a speech recogniser to identify the properties of speech signals for both native and non-native speakers. The speech recogniser was trained to recognise phonemes of the input speech in order to obtain a phonetic analysis which was used to give feedback to the learner. In this way, we can provide feedback about the acoustic data which we hope will enable learners to adjust their pronunciation. What is novel about the current research is (i) that we will be applying SR inside a CALL system for Arabic, for which we are concentrating on certain sounds which occur in Arabic but not in other languages. These sounds are more difficult for the speech recogniser (i.e., the HTK) to distinguish than the sounds that are more widely used\footnote{See Chapter 4 for a discussion of difficult aspects relating to emphatic sounds in Arabic}, (ii) we will be giving multiple types of feedback to the learners and allowing them to choose between them, and (iii) because we are providing multiple forms of feedback, we can evaluate the comparative effectiveness of each of them.

1.2 Organisation of the Thesis

The thesis is organised into eight chapters in addition to the appendices. The organisation of chapters is as follows:

Chapter 1. The first chapter contains the introduction and the aims of the research. It provides an overview of the research and the organisation of the thesis.

Chapter 2. This chapter contains the background and literature review of the research. It outlines the critical points of current knowledge and findings of the research topic. It briefly presents the need for speech recognition and the important tools that are used in this field. It gives a background on features of Arabic as well as its challenges and difficulties in terms of Automatic Speech Recognition (ASR). This chapter also introduces the importance of language learning using a computer agent for teaching non-native speakers.
1.2. ORGANISATION OF THE THESIS

Chapter 3. This chapter presents the architecture of our CALL tool. This tool involves a substantial amount of machinery, from driving the speech recogniser to animating the images of the vocal tract, synthesising utterances, and displaying an explanatory text. The integration of all this machinery is itself a challenging task which will be explained in this chapter.

Chapter 4. The aim of this chapter is to address the work done with the speech recogniser. This chapter shows several different experiments carried out using a speech recogniser called the Hidden Markov Model ToolKit (HTK). A brief discussion will be given of the HTK training steps and data files it uses.

Chapter 5. This chapter gives an explanation of the animation version of our CALL tool. It presents what we did to achieve the animation of the vocal tract. This includes drawing the vocal tract, performing the morphing from one drawing to another, and associating the phone sequences to their corresponding articulatory positions.

Chapter 6. This chapter provides an interpretation of the synthesis version of our CALL tool. It shows the tools that were integrated to acquire a clear voice as feedback from our tool. It highlights the diphone-based synthesiser which is used for the synthesis, and also points out some experiments for improving the naturalness of the speech.

Chapter 7. This chapter shows our classroom experiments using our CALL tool with non-native students. It presents the evaluation of our CALL tool through a usability study in which we gave people three forms of feedback and asked them (questionnaire) about which of them is better. The chapter also provides raw scores which show an improvement in learner pronunciation. Moreover, the chapter shows the behaviour patterns of what students did and how they used the tool.

Chapter 8. This chapter summarises our conclusions and what we suggest for future work to those who are interested in teaching non-native speakers how to sound like native speakers.
1.3 Research Questions and Research Tasks

The principle aim of this PhD project is to help non-native Arabic learners improve their Arabic pronunciation by developing a CALL tool, which will provide them with different types of feedback about their pronunciation errors. The two major research questions that this research attempted to answer were:

**RQ1** Does such a tool contribute to improving Arabic learners’ pronunciation?

**RQ2** If so, which form of feedback is the most effective for helping Arabic learners with their pronunciation?

The following research tasks are targeted to answer these questions and help achieve the principal aim:

**RT1** To ensure that the ASR system we are using can be made accurate enough to support the principal aim

**RT2** To design and implement an animation of the vocal tract (i.e., the animation tool)

**RT3** To design and implement a tool for synthesising what the speaker’s utterance sounds like (i.e., the synthesis tool)

**RT4** To carry out classroom evaluation of the tool

**RT1** We used SR technology and conducted a series of experiments with a standard speech recogniser (i.e., the HTK) in order to see how well it deals with the most difficult sounds in Arabic and to ensure that the recognition accuracy is reliable enough to support our CALL tool. We provided the learners with a number of pronunciation tasks. Then, we analysed their pronunciation and diagnosed errors by using the HTK. They have a number of choices after producing their pronunciation: they can repeat their utterance, go to the next utterance, or access a kind of feedback, and after each action they still can be provided with the full range of these choices.

During the course of the project, three tools for providing such feedback were developed. We will refer to these tools in the sequence as: animation tool, synthesis tool, and instruction tool.
1.3. RESEARCH QUESTIONS AND RESEARCH TASKS

**RT2** The first tool (i.e., the animation tool) aims to give learners a graphical representation of what they did when they pronounced a word and what they should have done in order to pronounce it correctly. So, this representation shows them the difference between their articulation and native speaker articulation. This tool is developed to be one source of feedback to Arabic learners by drawing the vocal tract, performing the morphing from one drawing to another, and integrating this animation with the running of the HTK.

**RT3** The second tool (i.e., synthesis tool) is a second source of feedback, which provides learners with a synthesised voice of the way they say something and the way they should say it. Therefore, the learners can differentiate between these two voices by listening to them carefully. Many experiments were carried out to improve the naturalness of the speech. The point of producing a synthesised version of the learner’s input even though we have the real input is that if the learners hear a recording of their own utterance and someone else’s, they might be taken by differences that are not the differences we want them to pay attention to. However, if we synthesise theirs as well as the correct answer, they can hear much more clearly where the differences are.

The last additional tool is the instruction tool. This tool displays an explanatory text of how the learner can pronounce the intended phoneme correctly. The displayed text shows the wrong phoneme and the right one. Furthermore, the text is accompanied with a picture, which helps clarify the text and make it easy to understand.

**RT4** After the complete development of our CALL tool with all of its versions (i.e., animation tool, synthesis tool, and instruction tool), we needed to assess its effectiveness. Therefore, we applied the tool in an Arabic class and started conducting the classroom experiments which were performed with Arabic learners to find out if the tool improved their pronunciation.

Although a number of tools for supporting language learners have been previously developed (e.g., Baldi [Mas01]), it has been very hard to compare their effectiveness. Each of the existing tools exploits different machinery for determining how a student’s pronunciation deviates from the norm so that it is difficult to isolate the benefits that arise from using one recognition engine rather than another from those arising from the different types of feedback. The current project utilises a single speech recognition engine for all three kinds of feedback, thus
making it possible to distinguish between effects that arise specifically from the form of feedback provided.

1.4 Research Methodology

A literature review was conducted into the area of speech recognition, with particular focus on the Arabic language. Furthermore, a discussion of existing CALL tools was provided in order to place this work in context. In particular, we looked at the use of an animated head in Baldi [Mas04] both to see how the animation works and to see how it is used in learning situations. We carried out a number of experiments using the HTK. The experiments focused on the confusable sounds which people find difficult to pronounce as they sound very similar to each other and which the speech recogniser also finds difficult to distinguish. These experiments are aimed at improving the accuracy of the recognition engine that we made using the HTK. We had to conduct these experiments because, as noted earlier, there are some sounds in Arabic that recognisers obtained by training the HTK have difficulty with, and we need very high accuracy for our purpose. Therefore, we carried out these experiments to make sure that we had the optimal recogniser.

The HTK was trained to recognise phonemes of the input speech in order to obtain a phonetic analysis, which is then used to animate the vocal tract. This training is important to make sure that the HTK can cope with minor variations of sound. A program has been developed in Python in order to make this training much easier (see Appendix B). These experiments help us to investigate the performance of the speech recogniser (i.e., the HTK), and to identify some factors affecting the accuracy of speech recognition of Arabic words. Each of these experiments is described in more detail in Chapter 4. After that, the recognised sounds were used to animate the vocal tract, as illustrated in more detail in Chapter 5.

In addition to animating the vocal tract, we used two sources to achieve our goal of supporting the pronunciation of non-native speakers. These additional sources are speech synthesis and written instructions as explanatory texts for how to rectify the learner’s mispronunciation.

Our CALL tool is made concentrating on sounds which experienced Arabic teachers report as causing problems for learners (personal communication). There are some sounds that native English speakers find difficult to produce which we
can diagnose and give speakers some information about.

After completing the development of our CALL tool, we started evaluating it. Therefore, we contacted people in places in Manchester where they teach Arabic, and arranged with their students to use our CALL tool. We carried out a set of sessions where each student used the tool for half an hour. Then, we investigated the results by carrying out a statistical analysis on their performance and by looking at the behaviour traces. These results are reported in Chapter 7.

1.5 Research Contribution

The work which was carried out contributed to a wider project in Arabic speech recognition, leading to the publication of a paper entitled “Dealing with Emphatic Consonants to Improve the Performance of Arabic Speech Recognition” in the 13th International Arab Conference on Information Technology, 2012, ISSN:1812-0857.

Our work has also led to publication of another paper entitled “Diagnostic tool for Arabic learners” which was presented as a poster in the 22nd ‘EUROCALL 2014’ conference on the use of CALL technology for language learning, 2014.

1.6 Summary

We aimed at teaching non-native Arabic speakers how to sound like native speakers. Therefore, we did a considerable amount of work to achieve our aim. Briefly, this work consists of the following:

- We conducted HTK experiments on confusable sounds which led to the publication mentioned in Section 1.5.

- We devised a new way of carrying out an animation of the vocal tract which is done in a less computationally expensive way than similar tools such as Baldi, but which produces realistic effects.

- We performed experiments to improve the naturalness of synthesised speech.

We integrated all this work together in a CALL tool that helps learners to improve their pronunciation.
Chapter 2

Background and Literature Review

Recall from (RT1) that this research aims to use SR technology to help Arabic learners improve their pronunciation by means of three sources of feedback. Therefore, in this chapter, we will review the literature on three main topics: (1) SR technology; (2) the Arabic language and why it can be difficult to learn and pronounce; and (3) language learning using a CALL tool.

2.1 Automatic Speech Recognition (ASR)

2.1.1 Introduction

Speech is a widespread and very effective means of communication among humans, comprising a complicated structure including: voice transmission, language, topic, gestures, and the ability of the listener to interact.

Speech recognition (also referred to as Automatic Speech Recognition (ASR)) is a process of converting acoustic waveforms to strings of words [JMK09]. In other words, it is a technology that allows a computer to identify the words that a person speaks through a microphone or telephone. This technology allows communication with a machine through natural speech instead of traditional input devices (e.g., keyboards). This makes communication easier because speech is the primary communication medium between humans, so it seems likely that people will also find it the best way to communicate with machines.

ASR technology can make life much easier when employed in several applications, such as command recognition (voice interaction with the computer),
2.1. AUTOMATIC SPEECH RECOGNITION (ASR)

hands-free operation and control (as in cars and airplanes), dictation (speech-to-text transcription), and interactive voice response [AEAG+08]. Moreover, it can also be used to help people with disabilities to interact with a machine. For example, personal computers (PCs) can be controlled by voice and used for dictation, which is important for handicapped people. Another application is that an environment can also be controlled, such as switching the light on, controlling the TV etc. [EB03] (as reported by [For03]). Furthermore, it is an ideal method for language teaching and testing. In fact, speech communication with computers, mobile phones, and domestic appliances is likely to be the main interface of human-interaction in the future [AEAG+08].

Considering the importance of ASR, there have been several research studies conducted on ASR which aimed at allowing a computer to recognise words in real time with 100% level of accuracy. This is a very challenging task due to various factors. Perhaps the complexity of the human language is the most significant challenge for speech recognition. Humans usually use their knowledge of the speaker, the subject and language system to predict the intended meaning while in ASR we only have speech signals. Languages have, for instance, various aspects of ambiguity. This is a problem for any computer-related language application. Many kinds of ambiguities arise within the ASR like homophones (words that are pronounced the same but with different meanings such as “to”, “too” and “two”) and word boundary ambiguity from words that are not clearly delineated in continuous speech [For03]. The following example [Mor00] shows the problems that are caused by faulty segmentation:

It’s not easy to wreck a nice beach.
It’s not easy to recognise speech.
It’s not easy to wreck an ice beach.

In contrast, achieving a high performance speech recogniser may be influenced by paralinguistic (i.e., non linguistic) factors [Sch02] such as the speaker’s age and sex, surrounding noise (i.e., undesirable information in the speech signal), signal channels, and other factors which make it impossible for the speech signal to be repeated [CT10].

Speech recognition systems can be classified according to the types of utterances they can recognise. One kind of classification concerns whether the system can only recognise speech by a particular person (speaker dependent) [JMK09] or it can recognise the speech from people whose speech has not previously been
recognised by the system (speaker independent). A second kind of classification concerns whether the system is able to recognise continuous speech or only isolated words [JMK09]. It can also be classified by the size of the system’s vocabulary from small vocabulary to large vocabulary. Large vocabulary means the system can support roughly 20,000 to 60,000 words. However, a given package can fit multiple classes.

Most investigations and much of the research that have been conducted in the field of ASR so far were aimed at English. In contrast, the amount of research on Arabic does not reflect the number of its native speakers (i.e., 5% of the world’s population) [AMM09]. Arabic has a number of characteristics which means that some of the steps usually carried out when recognising spoken English are likely to be difficult or irrelevant. We will discuss these (i.e., Arabic and Arabic ASR) in Section 2.2.

2.1.2 Hidden Markov Model (HMM)

The statistical approach for SR is dominated by a powerful technique named the Hidden Markov Model (HMM) [HAH01]. The HMM approach has attracted researchers in SR for many years. It has great power to model the development of a signal by a Markov process [MHF07]. A Markov process can be defined as a random process whose future states depend on the current state.

The HMM basically recognises speech by determining the probability of each phoneme at successive frames of the speech signal [MIL93][Sg88]. The fundamental part of the Markov model is the state. A group of states matches the positions of the vocal system of the speaker. These states are hidden (as noted from the model’s name), and an embedded task in an ASR system performs the decoding of the current state sequence to show the speech signal [Çöm03]. Afterwards, target words to which these states belong are modelled in a sequence of phonemes, and a search method is applied to match these phonemes with phonemes of spoken words [AEAG+08].

The HMM is depicted by a graph structure consisting of nodes and arcs. The nodes represent the hidden states, while the arcs show the transitions between these nodes. These nodes are connected vertically to other corresponding nodes which represent the observations from which hidden states may be inferred, and the connectors between these two kinds of nodes (i.e., observations and states) represent emission probabilities. Time characteristics and spatial characteristics
are modelled by the state probability distribution and the emission probability distribution, respectively. The states of the HMM cannot be observed directly; however, they can be deduced from a sequence of observations [MHF07].

From the above depiction, the HMM can be viewed as a set of transition probabilities and emission probabilities. These probabilities can be calculated by using a deterministic algorithm called Viterbi, which helps to find the most likely interpretation of a sequence of phonemes given a set of observations (i.e., acoustic signals). This view can be illustrated in the following example.

Figure 2.1 shows a concrete example of an HMM. In this example, we will assume a learner trying to pronounce the Arabic word “MIN” (which means “from”); the wave signal of sounding this word is shown in Figure 2.2. Figure 2.1 shows three hidden states, ‘M’ pronunciation, ‘I’ pronunciation, and ‘N’ pronunciation. We attempt to figure out whether the learner sounds the word ‘MIN’, or he mispronounces it by sounding the word “MAN” (meaning “who”). As noted earlier, we have two kinds of likelihoods: (i) state transition likelihood, and (ii) emission likelihood. Matrix ‘A’ (Example 2.1) indicates the state transition probabilities, while Matrix ‘B’ (Example 2.2) indicates the emission probabilities as follow:

\[
A = \begin{bmatrix}
M & I & A & N \\
M & 0 & 0.7 & 0.3 & 0 \\
I & 0 & 0 & 0 & 1 \\
A & 0 & 0 & 0 & 1 \\
N & 0 & 0 & 0 & 0
\end{bmatrix} \quad (2.1)
\]

\[
B = \begin{bmatrix}
M & I & A & N \\
1 & 1 & 0 & 0 & 0 \\
2 & 0 & 0.2 & 0.8 & 0 \\
3 & 0 & 0 & 0 & 1
\end{bmatrix} \quad (2.2)
\]

Each element of the matrix consists of a single likelihood. For instance, in Matrix ‘A’ (i.e., Example 2.1), ‘A_{M,I}’ is the likelihood of transition from state ‘M’ to state ‘I’ which is equal to 0.7, and ‘A_{I,N}’ is the likelihood of transition from state ‘I’ to state ‘N’ which is equal to 1, and this naming convention is followed with other probabilities. In Matrix ‘B’ (i.e., Example 2.2), ‘B_{1,M}’ is the emission probability that the speaker intended to say ‘M’ given the signal shown in the first observation, which is equal to 1, and ‘B_{2,A}’ is the probability that
the speaker intended to say ‘A’ given the signal shown in the second observation. As Figure 2.1 shows, $B_{2,A} = 0.8$ which means also that there was a 0.2 (i.e., $B_{2,I}$) possibility that the speaker intended to say ‘I’ given the signal shown in the second observation. Given this state of affairs, we can see that the more likely interpretation of this sound is ‘MAN’ rather than ‘MIN’. The reason is that if we go the ‘MAN’ route, then the probability is $(A_{M,A} \times B_{2,A} = 0.3 \times 0.8 = 0.24)$, whilst if we go the ‘MIN’ route, then it is $(A_{M,I} \times B_{2,I} = 0.7 \times 0.2 = 0.14)$. Using $A$ and $B$ matrices, we can define an HMM which can be described by the notation illustrated in Equation 2.3:

$$\lambda = \{A, B\}$$  \hspace{1cm} (2.3)

![Figure 2.1: Example of HMM](image)

The use of HMMs has been popularised by the availability of an efficient iterative method for obtaining optimal estimation of the likelihood of emission and transition probabilities. This method is called the expectation maximisation (EM) algorithm [MHF07]. The EM algorithm will let us approximate these two
kinds of probabilities in a reasonably efficient manner by providing maximum likelihood estimations for them iteratively. Every iteration consists of two steps: (1) Expectation (E) step; (2) Maximization (M) step. The E step of the algorithm gives both the emission and transition probability distributions, given the known values of the observed variables and the current estimate of the parameters. The M step re-estimates the current parameters to have maximum probability, given the probability distribution obtained in the E step. Each such iteration improves the true likelihood, or leaves it unchanged (if a current maximum likelihood has already been reached) [NH98].

Here, the most difficult point is calculating these probabilities correctly. EM algorithm assigns values to all these probabilities after it has been supplied with a set of training data. After this algorithm is run, we will obtain the optimal values of these probabilities (i.e., the HMM parameters).

There are numerous speech recognition toolkits based on the HMM. The HTK, Sphinx [LHR90], CSLU [SCDV+98] and Julius [LK09] are examples of such toolkits. We have used the HTK, which is described in much more detail in Chapter 4.

2.2 The Arabic Language

2.2.1 Introduction

Arabic is part of the Semitic language family as well as being one of the oldest languages in the world today. Currently, it comes fourth in terms of the number of native speakers (estimated to be 221 million) [LoL09]. Arabic is the official language of more than 22 countries. It is one of the six official languages of the United Nations. The Arabic script is read and written from right to left and has basically 34 phonemes (a phoneme is the smallest element of speech units) [Cry08], of which six are vowels, and 28 are consonants. Research on the Arabic language has mainly concentrated on Modern Standard Arabic (MSA), which is a version of Classical Arabic with a modernised vocabulary and is the language used in formal writing and speech.
Developing high-accuracy speech recognition systems for Arabic faces many hurdles. One of the critical issues for Arabic SR as for SR of other languages is the pronunciation variability (i.e., people have different accents). This is true even for MSA, which is supposed to be uniform across the Arabic speaking world, because of the influence of the informal dialects that people use in everyday speech. This means that the enormous dialectal variety is a major problem as the same word could be pronounced in various ways by different speakers with different dialects even within MSA. Take the example of /radʒul/ (which means “man”), which could be pronounced /radʒul/, /ragul/, or /raʒul/. These regional differences in pronunciations of MSA might create an error in recognition. In addition, the nature and complexity of the inflectional and derivational morphology means that there are large numbers of very similar sounding forms. This means that the SR has more chances of making mistakes, and is thus harder to guide. For example, the root /ʕal?m/ leads to: /ʕalam/ (meaning “flag”), /ʕalim/ (meaning “he knew”), /ʕilm/ (meaning “science”), /ʕulim/ (meaning “it was known”), /ʕallam/
(meaning “he taught”) or /ʔullim/ (meaning “he was taught”). Arabic readers infer the appropriate form based on the context and their linguistic knowledge.

From the point of the learner, the absence of diacritics (diacritics are marks placed above or below a consonant to specify what the following short vowel is; they are generally omitted in writing modern Arabic) in the previous example causes many ambiguities in the pronunciation of words. These ambiguities can arise in Arabic written without diacritics. This means that a single written form may correspond to numerous underlying forms which are pronounced differently, as shown in the previous example of the root /ʔ?l?m/. In this point, learners can have difficulty working out which underlying form is intended, and hence may attempt to pronounce the wrong one.

Arabic is also considered one of the most morphologically complex languages, commonly referred to as non-concatenative morphology. For example, a simple Arabic verb root such as /ktb/ (which means “to write”) can be modified into more than thirty words in a non-concatenative manner using processes like infixation and germination [AN98]. This will therefore increase the out-of-vocabulary rate [KBH+02].

Another problem with Arabic is the existence of some confusable phonemes such as emphatic sounds[1] These sounds have two related problems. Firstly, they are mispronounced most of the time, even by Arabic native speakers [SHC07]. The distinction between these sounds is recognised by Arabic listeners and it is critical in certain situations for determining the meaning. However, Arabic speakers are often lazy in the pronunciation of these sounds. When they are speaking carefully, they will attempt to produce this distinction, but in many situations the distinction is not maintained, leading to speech which is hard to understand. Secondly, the emphatic consonants have plain equivalent consonants in Arabic and these plain consonants are very similar to the emphatic ones. It is quite challenging to train the speech recognition system to distinguish between the emphatic consonants and the non-emphatic ones and to recognise each of them accurately.

[1] Also known as pharyngealised or coloured sounds. Not related to emphasis.
2.3 Pedagogy of Learning

CALL tools are intended to support language learning. They analyse learners’ performance, test them, and then help them gradually to improve their language skills (e.g., speaking, listening, reading, and writing) and overcome their weakness in different language areas (e.g., pronunciation, vocabulary, and grammar). The following sections carry out an exploration of pedagogical aspects of computer technologies, such as CALL, in learning.

2.3.1 Aptitude and Motivation

Each learner has his own individual characteristics. These characteristics can be accommodated by the use of Information and Communication Technologies (ICTs) [Ken99]. One of the difficulties facing learners is that the learning is standardised not individualised. For example, such standardisation can be seen in the design of most language learning programs, which expect all learners to be the same and work in the same way at the same speed. Learners will lose both their interest and their focus if their individual characteristics are not taken into account. Aptitude and motivation are two main factors of individual variation which play a role in the learning process [EZ00].

Aptitude is a major differentiating factor between learners. There has been debate on the domain of language aptitude [Ske91]. Aptitude for language is a concept that applies to both first and second languages which is mostly overlooked [LHE78]. Aptitude language measures, such as Carroll and Sapon’s Modern Languages Aptitude Test (MLAT [CS59]) developed in 1957, can be able to separate students with the ability of auditory acuity from those with the ability to ignore distractions. This might give essential sources of variations between learners [Wes81]. Another aptitude test is Pimsleur’s Language Aptitude Battery [Pim66] which is influenced by MLAT in its development as there is considerable overlap in the concepts suggested by these two language aptitude tests [GM92]. The significance of skills identified by the measures of language aptitude might be more important for language learning in natural conditions (i.e., informal contexts) than in schools (i.e., instructional contexts) [Rev83].

With the help of computer technology, it is possible to take different learner characteristics into account because the same materials can be covered with different types of learners who can work at their own speed of progress [EZ00].
Motivation is another major differentiating factor between learners. A cognitive view of motivation, belonging to Ushioda [Ush96], looks at motivational thinking as a mediating factor for language learning variation. If learners have positive learning experiences, this will tend to increase their self-motivation and interest, and vice versa. There are two crucial processes which characterise motivational thinking. The first process is called ‘causal attribution’ which means that learners attribute why they succeed or fail to internal or external causes. The second process is called ‘intrinsic motivation’ meaning that learners are self-motivated and independent.

With the assistance of computer technologies, motivational thinking can be promoted by providing feedback process for success or failure [EZ00]. For example, in our CALL tool, a response type of ‘smiling face’ vs. ‘unsmiling face’ is used in exercises to convey approval and disapproval. This means that computer technology can accommodate individual characteristics of learners.

2.3.2 Thinking Tools

Computer technologies, books, videos, or other learning means can only transfer information to learners. Indeed, learners are the ones who do learning via a variety of mental processes (i.e., thinking). These mental processes are activated by learning activities coming from instructional devices, such as computer technologies. Therefore, according to [Jon92], it is worthwhile to focus on how learners are required to think more effectively in finishing tasks, rather than focusing on developing a complicated device. In other words, computer technologies should provide learners with thinking tools that facilitate their thinking processes. Computer technologies can help to trigger cognitive language learning methods which orient learners to the process of knowledge construction and organisation.

2.3.3 Conditions for Learning Language

Esch and Zähner [EZ00] argue that there are four conditions that must be met if we want computer technologies, such as ICTs, to become relevant to language learners. Otherwise, without these conditions, ICTs may stay relevant to students’ daily life but irrelevant to their learning a new language. These four essential conditions are as follow:

1. Accessibility: Computer technology needs to be accessible, and this implies
that learners, in general, are confident with technology and find it easy
to use. Failure to meet this condition may result in student resistance to
the use of technology for language learning. Gillespie and McKee [GM99]
showed that the technology may be seen as relevant to language learning
but many users find it inaccessible. This condition was an issue since around
2000. However, it is probably no longer such a problem because most people
do now have regular access to computers and are familiar with computer
technology.

2. Autonomy: This condition helps learners to make change and build up their
confidence. Using new CALL tools for language learning requires learners
to adjust their cultural model. This means learners should be self-motivated
by their new experiences using CALL tools in order to review their preceding
model [TM97]. In our CALL tool, a learner must do his pronunciation tasks
independently at his own speed.

3. Reflectivity: This means learners have the ability to do a critical analysis
of their learning activities and their practices during the learning process.
For example, in our CALL tool, a learner could be aware of the difference
between his activity of pronunciation and the typical one.

4. Interactivity: This condition focuses on the effect of peer group and social
interaction in the language learning process, especially its effect in chang-
ing learners’ cultures. The reason of this influence in changing cultures is
that interactivity creates a safe environment which helps to build up the
motivation of learners.

2.3.4 Assessment in Learning

Assessment can be defined as the process that “focuses on describing student
learning, identifying where each student is in his or her personal learning pro-
gression, diagnosing any difficulties students may be having in their learning,
and providing direction to the teacher and the student in the steps to be taken
to enhance learning” [Ber08]. It is extremely important for a teacher to assess
a learner’s performance as this assessing has a strong effect on their learning.
Indeed, according to Trotter [Tro06], it is the most important aspect of learn-
ing. The main point of assessment in learning is to measure what students have
2.3. PEDAGOGY OF LEARNING

learned in a course or programme of study. By measuring what students learn, tutors can control student progress, decide what gaps in the student’s competence need to be addressed, and document attainment. There are different types of assessments which are used in learning. These types will be discussed below showing their effectiveness.

The first type is called formative assessment. Higgins et al. [HGTM10] define this type as "work that a student carries out during a module for which they get feedback to improve their learning, whether marked or not". So, it is employed during the learning process to modify teaching and learning activities in order to improve student achievement. Research shows that assessment that gives informative feedback while a student is learning has more effect on student attainment than any other factor [QAA07]. Formative assessment with its feedback during the learning process can strongly affect motivation, encouraging interest, commitment, intellectual challenge, independence and responsibility. This type of assessment plays a major role in effective feedback on a student’s progress [TGG94]. Some examples of formative assessment are: assignments, projects, quizzes, and asking questions.

The second type of assessment is named summative assessment. This type is contrasted with formative assessment as it focuses on the educational outcomes. It happens at the end of the learning process, and summarises the qualitative results of the teaching and learning process after the completion of the course. So, summative assessments provide teachers and students with information about the achievement of knowledge. Examples of summative assessment are: end-of-unit or chapter tests and end-of-term or semester exams (i.e., final exams). We can say that this type helps to reward final achievement of the student work and to determine his final mark of what he scored in the studied course. Our CALL tool, at the end of the teaching session, provides a final score as a percentage of his performance.

Another type of assessment is called diagnostic assessment (also known as pre-assessment). This type is an essential part of the learning journey. Its importance comes from the determination of the learner’s starting point. So, it can be viewed as a benchmark from which learners’ progress and attainment can be measured. In other words, it can help to form the basis for measuring the distance travelled during a student’s time on a studied course. Much research has been conducted focusing on how important diagnostic assessment is, such as the study by Alderson
Diagnostics provide feedback that can be acted upon. The teacher has to have a plan, a list of actions that he and his students can take to address a particular problem once diagnosed. For example, if it is a pronunciation problem, he could send them to the lab to do practice pronunciation tasks that include the sounds in question. He could also provide extra exercises for homework that target the areas of weakness, or even allow the students to do homework assignments tailored to their needs instead of doing all of the generic homework assignments.

Diagnostic assessment can be used as a means of assessing the teacher as this assessment can also be applied to the learner at the end of a learning process. In other words, it can be used to measure the teacher’s achievement. The teacher wants to know what level his students are at before starting to teach them, and what level they are at at the end. So, the teacher can see how much difference he made.

The Arabic learner who uses our developed CALL tool is asked to do pronunciation tasks for a half-hour session. The tool represents the formative assessment because during the session after each task, the learner progress is checked by giving him feedback about his pronunciation. Furthermore, we are using pre- and post-tests (i.e., diagnostic assessment) as part of the process of gathering information about whether over the body of students we can make a claim that the tool is useful.

2.4 CALL tools in Language Learning

2.4.1 History of CALL

Computers have been used for language teaching and learning since the 1960s. In the 1960s and 1970s, CALL featured repetitive language drills, referred as drill-and-practice. In this behaviouristic stage of CALL, CALL promoted the acquisition of knowledge or skill through repetitive practice, such as memorisation of spelling or language vocabulary, and practising arithmetic tasks like addition and subtraction. PLATO is a well-known tutorial system that represents an example of CALL in its early stage.

The acronym CALL was coined at the beginning of the 1980s. The first mention was found in [DS81], where the authors talked about the first steps in
CALL at Ealing College of higher education. By 1982, CALL was widespread in the UK, the USA, Canada and the rest of Europe. In the UK, CALL appeared in the first issue of the newsletter CALLBOARD (July 1982) and in [DH82]. In the USA, the acronym CALI (Computer Assisted Language Instruction) was preferred to be used at the beginning. This alternative acronym appeared in CALICO (established in 1982) which is the oldest organisation that promotes using computers in language learning [DWRH09]. CALL term became more dominant and favourable than CALI since the association of CALI tends to focus on a teacher-centred rather learner-centred approach. Now, CALICO uses CALL term instead of CALI.

In 1983, TESOL (Teaching English to Speakers of Other Languages) set up its CALL Interest Section (CALL-IS) to define issues and standards in the field of computer-mediated language instruction, and to encourage research and development in the area of language learning based on computers [Ken96].

Another term which is equivalent to CALL and which emerged in the 1980s is TELL (Technology Enhanced Language Learning). TELL was adopted in the 1990s by TELL consortium led by the University of Hull in order to produce courseware of language learning. Also, TELL appeared in the journal of CALL-Austria, TELL&CALL [BT97].

During the 1980s, the scope of CALL was widened to include the communication approach and range of a new technology, such as speech technology. In 1997, Levy defined CALL as “the search for and study of applications of the computer in language teaching and learning” [Lev97].

In the 1990s, CALL had advanced through its historical stages approaching to seek both integrating various skills (i.e., speaking, listening, reading, and writing) and integrating technology more fully into the language learning process. The students learn different technological tools in their ongoing process of language learning [WH98].

Many changes, with time, occurred in the historical stages of CALL. CALL changed from being a mechanical tutor that featured extensive drills to being an effective tool that gives students the ability to respond and adapt to changes rather than training in a single way to achieve a task. The roles of teachers have also changed with time. In this age of information-rich life, teachers are not the only source of language information. According to cognitive theory, teachers
do not pour information into the heads of their students, but rather those students interpret and organise the provided information, fit this information into prior knowledge, or revise prior knowledge upon the arrival of new information [VDK83]. So, a teacher has become a facilitator of learning language rather than the sole source of information. As facilitators, teachers should offer information in many ways based on students’ needs to improve their language skills [WH98].

2.4.2 Importance of CALL

Nowadays, there is an increasing demand for language tutoring. Many people have to learn a new language. They find that using a non-native language is a requirement for succeeding in different situations, whether in business or everyday conversations [Mas04]. In language learning, a learner is going to encounter various types of problems in various aspects of language learning including: vocabulary, grammar, morphology, writing system, reading, listening and pronunciation. In this research, we are interested in helping Arabic learners to overcome the last aspect, which is “pronunciation” (i.e., how to pronounce sounds of the language) by using the technology of ASR and CALL.

The use of technology in language learning has been the focus of recent studies because of its effectiveness in enhancing language learning. Zhao [Zha03] states that the support of technology in language learning is at least as effective as human teachers, if not more effective. Nutta [Nut98] investigated the learning of ESL (English as a Second Language) students who spend the same amount of time for one week (i.e., one hour a day) learning verb tenses in English. The investigation was conducted on two types of students: (1) students who attend a regular class and listen to teacher’s instructions without computer equipment or other instructional technology, (2) students who sit outside the classroom in front of a multimedia computer program which includes facilities of audio, video, recording, etc. The researcher found that the performance of those students who used the computer program is the same or significantly better than those students who attend the class regularly.

CALL technology is intended to support Second Language Acquisition (SLA). With the help of speech recognition and synthesis technologies, the learner can use a computer program to work on a very close natural conversation in a language that he is trying to learn. Furthermore, the learner can provide a computer program with spoken commands and then the program would respond and perform
the command.

A number of examples of Arabic CALL tools have been provided by Harless and others [HZD99] to test the effect of a virtual Arabic-talking program on language learning. The program allowed students to meet a virtual character of a native Arabic speaker, and speak to him via speech recognition technology. After students interact orally with this virtual character for at least eight hours per day for four days, their skills in reading and speaking were enhanced significantly, with a small improvement in listening skill. In another study, Holland et al. [Hol05] found that a computer program called ‘speech-interactive graphics microworld’, which allows the Arabic learner to select from a pool of short 72 Arabic commands to say to the program. Utterances in the program are processed by a speech recogniser, which trained on the 72 Arabic sentences for a demonstration scenario. The program was used with 6 university Arabic students and 16 soldiers who had studied MSA. This study reported that the learners can build objects by talking to the computer, helped them to be more motivated and to improve their speaking skill. [MJS+04] designed a CALL tool called Tactical Language Training System (TLTS). This learning tool is designed to teach Arabic spoken communication to American English speakers. TLTS detects learner speech errors, and this detection enables to give a tailored feedback to the learner. Another application of CALL is called “HAFSS” [AHR+06] which is aimed at teaching Arabic pronunciations for non-native speakers through teaching them the recitation of the Quran. “HAFSS” uses a speech recogniser to detect the errors in the recitation. The recognition is examined only on the probable pronunciation variants in order to increase the accuracy of the speech recogniser. This application is fed with prerecorded recitations, target text materials of recitation, and teaching animations. Performance evaluation of this application showed that “HAFSS” identified the error in 62.4% of pronunciation errors, reported a request of repeat for 22.4% of the errors and gave false acceptance of 14.9% of total errors. “HAFSS” proved to be useful in learning Arabic. However, it does not give a detailed feedback, and we hope to fill this gap in our research by giving multiple forms of feedback to Arabic learners.

[Mas04] developed a CALL tool called Baldi as an English language tutor, which was followed by other versions for teaching other language. For instance, Badr [OCM05] is an Arabic version of Badi, which dedicated for Arabic learners. All experiments with Baldi and Badr showed that training with such CALL tools
allows more improvements in the pronunciation of a learner (described in more
detail in Section 2.4.3).

There are other CALL tools developed for learners of a language other than
Arabic. Menzel et al. [MHM+01] developed a pronunciation training system
which is targeted at intermediate Italian and German students learning English.
Another CALL tool for English learners is called BetterAccent Tutor [KK00]
which allows students to compare stress and pitch patterns visually. Then, it
helps them to identify, understand and correct their pronunciation errors. For
Japanese pronunciation learning, Hew and Ohki [HO01] developed a system that
provides animated graphic annotations and immediate visual feedback in order to
form a Japanese computer-assisted language learning (JCALL) tool for Japanese
learners. Another CALL tool is created by Machovikov et al. [MSC+02] for Rus-
sian learners using ASR. “Tell Me More” is a speech recognition tool for language
learning developed by Aurolog, and this tool is available in English, Spanish,
French, German, Italian and Chinese [BLR05]. There are other researchers who
also use SR system in CALL tools, but it is not clear whether they carried out
any studies of the effectiveness.

Providing immediate feedback to learners through computers has been thought
of as a key factor in language learning, and such an important factor is recognised
by foreign language educators [Cha99] [Sal01]. ASR technology has the potential
feature of providing targeted feedback. Pronunciation is an essential part of lan-
guage learning. However, providing feedback which is beneficial and accessible is
not easy.

In traditional methods, providing feedback is often done by a tutor, who may
or may not be good enough to assess the pronunciation of the student. Typical
approaches of providing feedback include having students repeat the pronun-
ciation again and again after the tutor, or showing them how to produce the
pronunciation in a very abstract way. With the aid of advanced ASR technology,
the student can be provided with feedback in an effective way. When Menzel et
al. [MHM+01] developed their training system with the help of SR technology,
they stated that the system should provide learners with sufficient feedback to
allow them to identify their weakness and should point out the way to improve.
Mostow and Aist [MA99] suggested three features for feedback: visual, template-
based, and model-based. The first suggestion is that the computer program can
analyse the pronunciation of the student and display his articulation visually,
and compare it with that of a native speaker. The computer program could also display the position and the movements of the tongue when the student produces a sound, which can also be displayed in comparison to that with a native speaker. The second suggestion is that the computer program can compare what students say to prerecordings as a template. The third suggestion is that the program can evaluate the student’s pronunciation against the typical pronunciation. These suggestions are observed and followed in our research.

2.4.3 Detailed Example of a CALL tool - Baldi

It is worth considering in a CALL tool to have a feature of visual information provided to learners. Research from Baylor College of Medicine in Houston and the City College of New York reports that the visual information we assimilate when we see can enhance our comprehension of the spoken words by a six-fold increase [Col09]. Visual information from facial movements can improve the intelligibility of acoustic stimulus especially in a noisy environment [Mas04]. Considering this value, visible speech of an animated agent assists face-to-face oral communication. Moreover, it can serve as a language tutor in human-machine interaction. Massaro [Mas98] states that “speech as a multimodal phenomenon is supported by experiments indicating that our perception and understanding are influenced by a speaker’s face and accompanying gestures, as well as the actual sound of the speech”.

To improve the quality of spoken language, the combination of auditory (from the voice) and visual speech (from the face) is highly effective for many reasons. These reasons include: (i) robustness of visual information; (ii) complementarity of acoustic signal and visual speech; and (iii) efficient integration of auditory and visuality information [Mas04]. The first reason (i.e., robustness) means that the capability for obtaining speech information from the face (i.e., speech reading) is very high. The second reason (i.e., complementarity) means that if auditory information is a weak source, then visual speech is a strong source, and vice versa. The third reason (i.e., integration) means that the learner can combine auditory and visual information in an optimal way. Fuzzy Logical Model of Perception (FLMP) predicted many experimental results, and has shown the efficiency of integration [Mas98].

The above reasons have motivated some researchers to develop a 3D computer-animated talking head. One such development is called Baldi. Baldi is a
software-driven language tutor which was created in the Perceptual Science Lab (PLS) at California University at Santa Cruz [ASCa]. After Baldi’s success in improving the quality of spoken output, it led to more useful applications. One such application is the utilisation of Baldi as a language tutor.

[Mas04] claims that there are many advantages of using a computer-animated agent as a language tutor. One advantage is the spread of computers, allowing CBT (Computer Based Training) to become a widespread method for teaching non-native speakers. A second advantage is the CBT’s ability to use multiple sources of information, such as text, sound, and image at the same time. Moreover, the program is available at any time to the learner, and the interaction during lessons is face-to-face oral communication with individualised instruction. It seems plausible that this last claimed advantage is the most significant, because it enables learners to have immediate face-to-face feedback which teachers cannot typically provide.

The pedagogy of Baldi is effective in the learning of vocabulary and grammar for both children with language challenges (e.g., hearing loss and autism) and adults learning a new language [Mas04]. Baldi produces visible speech which approaches the accuracy of a native speaker, and is effective enough to be useful for non-native speakers [Mas04]. A study was carried out with eleven Japanese speakers ranged in age from 19 to 36 for whom English was a second language. They were trained to articulate 16 minimal pairs of prompt words having the two phonetic contrasts, /r/ and /l/. Each subject recorded 32 prompted words, one utterance of each the 16 pairs. The study showed that speech articulation improved as the overall performance of their pronunciation improved from about 54% to 60% across 3 days of training. In addition, although the exercise was aimed at helping them with pronunciation, it also helped them hear the difference when learning new words [ML03].

In the study by Liu et al. [LMC+07], an experiment was performed using a Chinese version of Baldi, which is called Bao. This experiment aimed to teach Chinese syllables to English native speakers. In this experiment, 101 learners were assigned to two training sessions followed by two testing sessions. In the first training session, the learners were trained only by hearing 23 syllables articulated by a Chinese native speaker, and this audio is accompanied by a corresponding text displayed on the screen. After that, the learners finished the first training session and took the first testing session. In the second training session, the
learners were allocated randomly to one of three training conditions. The learners who were assigned to the first condition were trained only by hearing the audio in the first training session again, whereas those who were assigned to the second and third conditions were trained by the same audio as well as a corresponding visible speech. In addition to hearing the same audio, the learners who belonged to the third training condition were given visible articulators of Bao, a modified version of Baldi. Subsequently, the learners finished the second training session and took the second testing session. The result conveyed that training with Baldi allows more improvement than training with normal speech.

Research has been conducted to expand the ability of Baldi to be multilingual. An example is a study which was done using Badr (i.e., an Arabic version of Baldi) [OCHO15]. In this study, there were 19 Arabic native speakers and 100 generated Arabic utterances in which each utterance included three words. These utterances were randomly given to the experiment participants under three conditions: (i) auditory information alone, (ii) audio input aligned with Badr, and (iii) audio input aligned with the video of a real face. The participants were asked to write down the words they recognised for each utterance under the previous conditions. The average accuracies of recognised words were 30%, 54%, and 69% for the unimodal auditory condition, bimodal synthetic face condition, and bimodal real face condition, respectively. Although videos of real faces seem to be substantially better in this experiment, using synthesised images (i.e., Badr) allows teachers to create new examples for on-the-fly teaching without having to do new video recordings of all prompted words being pronounced right and wrong.

It is well-known that learners will just improve through practice even without any tool. Therefore, in our CALL tool, we were careful to attempt to eliminate the practice effect by running a version of the tool called “language lab”, which works as a control group (described in more detail in Section 7.3.1). It is not clear from the Baldi study what the control group is (i.e., the developers do not publish the details of the control group).

2.4.3.1 How does Baldi work?

Figure 2.3 [ASCb] shows a 3-dimensional wire frame model of Baldi. The inside view of Baldi is valuable for language learning. Figure 2.4 reveals the four views of Baldi with transparent skin showing inside articulators (back view, sagittal
Figure 2.3: Baldi’s wire frame model

view, side view, and front view) [ML04]. A sagittal view is a vertical view which passes from front to back dividing the body into right and left sections, and it can be shown more clearly in Figure 2.5 [OCM05]. Baldi has an authoring engine consisting of automated software tools that enable users to control its appearance (e.g., facial movements and emotions). With the assistance of these tools, Baldi can be programmed to utter any voice. Baldi can be represented as an electronic puppet with more than 100 strings. Sixty one of these strings are used to control the facial movements. Each of these 61 strings has a typical set of mathematical parameters which allow Baldi to mimic the human face. The parameters of a string can be adapted for adding features to specific words and creating a unique attribute. Baldi deals with two types of input: either a recorded voice or text that is converted into a voice [ASCh].

2.4.4 Graphical Simulation of the Vocal Tract

The human vocal tract plays a critical role in producing speech. Figure 2.6 illustrates some basic anatomy of the vocal tract. As Figure 2.6 shows, the vocal tract consists of a number of articulators which are involved in the synthesis of speech (i.e., the larynx (also called voice box), tongue, lips, velum, palate, alveolar ridge, and teeth) in which lips, tongue, and teeth are the main articulators used in the visual representation of speech sounds. In this figure, the larynx can be viewed as an input source, whereas the remaining parts of the vocal tract can be
2.4. CALL TOOLS IN LANGUAGE LEARNING

considered as a filter which produces an output, and this output is our speech. This means that the folds of the larynx vibrate when air comes up from our lungs. After that, the auditory energy is filtered to become an output sound [Ass06].

In addition to the acoustic output, the vocal tract is also involved in the graphical representation of the sound produced (i.e., articulatory view). This is because each articulator gives information about the output sound which can help in learning language. In our research, we animate the vocal tract in order to teach non-native speakers how to speak like native speakers: more details of this are presented in Chapter 5.
2.5 Speech Synthesis

Simply, speech synthesis is the process of making the computer talk or speak to the user. Text-to-speech (TTS) synthesis (i.e., the generation of speech from text input) provides another method of giving spoken language as a beneficial output to language learners who use CALL tools [Han09]. TTS synthesis systems are different from classic methods of providing the computer with a voice, such as a human recording, in a number of respects. TTS synthesis generates speech from text input, has the unique ability of creating and editing speech models, and generates feedback on demand to learner interactions [She81] [EL00]. There are many examples of TTS applications that have been developed. Some of these examples include talking clocks, fixed weather reports, talking dictionaries, talking text, dictation and dialogue systems for booking flights, hotels, and so on [BL00].

Typically, TTS synthesis systems are composed of two modules: (1) NLP module, and (2) Digital Signal Processing (DSP) module. The NLP module aims to generate an unambiguous phonetic transcription of the input text, while DSP
module is aimed at transforming the control parameters into waveforms (i.e., speech). No standard technique has been established for the NLP module. By contrast, there are two standard techniques that are used to develop the DSP module, namely formant and concatenative synthesis. Before the turn of the century, formant synthesisers, which produce speech by modelling the acoustic formant patterns of speech sounds, were the dominant technology. Concatenative synthesisers, which are based on the concatenation of diphones, are currently the dominant technology. A diphone consists of the second half of one allophone (i.e., a phonetic variant of a phoneme) and the first half of the next allophone [Han09] [Dut97] [HAH01].

Deploying speech technology, such as speech synthesis, in CALL tools can be used to teach foreign language skills. This integration has emerged from a growing need for instructional exercises that give students a chance for interactive speaking practice outside the classroom. The ability to engage students in conversational interactive environments of the target language is an important, if not the most important, aim of second language acquisition. CALL tools integrating speech technology have made it possible to create self-paced interactive environments for language learners [EK98].

TTS synthesis could play a major role in response to the above need. Sherwood [She81] noted that typing text is easier than recording a voice again and again, and that searching through a textual database is easier than retrieving recordings from audiotape. Moreover, he observed that TTS synthesis has the advantage of generating speech models on demand, which gives the opportunity to exploit this in a CALL tool in order to provide learners with tailored feedback. These advantages were realised by the technology specialists themselves, and they see TTS synthesis as a tireless substitute to a native speaker [Dut97] [KZK00]. These advantages of TTS synthesis can be exploited in different CALL tools.

There are many CALL tools that have been developed based on TTS synthesis. One example is called FreeText which is targeted at advanced learners of French [Ham03a]. Another CALL tool is named Appeal which helps students learn via individual interactive computer sessions. Appeal uses both a word concatenation technique and PSOLA (Pitch Synchronous Overlap and Add) technique that helps to manipulate a waveform directly [dP97]. DICTOR and Ordictee are other examples of CALL tools devoted to dictation [SO99] [MGS00]. Another tool is called SAFRAN, which focuses on presenting individual and combined
sounds to the learner. These sounds are retrieved from a database where they are kept in textual format [Ham03b]. Other examples of tools are SDS (Let’s Go Spoken Dialogue System) [RE04] and SCILL (Spoken Conversational Interaction for Language Learning) [SWZ04]. They are spoken dialogue systems that exploit the ability of TTS synthesis to generate spoken utterances from text on demand. This feature of TTS synthesis is a solution to the difficulty of storing all possible responses to learners in the form of human speech recordings.

According to Handley and Hamel [HH05], CALL tools place demands on the quality of the output of TTS synthesis systems, especially in three aspects: comprehensibility, naturalness and accuracy. The first aspect (i.e., comprehensibility) means that the quality of the output should be easy to follow and the learner should understand the speaker’s intended message [FN99]. The second aspect (i.e., naturalness) means that the produced speech should be natural as if it was produced by a native speaker. The third aspect (i.e., accuracy) simply means that the produced speech is free from error. We have worked on the quality of speech synthesis for the synthesis version of our CALL tool as described in Chapter 6.

### 2.6 Conclusion

This chapter set out to provide a background discussion on the topics needed for this research. These topics covered our research aim of developing a CALL tool as a pronunciation support for Arabic learners. First, the topic of ASR technology was presented as an important topic to cover since our CALL tool will use this technology. Then, our target language (i.e., Arabic) was presented as another important topic. After that, many CALL tools for Arabic and other languages were reviewed. Some of these tools (e.g., Baldi) showed the usefulness of having a graphical representation of the vocal tract, which we will use in our CALL tool as one source of feedback. Other tools showed the value of having the synthesis of speech, which we will also use as another source of feedback in our CALL tool.

After this review of many CALL tools in this chapter, we will now open the door for our CALL tool in the next chapter (Chapter 3) and describe its architecture.
Chapter 3

Architecture of the CALL tool

3.1 System Architecture

The implemented tool involves a substantial amount of machinery, from driving the speech recogniser to animating the images of the vocal tract, synthesising the speech, and the integration of all this machinery is itself a challenging task. To illustrate this machinery, Figure 3.1 shows the system architecture of our CALL tool.

Figure 3.1 shows that when a learner pronounces a word, he produces a set of phones which will be processed by an ASR system (i.e., the speech recogniser). After that, the speech recogniser will match what the learner said to the nearest native speaker sounds which constitute our training data. If the pronunciation of the learner is poor, the sounds he produces may be more similar to some other set of phones than to the ones he intended. In other words, if the ASR thinks that the phones it received are different from the ones the learner intended, it will use this to provide feedback. Therefore, we set the recogniser to expect one of a set of minimal variations of the target word. If it receives one of the variations, it will provide a description (animated head, synthesised voice, and written explanatory text) of the difference between the target and uttered phones.

3.2 System Integration

The integration is a major issue in producing our diagnostic tool. In this task, we carried out a substantial piece of implementation which is done by integrating
Figure 3.1: System architecture
A number of existing pieces of software are mentioned below.

### 3.2.1 Identifying Mispronunciation by the HTK

The first piece of software is for identifying mispronunciation by driving the speech recogniser (i.e., the HTK) in which we use phonemes as terminal symbols instead of words. We carried out many experiments for training the HTK with native sounds, and we became more confident with the recognition accuracy, which improved and reached 100% for the minimal pairs we are using in the tool as it is presented to learners. More details on this section are provided in Chapter 4.

### 3.2.2 Animating the Vocal Tract

The second piece of software is for animating a sequence of images of articulatory positions. This piece of software takes the output of the HTK consisting of a set of phonemes and animates the images assigned to these phonemes in order to be a feedback for a learner. More details on this section are provided in Chapter 5.

### 3.2.3 Synthesising Utterances

Given that we have a set of phones which are what the system believes the user uttered, we can fairly easily generate a synthesised version of this phone sequence. This gives us a second source of feedback to give to learners. In this piece of software, the learner can play his voice, listen to his synthesised voice, and listen to the correct synthesised voice. More details on this section are provided in Chapter 6.

### 3.2.4 Instruction Feature

This feature displays an explanatory text of how the learner can pronounce the intended phoneme correctly. The displayed text shows the wrong phoneme and the right one. Furthermore, the text is accompanied with a picture which helps clarify the text and make it easy to understand. The effect of visual information (imagery) on text comprehension has received wide attention. Research on a second language (L2) has revealed that foreign words associated with images are learned more easily than those without associations [KH71]. A study conducted by Omaggio showed that using pictures helped to improve the reading comprehension...
of French students in college [Oma79]. Another study compared the effects of pictorial and textual information on foreign language learners, and the results showed that combination of text and picture was more effective than textual information alone or pictorial information alone [KFL99]. A study by Oxford and Crookall [OC90] stated that visual imagery helps learners absorb information more efficiently than just text alone because most learners can easily associate new information to concepts in memory by means of meaningful visual images. So, this feature forms a third feedback to Arabic learners. The written instructions provided by our CALL tool advise students on how to properly articulate the phoneme in case of mispronunciation as shown in Figure 3.2.
Figure 3.2: Articulatory advice
3.3 Language Lab Version

When a learner sits for a specific amount of time pronouncing Arabic words and listening to himself, this will lead to an improvement in his pronunciation due to the practice effect. In order to eliminate this effect during the evaluation of our tool, we developed a version to roughly replicate a language lab. This version allows the learner to look at the two animations, listen to the two synthetic forms, and read the instructions. However, it does not tell him whether he pronounced it right or wrong. With this version, we are attempting to find out whether learners have improved more than they would have performed with the same amount of practice and the same tools but without any indication of whether their pronunciation is right or wrong.

This version is integrated into our CALL tool in order to estimate the practice effect so that we can eliminate this effect as a contribution to the other versions.

3.4 Conclusion

This chapter can be considered as an entrance to the next three chapters (i.e., Chapter 4, Chapter 5, and Chapter 6). In this chapter, the architecture of our CALL tool showed the use of ASR technology which will be discussed experimentally (i.e., with the HTK) in more detail in the next chapter (Chapter 4). Moreover, the architecture illustrated that the animated vocal tract and synthesized speech are two sources of feedback to learners, which will be described in Chapter 5 and Chapter 6, respectively.
Chapter 4

HTK Experiments

Our pronunciation support tool (i.e., our CALL tool) is aimed to be a diagnostic tool for the pronunciation of non-native Arabic learners. We attempt to diagnose a learner’s ability to pronounce Arabic by using speech recognition to see whether what the learner says matches the proper pronunciation of the target or not. In general terms, the procedure is composed of creating native pronunciation models through the HTK and after that measuring the non-native responses against the native models. This requires models trained on native speech data in the target language (i.e., Arabic), and supplemented by a set of parameters for acoustic analysis which is useful in distinguishing native from non-native speech.

Non-native speakers find some sounds difficult to pronounce. This difficulty arises because of two reasons:

1. in Arabic, there are certain sounds which are different from those in other languages. Since the different languages use different sounds, the learner is not used to putting his articulators into the right positions to pronounce these difficult sounds correctly.

2. since these sounds are not distinguished in the learner’s native language, he will have difficulty hearing the difference between them. Thus not only does he find producing them difficult, he will find it hard even to know whether he has done it right or wrong.

Therefore, the conducted experiments were focused on the sounds which most non-native Arabic speakers have difficulty with. We know these sounds by consulting both linguists and Arabic teachers who interact with non-native Arabic speakers.
Since our CALL tool is based on the HTK toolkit, which is considered to be one of the most successful tools used in speech recognition research over the last two decades, this chapter presents our work on this toolkit, showing the needed files and the HTK commands which were used to carry out a number of experiments. These experiments are a major stage to help determine what will be needed in order to animate the vocal tract and to synthesise the speech as feedback to learners. We have investigated many factors (gender, words as terminal symbols, phonemes as terminal symbols) in order to improve the recognition accuracy as much as we can.

Using these experiments, the HTK, with a high recognition accuracy, is embedded into our CALL tool, which provides what a learner says when he uses our CALL tool. So, this chapter presents a part of the system architecture shown in Chapter 3. Figure 4.1 reveals this part as the following:

![Figure 4.1: HTK experiments](image)

Figure 4.1 shows that this chapter deals with the events that occur immediately after a learner attempts to pronounce the target word. So, we have done a lot of experiments with the HTK in order to recognise what the learner says with high accuracy. As illustrated from the figure, the output of this chapter (i.e., HTK experiments) is to have a word recognised accurately. To this end, we have conducted many experiments with the HTK leading to an idealised experiment with the best configuration. This output is, after that, used to animate the vocal tract and synthesise the speech which will be described in subsequent chapters.

In this chapter, we will describe some pilot experiments we carried out. Since these experiments are based on the HTK, we will first explain how to use the HTK. Then, we will give an account of our pilot experiments with their training data, platform, design, and results. These pilot experiments involved investigating
the following issues: training data based on male voice recordings, training data based on female voice recordings, training data based on word level, training data based on phone level. Once we have conducted these experiments, we realised that actually one of the problems we were having was that the voices were not recorded very clearly. Therefore, we have conducted some more experiments with new data for the purpose of having clear voice recordings.

4.1 Using the HTK

In our experiments, we roughly followed the steps mentioned in the HTK book \cite{YEG+06} in order to train the HTK as shown below. There were, however, a number of problems with the steps given in the book, so we also consulted the VoxForge site \cite{Vox}, and also made a number of modifications of our own. Furthermore, in order to be able to vary the experimental settings, we encapsulated these steps as a Python script. Section 4.1.1 gives a brief overview of the major steps of training the HTK and describes encountered problems and how we overcame them. Moreover, we describe our modifications, the necessary files, and the Python script that have been used for our experimentation.

4.1.1 The HTK training steps

Figure 4.2 gives the abstract picture of the major steps that we went through to train the HTK. These steps include: preparing the data, training the data, and evaluating the recogniser.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{htk-training-steps.png}
\caption{The HTK training steps}
\end{figure}
4.1.1.1 Data Preparation

Prior to commencing the HTK training, we prepared different important files (shown in Figure 4.3) which are compulsory to conduct the HTK experiments.

![Figure 4.3: Step 1: Data preparation](image)

In this step, we prepared a crucial file called **prompts file**. This file contains the transcriptions of all our recorded words (i.e., the training sentences). These transcriptions are equivalent to the textual information stored inside the file which includes audio files' names and their corresponding transcriptions. If we move deeper into the file, we can see that each line of the prompts file represents one audio file and its transcription. Furthermore, the first column corresponds to the audio file name and the subsequent columns on the same line correspond to the text transcription of this audio file (i.e., SENT-START, the target word, and SENT-END). Figure 4.4 shows this format of our prompts file.

Similar to the prompts file, there is a file called **testprompts** (shown in Figure 4.5). The difference between the two files is that the prompts file is used for the purpose of training the HTK, while the testprompts file is used in the testing phase for the recognition evaluation.

As shown in Figure 4.4, the prompts file can be used as guidance on which words we need to record for our speech files (i.e., **wav files**). Our recorded files need to be managed and organised in such a way that all files are kept aligned with their transcriptions and tied up to a specific folder so that we can run the HTK experiment. HSLab, which is the default recording tool in the HTK, and standard speech recording tools do not give this facility. Therefore, we

---

1 SENT-START and SENT-END represent silences occurring at the start and at the end, respectively, during recording.
developed a recording tool called “SpeechRecord” designed to perform this task. “SpeechRecord” reads the prompts file line by line in order to be compatible with the HTK, presents the transcription of the prompted word to be recorded by the subject (i.e., the native speaker), and saves them in wav files with the right names and in the right place. Furthermore, another significant advantage of “SpeechRecord” over HSLab and other standard recording tools is the ability to include the Arabic form in order to make it easier for native Arabic speakers to use this adapted recording tool. These valuable features are unavailable in HSLab and other recording tools, and hence we moved to our developed “SpeechRecord” tool, which is shown in Figure 4.6 below.

After having the wav files, it is necessary to convert them into MFCC format, which stands for Mel-Frequency Cepstral Coefficients (MFCCs), in order to be understood by the HTK. MFCCs are common features used in many speech recognition systems. To do the conversion to MFCC, the HTK has a tool called \texttt{HCopy} which parameterises the wav file and stores the result in mfc file. \texttt{HCopy} takes a file called \texttt{codetr.scp} (shown in Figure 4.7) as input containing a list of each wav file and its corresponding mfc file.

Another substantial file which we prepared was the \textbf{grammar file} that defines all of the legal word sequences. The HTK gives a grammar definition language for determining a suitable grammar. This grammar is stored in a file containing a set of variables followed by a regular expression which gives an account of the

\begin{verbatim}
1 */sample1 SENT-START youbhir SENT-END
2 */sample2 SENT-START youbhir SENT-END
3 */sample3 SENT-START youbhir SENT-END
4 */sample4 SENT-START youbhir SENT-END
5 */sample5 SENT-START youbhir SENT-END
6 */sample6 SENT-START youbhir SENT-END
7 */sample7 SENT-START youbhir SENT-END
8 */sample8 SENT-START youbhir SENT-END
9 */sample9 SENT-START youbhir SENT-END
10 */sample10 SENT-START youbhir SENT-END
11 */sample11 SENT-START Darb SENT-END
12 */sample12 SENT-START Darb SENT-END
13 */sample13 SENT-START Darb SENT-END
14 */sample14 SENT-START Darb SENT-END
15 */sample15 SENT-START Darb SENT-END
16 */sample16 SENT-START Darb SENT-END
17 */sample17 SENT-START Darb SENT-END
18 */sample18 SENT-START Darb SENT-END
19 */sample19 SENT-START Darb SENT-END
20 */sample20 SENT-START Darb SENT-END
\end{verbatim}

Figure 4.4: Prompts file
words to recognise. Figure 4.8 shows a grammar which was used in one of our experiments. We will discuss other grammars as they arise later.

In Figure 4.8, the dollar sign ($) denotes the starting of a variable name that holds what we want to pronounce, the vertical bar (|) represents an alternative to the word which is picked for pronunciation, and the semicolon (;) indicates the end of variable values that are equal to words intended to be pronounced. SENT-START and SENT-END identify the silence model which can be found at the beginning and end of sentences.

As shown in Figure 4.8, the grammar is written in pairs where each pair contains two words that are very close in pronunciation. This way helps us to make the grammar clear when we look at it making sure that it consists of a set of words that are minimal pairs. At the end of the grammar, all minimal pairs, any of which a learner can be asked to pronounce, are grouped together. In other words, this grammar describes an ‘anyword’ grammar, but the way we write it presents the minimal pairs clearly and later helps to set the recognition grammar to choose one of these minimal pairs leading to a high recognition accuracy. Our final level of recognition accuracy comes from directing the system’s attention to a specific minimal pair with the help of this way of writing the grammar.
One more important file for data preparation is the dictionary file. The pronunciation dictionary is one of the essential parts of large vocabulary SR systems. This dictionary works as the mediator between the acoustic model and the language model in SR systems. It consists of all language words as well as their basic sounds (i.e., phones). In SR systems, each word is converted into a sequence of phones using this dictionary [AEAG+08, You96].

The HTK requires a pronunciation dictionary file containing a sorted list of the required words, which are the words in the prompts and grammar files, and their pronunciation. We produced a small source Arabic pronunciation dictionary called lexicon. Figure 4.9 shows the structure of the generated dictionary.

The contents of lexicon (Figure 4.9) consist of two columns. The first column represents what is called “Buckwalter transliteration” with an ASCII transliteration scheme representing an Arabic word pronunciation written in English (see Appendix [7]). The reason for using Buckwalter transliteration is that the HTK
does not allow us to put an Arabic script for the words in the dictionary file. In other words, the HTK assumes that every pronounced word in the dictionary file must be transcribed in ASCII format. The second column refers to what is called “SAMPA” (Speech Assessment Methods Phonetic Alphabet) which is a phonetic script using ASCII characters based on the International Phonetic Alphabet (IPA) (see Appendix [H]).

We modified “SAMPA” in order to meet our needs taking advantage of the fact that the HTK does not require us to use a predefined set of phonemes (phonemes names). Therefore, we are allowed to choose any phoneme name we need in the dictionary file. This feature is helpful because some of the phonemes we need (particularly emphatic ones) do not have equivalents in other languages like English (i.e., do not occur in standard SAMPA). So, we added these phonemes into our modified SAMPA. Moreover, some IPA symbols are not accepted by the HTK, and so we modified them to other symbols. For example, we modified $\delta$ to

---

<table>
<thead>
<tr>
<th>Word</th>
<th>Pronunciation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sal</td>
<td>$D^\prime a l$</td>
</tr>
<tr>
<td>furap</td>
<td>$D^\prime u r a h$</td>
</tr>
<tr>
<td>DAl</td>
<td>$d^\prime a l$</td>
</tr>
<tr>
<td>Darb</td>
<td>$d^\prime a r b$</td>
</tr>
<tr>
<td>SAbir</td>
<td>$s^\prime a b i r$</td>
</tr>
<tr>
<td>SHIFT-END</td>
<td>$s i l$</td>
</tr>
<tr>
<td>SHIFT-START</td>
<td>$s i l$</td>
</tr>
<tr>
<td>TALib</td>
<td>$t^\prime a l i b$</td>
</tr>
<tr>
<td>Tyn</td>
<td>$t^\prime i n$</td>
</tr>
<tr>
<td>ZAbir</td>
<td>$z^\prime a b i r$</td>
</tr>
<tr>
<td>Zal</td>
<td>$z^\prime a l$</td>
</tr>
<tr>
<td>bayd</td>
<td>$b a y d^\prime a b$</td>
</tr>
<tr>
<td>dark</td>
<td>$d a r b$</td>
</tr>
<tr>
<td>dub</td>
<td>$d u b$</td>
</tr>
<tr>
<td>k severe</td>
<td>$k s t a b a t$</td>
</tr>
<tr>
<td>miqas</td>
<td>$m i q a s$</td>
</tr>
<tr>
<td>qsw</td>
<td>$q a w c$</td>
</tr>
<tr>
<td>syyid</td>
<td>$s a y y i d$</td>
</tr>
<tr>
<td>silence</td>
<td>$s i l e n c e$</td>
</tr>
<tr>
<td>siradab</td>
<td>$s i r d a b$</td>
</tr>
</tbody>
</table>

Figure 4.9: Pronunciation dictionary file
(D^\circ) for the Arabic word ٥ُ ("al) (Figure 4.9) as the symbol δ is not accepted by the HTK (see Appendix[4] for all modifications).

In addition to the modification of SAMPA, Buckwalter is also modified. The reason why we have made some changes to Buckwalter is that the grammar used with the HTK treats certain characters such as dollar sign ($) as special characters. This character denotes the starting of a variable name in the grammar file (shown in Figure 4.8). However, in standard Buckwalter, it is a character corresponding to the Arabic character شيء /ʃ/ which is usually represented in English writing with /sh/, as in ‘ship’. This means we cannot use ($) as an entry of textual transliteration in Buckwalter because this will cause a clash with the special meaning of ($) in the HTK (i.e., considering that ($) is from the grammar not as the character of alphabet). Therefore, we modified ($) to be (ˆs) in our modified Buckwalter to avoid having one symbol of different meanings which causes conflict (see the modifications of Buckwalter in Appendix[4]).

The dictionary file (dict) is created by using one of the HTK tools called HDMan tool where the source dictionary lexicon is searched to extract pronunciations for each word in the prompts file. This means that HDMan is used not only against a training dictionary, but also here against the entire dictionary (i.e., lexicon). In order to process the merged dictionaries, HDMan provides commands such as AS command to append, RS command to remove stress marking, and MP command to merge and rename. These commands are written in a file called global.ded. The script prompts2wlist is used to extract the training word list (wlist) automatically from the prompts file. Figure[4.10] summarises the job of the HDMan tool.

By the end of this step (i.e., data preparation), it is time now to train the system and construct a set of HMMs using the provided data. The following section describes the training phase and the run of a python script for our experimentation.

4.1.1.2 Training the HTK

Why do we use a script? We developed a Python script to run all the training HTK commands at once instead of executing each command individually at the command line. This is because we carried out many experiments repeatedly with different training sets, different grammars, and different dictionaries. Using a script helped us to control the experiments. In this script, we integrated all the
tools needed for the HTK, such as perl, hed, and led scripts, and then we ran them from within the script by executing them as subprocesses. For example, the perl script ‘prompts2wlist’ can be executed in the command line:

$s perl prompts2wlist prompts wlist

However, in our script, we created ‘prompts2wlist’ and executed it from within the main script:

create_prompts2wlist(exptdir)
test(["perl",exptdir+"prompts2wlist",exptdir+"prompts", exptdir+"wlist"])

where exptdir is a variable which denotes the path of the working or experimental directory that contains all generated files, and wlist is a file that contains a word list from the provided sentences that are stored in the prompts file. This helped us to make sure that there were no missing files of the HTK tools as they are generated automatically from within the main script, and thus gave us more control over the HTK.

There was a problem which we faced in running the script which can be described as follows:

- The HTK sometimes refused to accept a file that had been given as part of the training set as a wav file, even though it accepted that wav file when
it was isolated from the training set. It was impossible to predict which files would cause a problem, and indeed when we removed one problematic file, it often happened that another which had previously been processed satisfactorily would become problematic. In order to cope with this, the script monitors the output of the relevant HTK tool and collects the names of the files that have been flagged by the tool. These files are then eliminated from the training set and the script is rerun. Occasionally, the second pass reveals future problems, but in general it will run to completion after a couple of passes.

The final script, which is included as Appendix B, is fairly substantial, over 3500 lines of Python, of which 530 lines are the prototype scripts (perl, hed, and led files). Figure 4.11 shows a high level description of what our script does in order to have the HTK trained.

4.1.1.3 Recogniser Evaluation

After completion of the training process, the performance of the recogniser (i.e., the HTK) can be evaluated by running an HTK tool called “HResults” which gives us the recognition accuracy of the HTK using our collected training data as described in the previous section. HResults compares between recout.mlf file, which contains the recognition results, and testref.mlf file which is the corresponding reference transcriptions. After conducting this comparison, HResults will print out the analysis of the HTK’s results as follows:

```
======================================= HTK Results Analysis ======================================
Date: Tue Oct 18 12:02:21 2011
Ref : C:\Documents and Settings\alsabaam\expPaper\testref.mlf
Rec : C:\Documents and Settings\alsabaam\expPaper\recout.mlf

Overall Results

SENT: \%Correct=40.16 [H=257, S=383, N=640]

WORD: \%Corr=72.62, Acc=64.76 [H=3268, D=318, S=914, I=354, N=4500]
```

The HTK results has two main parts: the first part provides information about the date and time of the experiment with the names and locations of the used files; the second part provides statistics in sentence and word levels. The line which starts with SENT gives the sentence level statistics and shows that
the total number of utterances is 640. Out of these utterances, 257 sentences (H) were recognised correctly (i.e., \%Correct = 40.16\%), while 383 sentences (S) were substituted. The next line starting with \texttt{WORD} denotes that 4500 words were tested, and the number of correctly recognised words is 3268 (i.e., 72.62\%). The results show that there were 318 deletion errors (D), 914 substitution errors (S), and 354 insertion errors (I). As can be noticed, the percentage of \texttt{Acc} (i.e., 64.79\%) is lower than the percentage of \texttt{WORD Correct} (i.e., 72.62\%). The reason for this lowness is that \texttt{Acc} percentage takes into account the insertion and deletion errors (if there are any) which are ignored by the \texttt{WORD Correct} percentage.
In a lot of the HTK literature, the HTK word-level outputs are misleading because the accuracy is always overestimated. The reason of this overestimating is that SENT-START and SENT-END, which denote silences, in the prompts file (see Figure 4.4) are always counted as correct words. So, when we compute the accuracy, we ignore SENT-START and SENT-END (i.e., silences) in our calculation. For example, in the previous output of HResults, the recognition accuracy at the word level is \( \frac{H}{N} = \frac{3268}{4500} = 72.62\% \), but we perform a further calculation to ignore silences and get more accurate accuracy. The number of silences is computed by \( N \times 2 \) since there are two silences (i.e., SENT-START and SENT-END) occurring in each sample in the prompts file. This means the net accuracy will be \( \frac{(H-N \times 2)}{(N-N \times 2)} = \frac{3268-(640 \times 2)}{4500-(640 \times 2)} = \frac{1988}{3220} = 61.7\% \).

### 4.2 Experiments

In our experiments, we were concerned with developing a speaker-independent Automatic Speech Recognition (ASR) system that can deal with all the sounds that English people find difficult to pronounce including the emphatic\(^2\) consonants of the Arabic language and other sounds that non-native speakers have difficulty with. The database used for training consists of 3200 tokens in total, created with the help of 20 Arabic native speakers who pronounce each token for 5 times. The point of these experiments was to investigate the following questions:

- What factors may affect the recognition accuracy?
- What is the optimal granularity of the states of the HMM in terms of word level and phoneme level?

After this investigation has found which configuration of settings of the HTK produces the highest recognition accuracy, we then use that configuration in our CALL tool to help non-native Arabic speakers to pronounce these difficult sounds correctly.

This experimental study shows that phoneme confusability is a major source of difficulty for developing an Arabic ASR. Furthermore, it highlights the factors

\(^2\) pharyngealisation or coloured sounds
that may affect the ASR system’s performance negatively or positively. Therefore, these experiments were designed using minimal pairs of lexical items where the only variable was the presence or absence of confusable consonants. The experiments also used different speakers (in terms of age, gender, and nationality) in order to identify factors that affect the recognition accuracy.

Because we were dealing with the emphatic consonants of Arabic, which are the most confusable sounds and have no counterpart in English, in these experiments, the following section will give a brief introduction to these consonants.

4.2.1 Arabic Unique Consonants

Arabic has some unique consonants which do not exist in any other languages. For instance, there are four emphatic consonants in Arabic: [dˤ], [tˤ], [sˤ], and [ðˤ]. Each emphatic consonant has an equivalent plain consonant, [d], [t], [s], and [ð], respectively. This can be shown in the following examples of minimal pairs (two words with different meanings when only one sound is changed):

\[
\begin{align*}
\text{\textit{darb}} & \text{ (means “path”) vs. } \text{\textit{dār}} & \text{ (means “hitting”) } \\
\text{\textit{ti:n}} & \text{ (means “fig”) vs. } \text{\textit{ti:n}} & \text{ (means “clay”) } \\
\text{\textit{ḍal}} & \text{ (means “cringed”) vs. } \text{\textit{ḍal}} & \text{ (means “still”) } \\
\text{\textit{násaba}} & \text{ (means “imputed”) vs. } \text{\textit{nāsaba}} & \text{ (means “erected”) }
\end{align*}
\]

The emphatics and their non-emphatic counterparts have many features in common. Table 4.1 provides a description of these two groups in terms of place and manner of articulation.

Different sounds are made by placing the articulators in different positions. Phonetic descriptions are usually provided using a set of labels referring to these positions. Table 4.1 shows the descriptions of the emphatic Arabic sounds and their non-emphatic counterparts using the standard set of labels. It is clear from Table 4.1 that in every case the two members of each pair have the same description.

Nonetheless, the different sounds are produced by different configuration of the articulators. There is, after all, no other way of producing different sounds. In other words, the usual labels for places of articulation of the two groups are
4.2. EXPERIMENTS

Table 4.1: Emphatic and non-emphatic sounds

<table>
<thead>
<tr>
<th>Properties</th>
<th>Place of Articulation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inter-dental</td>
</tr>
<tr>
<td>Plosive Voiced</td>
<td>Emphatic</td>
</tr>
<tr>
<td>Plosive Unvoiced</td>
<td>Non-emphatic</td>
</tr>
<tr>
<td>Plosive Emphatic</td>
<td>-</td>
</tr>
<tr>
<td>Plosive Non-emphatic</td>
<td>-</td>
</tr>
<tr>
<td>Fricative Voiced</td>
<td>Emphatic</td>
</tr>
<tr>
<td>Fricative Non-emphatic</td>
<td>[ð]</td>
</tr>
<tr>
<td>Fricative Emphatic</td>
<td>-</td>
</tr>
<tr>
<td>Fricative Non-emphatic</td>
<td>-</td>
</tr>
</tbody>
</table>

not fine-grained enough which means that the differences between them is not usually discussed as in other languages.

This issue between the emphatic consonant and its non-emphatic counterpart was addressed early by the famed Arab grammarian Siybawayh (796 A.D.) who said that the difference between the two groups is in the place of articulation where the emphatics have two places of articulation. He called the emphatic consonants (Alhuruf AlmuTbaqa) ‘covered letters’ because they are produced by having the tongue covering the area extending from the main place of articulation towards the palate. This has been verified by modern studies which confirm that the main articulatory difference between the two groups is that the articulation of the emphatic consonants involves a secondary articulation where the tongue root is retracting which results in a narrowing in the upper portion of the pharynx accompanied by a retraction in the lower part of the pharynx’s interior wall [Hes98] [AA70].

Figure 4.12 shows the tongue configuration during the articulation of the emphatic consonants and the articulation of the non-emphatic counterparts based on the acoustic description presented in [Hes98] and [AA70].

Al-Omar [AO09] pointed out that wherever emphatic sounds occur in a word, emphasis (i.e., pharyngealisation) may spread to the adjacent sounds in the same word. Spectrogram representations of a minimal pair: دَرَبَ /d$^t$arb/ and دَرَب /darb/ are shown in Figure 4.13 and 4.14, respectively. Figure 4.13 contains an
emphatic consonant ض /d⁵/, while Figure 4.14 contains its non-emphatic counterpart د /d/. After comparison between both spectrogram, the formants are clearer and less spread in Figure 4.14 than in Figure 4.13 as the latter contains an emphatic sound.

After collecting some information of difficult aspects for some Arabic consonants (i.e., emphatic and their non emphatic counterparts), the following sections describe our experiments done on such consonants. These experiments not only involved emphatic and their non emphatic counterparts, but also involved other confusable consonants found in the Arabic language.
4.2. EXPERIMENTS

4.2.2 Training Data and Platform

A set of real Arabic words was chosen in which half contain confusable consonants and the other half contain their counterparts (i.e., the ones that are very close to them in pronunciation). This selection of Arabic words is based on the consultation of both linguists and Arabic teachers who interact with non-native Arabic students. The data contains a set of words where the confusable sound and its counterpart consonant occur within the same context (minimal pairs). These words are difficult for students learning Arabic to pronounce. Table 4.2 contains 7 minimal pairs (i.e., 14 Arabic words) that have been used in our experiments.

Table 4.2: Arabic minimal pairs

<table>
<thead>
<tr>
<th>Arabic Word</th>
<th>Counterpart</th>
</tr>
</thead>
<tbody>
<tr>
<td>يصبر</td>
<td>يسبر</td>
</tr>
<tr>
<td>ضرب</td>
<td>درب</td>
</tr>
<tr>
<td>طين</td>
<td>تين</td>
</tr>
<tr>
<td>ظل</td>
<td>ذل</td>
</tr>
<tr>
<td>يأمر</td>
<td>يعمر</td>
</tr>
<tr>
<td>همزة</td>
<td>حمزة</td>
</tr>
<tr>
<td>كل</td>
<td>قل</td>
</tr>
</tbody>
</table>

In Table 4.2, each Arabic word differs from its counterpart in only one phoneme. So, with each Arabic word using the technique of minimal pair, we have two phones that are confusable and hard to produce by non-native learners. The first
phone occurs in one word; the other occurs in its counterpart. For example, the Arabic word ﺩ ﺱ (yaSbir) differs from its counterpart ﺱ ﺱ (yasbir) in only one phone, which is located at the third position in both words. Therefore, the target phone in the first word ﺩ ﺱ (yaSbir) is ‘S’; the other one in the second word ﺱ ﺱ (yasbir) is ‘s’. This means, based on our careful selection of these words, that the two phones: ‘s’ and ‘s’ are confusable, and learners have difficulty to produce the one which does not exist in his mother language (i.e., ‘S’) and he is likely to pronounce its counterpart instead (i.e., ‘s’).

Going through the rest of words in the table: ‘D’ is confused with ‘d’; ‘T’ is confused with ‘t’; ‘Z’ is confused with ‘z’; ‘Q’ is confused with ‘E’; ‘h’ is confused with ‘H’; and ‘k’ is confused with ‘q’. These phones with their counterparts constitute the most difficult phones in Arabic which we are trying to help students learning Arabic pronounce correctly.

As mentioned above, we obtained our material by asking 20 native Arabic speakers to repeat each word 5 times. The total number of utterances was 3200 tokens collected from these native speakers from different nationalities: three from Kuwait, three from Egypt, three from Syria, two from Saudi Arabia, two from Yemen, two from Jordan, two from Sudan, one from Bahrain, one from Iraq, and one from Palestine. Twelve of these speakers were male, and eight were female. They were aged between 25 and 35.

Although SR technology is becoming more advanced, the recognition of different accents of speech remains a largely unsolved issue despite its effectiveness. Since we do not have a large number of speakers from any specific accent group, it did not seem sensible to split them into accent groups, thus we simply grouped each one into a single set. However, we know that there are major differences between male and female speakers, and thereby we have carried out experiments based on gender.

To ensure that we obtain reliable results, we used 5-fold cross-validation as a way of interpreting the results. This involves randomly partitioning the data into 5 equal size subsets, performing the training on 4 subsets and validating the
4.2. EXPERIMENTS

This process is repeated 5 times with each of the 5 subsets used only once as the validation data. The results from the 5 folds then can be averaged to compute a single estimation. The advantage of this method over the standard evaluation approach is that all observations are used for both training and testing which gives more robust testing for experiments with small data sets.

4.2.3 Designed System

The system developed in these experiments is designed to recognise Arabic phonemes with respect to the confusable consonants using the HTK. This includes two major processing stages.

The training phase is concerned with initialising and estimating the maximum likelihood of the HMM parameters. It requires having the speech data supplemented by their phonetic transcription. A set of HTK tools is used for the purpose of building a well-trained set of HMMs.

The second stage is the testing stage, whose function is to use a network of HMMs and to find the most likely sequence of phonemes and hence to return a transcription for each speech signal.

In this experimental study, we conducted three main experiments. The first experiment (Section 4.2.4) shows how using different speech units (phoneme level or word level) can affect the accuracy of recognition. From this experiment, we analysed the confusion matrix to identify the most confusable phonemes by the recogniser. In the next experiment (Section 4.2.5), the system was trained and tested by only male speakers and then only female speakers with the same size of data set. The aim of this division is to determine the influence of having the same gender of speakers on the recognition accuracy. The third experiment (Section 4.2.6) is the one for which we have the highest accuracy, when we replaced our collected data with new one. We used the model that resulted from this final experiment in our CALL tool to help Arabic learners to distinguish between difficult sounds.
4.2.4 Experiment 1: Word Level vs. Phoneme Level

4.2.4.1 Experimental Design

What are word and phoneme levels? The HTK, like all speech recognition engines, requires a dictionary file that contains words and their phonetic transcriptions, and a grammar file that defines the sequences of all allowed words (i.e., constrains what is going to be said). However, there is a question about what the terminal symbols should be: should they be words or phonemes? Generally, words are the main speech units that are used to build HMMs. Using words as terminal symbols can be called word level. This typical choice of terminal symbols (i.e., words) can be altered to phonemes as basic units in building HMMs (i.e., phoneme level). Using phonemes as terminal symbols has been suggested in previous studies such as [Alo10], [AA10], and [Rab89]. Alghamdi et al. [AA10] states that phoneme-based models are good at capturing phonetic details and they can be used to characterise formant transition information. Alotaibi [Alo10] confirms that phoneme level models are superior to word level. Moreover, he reports that for any word, we can get various vocalisations because of the following: the speaker behaviour, speaker emotion, speaker educational level, dialect, native language, acquired second language, word context, and Arabic language grammar rules. All these factors affect the production of different words that must have different HMMs.

All our experiments were carried out using minimal pairs represented as isolated words in the prompts file. As shown in Figure 4.4, each sentence is actually one word. At the word level, the HTK thinks of each word as one word of the sentence and each sentence as a group of words (which is one word in our case). At the phoneme level, the HTK thinks of each phoneme as one word and each sentence as a group of sets of phonemes where each set is treated as one word (i.e., each sentence is one phoneme set in our case).

This experiment is aimed at investigating how using different speech units (phoneme level or word level) can affect the accuracy of the recognition. It is actually composed of two experiments: the first one was conducted at word level; the other one was carried out at phoneme level. Word-level experiments were carried out with twenty native speakers, using a grammar with words as terminal symbols and a dictionary file at the word level as shown in Figures 4.15 and 4.16.
Phoneme-level experiments were carried out with the same speakers, using a grammar with phonemes as terminal symbols and another version of the dictionary file. This new version of the dictionary file is produced by adding all used phones to the original dictionary file in order to make it usable with the grammar file at the phoneme level. Both files, grammar and dictionary, are shown in Figures 4.17 and 4.18. These speakers were asked to record 32 words and repeat the recording of each word five times. So, we have 3200 utterances collected in total.
4.2.4.2 Results

The recognition accuracy was 22.6% using the word level grammar and 39% using the phoneme level grammar.

Table 4.3 shows an extract from the confusion matrix for recognised sounds of this experiment at phoneme level, plotted against target sounds. This table shows the main sounds where there is a substantial level of confusion, but there is a distribution of these sounds among other sounds which scatter over a wide variety of other sounds. By analysing the confusion matrix results from a phoneme level experiment, we can confirm that the emphatic sounds pose a major source of difficulty in recognising Arabic speech. Apart from (ش[s]) and its non-emphatic...
counterpart (س[s]), it is apparent from Table 4.3 that the recognition of the emphatic sounds and their non-emphatic counterparts is below 50%. The poorest accuracy was found in the (ش[d]) sound (20%) and it was confused most often with the emphatic sound (ط[ð]) (16.2%). This confusion may result from the fact that most Arabic speakers tend to pronounce the sound (ش[d]) as (ط[ð]) [Mas98]. It can also be observed that the (ط[d]) sound shows relatively high confusion with the sound (ت[t]) and vice versa. This might be due to the fact that these two sounds share all the features except that the (ط[d]) is voiced and the (ت[t]) is voiceless.

<table>
<thead>
<tr>
<th>Prompted Phones</th>
<th>Recognised As</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[s]</td>
<td>[s]</td>
<td>[d]</td>
<td>[d]</td>
<td>[t]</td>
<td>[t]</td>
<td>[ð]</td>
</tr>
<tr>
<td>[s]</td>
<td>52.80%</td>
<td>6.40%</td>
<td>44.10%</td>
<td>6.20%</td>
<td>48.10%</td>
<td>12.50%</td>
<td>42.50%</td>
</tr>
<tr>
<td>[s]</td>
<td>8.70%</td>
<td>51.20%</td>
<td>20%</td>
<td>5%</td>
<td>48.10%</td>
<td>7.30%</td>
<td>16.20%</td>
</tr>
</tbody>
</table>

Table 4.3: Confusion matrix for emphatic sounds

4.2.4.3 Conclusion

As mentioned earlier, the word level means that the HTK considers each word as one word of the sentence and each sentence as a group of words which is one word in our case (i.e., isolated words system). However, as mentioned in Experiment 1, the phoneme level means that the HTK considers each phoneme as one word and each sentence as a group of sets of phonemes where each set is treated as one word (i.e., each sentence is one phoneme set in our system of isolated words). This makes it difficult to compare the two approaches. To compare word level experiments and phoneme level experiments, we made the measurement at the sentence level accuracy, where sentence level accuracy in both is equivalent to word accuracy; since what we did is isolated word experiments. Figure 4.19 below highlights that our measurement of recognition accuracy is at sentence level not at word level.
CHAPTER 4. HTK EXPERIMENTS

To show the difference between the measurement at word level and sentence level, assume that a learner attempts to pronounce “يا سبي” (yaSbir) as a target word. Considering the measurement at word level (shown in Figure 4.20), when the learner performs the task successfully, the recognition accuracy of this word will be 100% at both phoneme and word levels. However, if he mispronounces the target word saying, for example, “يا سب” (yasbir) instead of “يا سبي” (yaSbir), the recognition accuracy will be 0% at the word level, but 83% at the phoneme level which is misleading in providing a very high accuracy for experiments at phoneme level.

Therefore, we moved to do the measurement at sentence level as shown in Figure 4.21. This figure indicates that if the learner pronounces the target word correctly, the recognition accuracy will be 100% at both word and phoneme levels. Otherwise, the recognition accuracy will 0% at both levels regardless of how many phonemes are correct at the phoneme level.

By comparing the sentence level accuracies in the two experiments and after applying the 5-fold cross-validation way (discussed in 4.2.2), the recognition accuracies were 39% and 22.6% at the phoneme level and word level, respectively. This result supports the observation that using phonemes as the HMMs is superior for limited vocabulary ASR systems as discussed by Alotaibi [Alo10]. For this reason, the phoneme level was used as a baseline system for the forthcoming experiments.

Although the word-level scoring can help the student to determine the mispronounced word, this might not be sufficient to give him/her a helpful clue on how
4.2. EXPERIMENTS

Figure 4.20: Word level measurement of recognition accuracy

Figure 4.21: Sentence level measurement of recognition accuracy
to improve his pronunciation. At the phoneme level, the system is in a position to provide detailed error explanations for pronunciation improvement, and to guide the learner into specifically tailored exercises for practice according to his current needs. This is reiterated by Menzel et al. [MHM+01] when they developed their pronunciation training system.

4.2.5 Experiment 2: The effect of Gender

4.2.5.1 Experimental Design

Training the HTK with random and different speakers who have many kinds of variations will make the recognition task much more complicated. For example, every speaker has his own vocal anatomy which ends in producing unique speech sounds. Furthermore, speaker’s gender, accent, age, and speaking style are all major sources of speech variation.

In this experiment, we attempt to find out the effectiveness of one of these sources of speech variation, which is speaker’s gender, on the performance of the recogniser (i.e., the recognition accuracy). Typically, the more data we have, the better training models will be. In this experiment, we attempt to answer the following question: will splitting a given amount of data into gender subsets affect the recognition accuracy? In other words, we attempt to check whether having all the data to make one composite model is providing higher accuracy than having less data but splitting it into two models (i.e., male and female).

This experiment was performed at phone level and it consisted of two experiments. The first experiment was carried out using twelve male native speakers as a first step to examine the effect of speaker’s gender. The male speakers were asked to record 32 words, repeating five times for each one. So, in total, we have collected 1920 tokens of male recordings.

As a second step to examine the effect of speaker’s gender, we carried out the second experiment at phone level in order to investigate the effect of female speakers on the recognition accuracy. The experiment was performed using eight female speakers. They have been asked to record their voices for 32 words and repeat the recording of each word five times. This means that we have 1280 wav files of female voices.
4.2. EXPERIMENTS

4.2.5.2 Results

The results show that the HTK had an accuracy of 40.2% for our investigation of male factor, while the HTK provides an accuracy of 45% for our investigation of female factor.

4.2.5.3 Conclusion

The answer to the question: “Does splitting a given amount of data in gender subsets affect the recognition accuracy?” is yes. After differentiating the complete data into two groups based on gender, the recognition accuracy is improved from 39% to about 42.6% (i.e., the average of the two accuracies 40.2% and 45 of male and female, respectively).

4.2.6 Experiment 3: Better training data

4.2.6.1 Experimental Design

We aimed to use the HTK to distinguish between minimal pairs in order to teach non-native speakers how to sound like native speakers by giving them feedback about their pronunciation. If the HTK cannot distinguish minimal pairs very well, we cannot afford to use it to provide learners with feedback about their pronunciation. Therefore, we put considerable effort into ensuring that the HTK is as accurate as possible on minimal pairs. It is not clear what level of accuracy would be satisfactory for this task. However, in this experiment, we obtained 100% of recognition accuracy under the constraints (described in more detail in Section 4.2.6.2), which we would apply in our CALL tool. This led us to use the model that resulted from this experiment in our CALL tool.

In this experiment, we replaced the training recording files (i.e., wav file) with new wav files by using a high quality microphone with the same native speakers. This microphone produces pristine, studio-quality recordings with ease of use, and uses a technology called tri-capsule, which gives four distinct pattern modes to choose from. The mode which we chose is called cardioid mode, which records sound sources that are directly in front of the microphone, delivering a high-quality sound. All new recordings, as the old ones, were done individually in a quiet room on a desk and a laptop computer. The room was ordinary not acoustically prepared. Furthermore, we added five more native speakers to our training data. This final experiment was performed at the phoneme level using
25 male and female native speakers. Each speaker is asked to record 32 words and repeat the recording of each one five times, thereby having 4000 wav files as a training data.

4.2.6.2 Results

The recognition accuracy of the HTK was improved and reached 77%. However, this recognition accuracy (i.e., 77%) is over the complete set of words that we used as confusable.

In our CALL tool, a learner is provided with a single word to pronounce and his pronunciation is assessed whether what he pronounces is the target word or the one specified confusable word. As such, in the actual target application, the grammar consists solely of one word and its confusable minimal pair. This would not be a realistic grammar in most domains, but it is appropriate in our domain as that is what we attempt to do for diagnosing the user pronunciation. When we moved to this kind of grammar, the recognition accuracy reached 100%.

To state the matter differently, remember in Figure 4.8, the grammar is written in pairs where each pair contains the word and its confusable counterpart. This grammar can be seen as ‘anyword’ grammar, but it was written in a way that presents the minimal pairs clearly so that it helps to set the recognition grammar to select one of these minimal pairs. So, we constrained what was expected to be heard to be either the target word or the one that it could be confused with (i.e., single minimal pair).

Although this constraint is not realistic for general purposes, it works well for our purpose to have a very high accuracy over the minimal pairs which we use in our CALL tool. The grammar is changed from anyword grammar to be one of a minimal pair which we actually use in our CALL where learners are prompted to pronounce words to explore whether what they say is the target word or its minimal pair (i.e., its counterpart). For example, if a learner is prompted to pronounce ضرب (Darb), the recognition grammar is switched from ‘anyword’
4.2. EXPERIMENTS

The method above provides a facility to perform very targeted recognition. Under those circumstances, the system reached an accuracy of 100% with native speaker data. So, this very high accuracy is what we need for our purposes and is what led us to believe that we would be able to use this system to help improve the pronunciation of Arabic learners. Therefore, the model obtained using the data used for Experiment 3 is the one that was used in our CALL tool.

As compared to the study which was done using an Arabic version of Baldi called Badr [OCM05] (described earlier in Chapter 2 in Section 2.4.3), the reached accuracy of recognition using Badr is 54%. An enormous range of scores have been published, where the variation depends upon various parameters including the grammars and the size of the vocabulary. We can afford to have a constrained grammar and set of vocabulary which enable us to obtain this very high accuracy of recognition (i.e., 100%). Our claim is not that we are better at training the
HTK than others, but if we simplify the problem in a way which is compatible with our overall task, we can obtain this good result.

4.2.7 Summary

In this experimental study of our research, the Arabic emphatic consonants were investigated from the ASR perspective. The experiments confirmed that those consonants are hard to recognise. The experimental study found that the [dˤ] consonant is the most confusable consonant among the emphatics. This refers to the acoustic confusability in addition to the difficulty that native speakers find in pronouncing this sound. The research also showed that using phonemes as the acoustic units can give better results when compared with using words. This finding supports previous research into this area which indicated that the phoneme level HMMs are superior in limited vocabulary ASR. The research shed light on the influence of gender as it found that splitting the data into two classes (i.e., male and female) yields higher recognition accuracy than having all data together, even though each class of data is smaller than the whole data. Finally, we have obtained a very high recognition accuracy (i.e., 100%) when we replace our training data with new one and restrict it to be over minimal pairs as illustrated in the last experiment which have been used in our CALL. Table 4.4 summarises the experiments described in this chapter.

Table 4.4: Experiments’ summary

<table>
<thead>
<tr>
<th>Experiment Type</th>
<th>Factor</th>
<th>Recognition Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech Units</td>
<td>Word Level</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>Phoneme Level</td>
<td>39</td>
</tr>
<tr>
<td>Gender</td>
<td>Male</td>
<td>40.2</td>
</tr>
<tr>
<td></td>
<td>Female</td>
<td>45</td>
</tr>
<tr>
<td>Better Training Data</td>
<td>anyword grammar</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>constrained grammar</td>
<td>100</td>
</tr>
</tbody>
</table>

4.3 Conclusion

This chapter completed the research task RT1 “To ensure that the ASR system we are using can be made accurate enough to support the principle aim” which
is the first research task mentioned in Chapter 1 (i.e., Section 1.3). Now, we will move to Chapter 5 which will deal with the second research task regarding the development of the animation tool (i.e., one version of our CALL tool).
Chapter 5

Animating the Head

This chapter shows in detail the first source of feedback to learners trying to pronounce Arabic words correctly. This feedback is an animated head to show learners the difference between their articulation and the native articulation. Therefore, this chapter is aimed at giving an account of how non-native Arabic speakers can be supported in making the different sounds that make up Arabic (i.e., help them to sound more like native speakers) by showing them an animation of the vocal tract corresponding to the sounds they have made. By developing this feedback method in our CALL tool, the learner will be given a graphical representation of both the way that they have pronounced the sounds, and the way that the sound is produced correctly. As mentioned in the previous chapter, this research uses a speech recogniser (i.e., the HTK) to identify the properties of speech signals for both native and non-native speakers. We trained the speech recogniser using native speaker utterances to recognise phonemes of the input speech in order to obtain a phonetic analysis, which is used to animate the vocal tract. In this way, we can provide a visual representation of the acoustic data which we hope will enable learners to adjust their pronunciation. Figure 5.1 shows the articulators which move around when somebody says something.

5.1 Animation Version of our CALL Tool

This version of our CALL tool is illustrated in Figure 5.2, which is an example of an Arabic speaker learning English. In this figure, we assume that our target word is “PRAY”. When our learner says “PRAY”, our tool will match what he
5.1. ANIMATION VERSION OF OUR CALL TOOL

Figure 5.1: Movable articulators

said to the nearest native speaker sounds. The figure shows that the output of the HTK is “BRAY” not ‘PRAY”. This means that our learner pronounced the target word (i.e., “PRAY”) incorrectly. Therefore, our tool will provide the learner with an animation of what he said (i.e., “BRAY”) and what he should have said (i.e., “PRAY”). In other words, the learner will be given two feedbacks in the case of incorrect pronunciation.

This is what might happen for an Arabic speaker learning English because some people with this linguistic background have difficulty with “P” and “B” sounds. These two sounds are confusable and have similar features which make them hard to produce for non-native speakers. In this case, if the Arabic learner tried to say “PRAY”, it is more likely for him to say “BRAY” instead of “PRAY”. This example is a concrete example of the type of mistake that we are trying to identify from non-native speakers learning Arabic.

To achieve this animated version of our CALL tool, we completed the following two main steps:

- we carried out morphing from one drawing to another

- we integrated the running of the HTK with the animation

Now, we will talk about these two steps in more detail.
In order to do the animation, we could construct an abstract model of the head as a 3D model, providing models of the muscles such as the tongue and chewing muscles, and making the head move around in response to contracting and stretching the muscles, as is done with Baldi [Mas04]. However, we did not go down the route of making a 3D model of the head to include the muscles of the inside articulators. Alternatively, we chose to construct a simpler presentation of articulator movements by taking a set of line drawings of where the articulators are for various positions and morphing between them. This simple approach is quite approximate (i.e., not an anatomically rigorous correct description), but we believe that it will work as a guide to what learners should be doing with their
articulators because they have never seen the inside of their own mouth. Therefore, if we show learners a picture of, for example, a tongue being put against the back of a set of teeth, they will try to internalise that in order to produce the correct sound which gives the impression needed.

First of all, we drew a set of lines to constitute the drawing of the vocal tract which is the first step in order to obtain our animation tool. In this step, we used a tool called "VisionKit" which helps in tracing a set of pictures (i.e., line drawings) by ‘mouse click’ and acquiring their coordinates. For example, Figure 5.3 presents source pictures which we traced from by using the VisionKit, and Figure 5.4 presents the traced copies. The traced versions have many missing points, which were filled in by interpolation. Despite the use of interpolation, the traced versions look like the originals.

The motion is the key issue if learners want to know how their articulators work. Therefore, if learners just put, for instance, their tongue in the required position without motion, this leads to no sound, meaning that the transition from one position to another is the key factor here. For example, in Figure 5.3, if a learner wants to say /t/ (i.e., the top left picture), the picture we draw is the point in the middle of /t/ when the tongue is behind the teeth, but what
makes the /t/ sound is moving the tongue from somewhere else to that position and then taking it away.

Therefore, the next crucial step is that we perform the morphing from one drawing to another in which we assign phonetic units to individual snapshots (i.e., merging two images, point by point, over a period of time). This part of the project is quite significant. In more detail, we generated a set of morphs from a set of drawings using a hash table written in Java as shown in Appendix D.

This table (containing all the geometries that make up the animation we are trying to make) is generated using the DTW (Dynamic Time Warping) algorithm which makes the alignment of coordinates (see Appendix C). This alignment must be done because the morphing cannot be carried out unless all morphed pictures have the same length sequences (i.e., the same number of points). So, to make use of a nice smooth morphing, we have to have pairs of images with the same numbers of points. For example, the drawing of the phoneme ‘b’ (Figure 5.5) can be morphed to the drawing of the phoneme ‘i’ (Figure 5.6) which can then be morphed to the drawing of the phoneme ‘r’ (Figure 5.7) to give the animation of the Arabic pronunciation ‘b i r’.

![Figure 5.5: Drawing of the phoneme ‘b’](image)

This morphing could be done in a way that requires us to take very fine control over the timing of update on the screen. However, we did the morphing by using Java3D where its settings do the job more smoothly. Java3D requires us to create a certain type of object. In Java3D, we used a class called Morph which is related to Shape3D. This class allows us to combine several GeometryArray objects in order to give a smooth transition from one shape to another. GeometryArray class helps us to draw primitives such as points and lines that are defined using vertices. For example, a Morph object may be used to show a hand (i.e., containing...
5.1. ANIMATION VERSION OF OUR CALL TOOL

Figure 5.6: Drawing of the phoneme ‘i’

Figure 5.7: Drawing of the phoneme ‘r’

many GeometryArray primitives) that can be opened and closed in a smooth motion [WG01]. Furthermore, we used the TransformGroup class and its closely associated Transform3D class which form the basis of all spatial manipulation [Bar00]. These associated classes helped us to change the transformation of the target object, giving it a different position, orientation, or size. In terms of performing animations over time, we also used a class called Alpha which has many components with key values that can be changed to control the speed of the morphing operation and obtain the most suitable timing for the animation [BP99]. These main classes of morphing work together as presented in Appendix D in order to animate the vocal tract as a feedback to Arabic learners.

Morphing produces smooth animations of line drawings, and turns out to be useful for learners as a form of feedback. The experiments reported in Chapter 7 show that learners benefit from using this kind of feedback.
5.1.2 The Integration

This step is the last step for producing our animation tool. In this step, we carried out a substantial piece of implementation which is done by integrating two existing pieces of software. The first piece of software is for identifying mispronunciation by driving the speech recogniser (i.e., the HTK) in which we use phonemes as terminal symbols instead of words. The second piece of software is for animating a sequence of images of articulatory positions.

The integration can be described in the following scenario: in the interface of our CALL tool, when a learner records his speech, he will press the ‘Record’ button in the environment of Java (SpeechTeach.java). Then, after finishing the recording, he will press ‘Stop’ leading to running the HTK in order to assess his speech, and this running of the HTK happens in the environment of our Python program (scripts.py). Therefore, we are trying to build a conversation between Java (i.e., SpeechTeach.java) and Python (i.e., scripts.py) and the HTK. Java performs the recording in the user interface of our CALL tool. When a learner finishes his recording and presses the ‘Stop’ button, RunTheHTK() class is called in order to have a dialogue with Python for running the HTK and assessing the learner’s pronunciation. Python responds to Java and runs the HTK to carry out the training. The code that implements the integration between Java, Python, and the HTK is presented in Appendix E.

Consequently, the animation version of our CALL tool is made from the integration of running the HTK from Python and using its output to animate the head from Java. The output of the HTK is the recognised phonemes (written in a text file) which the HTK thinks that a learner says when he uses the tool. These phonemes coming out from the HTK were, after that, assigned to certain images and these images were morphed to give an animation source of feedback to learners. Running the HTK through Python is an underlying process which a learner is not aware of, but it is an essential part in this version of our tool.

After we have done the integration, the learner can speak to our CALL tool and then the movable articulators of both his speech and the correct speech are displayed to him. For example, Figures 5.8 and 5.9 show two animated feedbacks to a learner trying to pronounce the word /tʰi:n/. The learner pronounced this target word incorrectly as he said /tʰi:n/ instead of /tʰi:n/. Therefore, Figure 5.8 shows him an animation of what he said (i.e., /tʰi:n/), and Figure 5.9 shows him
another animation of what he should have said (i.e., طين). The core difference between the two animations is located in the first phone which the learner finds difficult to produce. The two words طين and تين are identical apart from one letter (i.e., minimal pair) which is located at the beginning in this example, and this letter (ط) is a confusable phoneme with its counterpart (ت). The learner can look at this difference and attempt to adjust his pronunciation according to the provided feedback.

Figure 5.8: Animated feedback on learner’s pronunciation

Figure 5.9: Animated feedback on correct pronunciation

5.2 Conclusion

In this chapter, we achieved the completion of the research task RT2 “To design and implement an animation of the vocal tract (i.e., the animation tool)” which is the second research task mentioned in Chapter 1 (i.e., Section 1.3). In Chapter 6, we will deal with the third research task regarding the development of the synthesis tool (i.e., one version of our CALL tool).
Chapter 6

Speech Synthesis in CALL Tool

This chapter gives an explanation of the synthesis version of our CALL tool as a form of feedback to Arabic learners. It shows the tools that were integrated to produce a clear voice as a feedback from our tool. It highlights the diphone-based synthesiser called MBROLA which is used for the synthesis, and also points out some experiments for improving the naturalness of the speech.

6.1 Generating Synthetic Speech

Since we have the recognised phonemes as an output from the HTK (Chapter 4), we can extract the phoneme sequences and thus we have the ability to give learners feedback about their pronunciation. We used synthetic speech as one source of feedback to learners. To generate synthetic speech from a set of phonetic transcriptions, we used the MBROLA speech synthesis tool based on the concatenation of elementary speech units called diphones [DPP+96]. This means we had to supply a diphone set for Arabic, which is difficult to make, but fortunately an Arabic diphone set is supplied for MBROLA and therefore we can use it. We supplied MBROLA with the phonemes of each word used in the training samples, and also supplied it with other information such as phone length and pitch value.

When we write an MBROLA script, we write the following: phones, their lengths, and their pitch values, as shown in Figure 6.1. Phone lengths or durations are specified in milliseconds (thousandths of a second). Hence, the second line in Figure 6.1 tells MBROLA to synthesise the ‘j’ phoneme and make it last for 204 ms (Note: the first line tells MBROLA to have a silence at the beginning to last
for 100 ms, and the last line as well).

![Figure 6.1: MBROLA script](image)

MBROLA is easy to use as we just specify a set of phones, durations, and pitches. However, if we do not choose good values for these parameters, it is likely to produce robotic, unnatural sounding speech, which is particularly unfortunate for our task. We cannot afford to have strange sounding speech because this speech is what our learners are supposed to learn from. Therefore, we need to optimise the specifications of the MBROLA script in terms of the appropriate values of its entries.

We have the advantage that the Arabic diphones in the default database within MBROLA were recorded by an Arabic native speaker which means the acoustic properties of each individual phoneme are guaranteed to be right. This leads to the conclusion, at some level, that we can make the sound perfect. So, the generated sound can be absolutely perfect if the specifications of the MBROLA script (i.e., phone lengths and pitches) are controlled to be very close to the Arabic real speech. The following Sections 6.2 and 6.3 show our work to improve these specifications of the MBROLA script.

6.2 Estimating Phone Length

MBROLA converts the script of word phonemes into speech. However, this speech sounds robotic and flat if we have a fixed length for each phone and a fixed pitch. Therefore, we would like to have better lengths for phones and better pitch variations.

One issue regarding estimating phone length is that MBROLA finds an adjacent pair of phones (i.e., diphones) and makes an invisible ‘black box’ use of
the durations which we give for the individual phones in order to compress them into diphones. So, producing diphones could lead to durations which are different from what we assign as individual phones. We certainly do not know what MBROLA does with the durations here, and how it converts phone lengths to diphone lengths is not specified in any documentation. We can, however, attempt to compensate for places where a particular diphone is going to be produced for a too long or too short duration by varying the amount of time (i.e., phone length) of one of its constituents.

The alignment process with the real Arabic speech is used in order to get the phone length right. Firstly, we used the Praat tool [BW11]. Praat helps to find a list of intensity contours for a wav file of the following: real speech (i.e., recorded speech), synthesised speech (i.e., generated speech), and realigned synthesised speech. Then, the Dynamic Time Warping (DTW) algorithm is used to align both intensity lists, and thus the lengths of the phones were recalculated and realigned iteratively until the lengths converged. This alignment is used to estimate the ideal phone length which leads to improving the quality of the synthesis.

Figure 6.2 is extracted from Praat and shows an intensity contour of a real sound of a native Arabic speaker as a function of time drawn as a green line. This real speech is aligned with the intensity contour of a synthesised sound which is generated from MBROLA (Figure 6.3). Finally, the algorithm of DTW is used to do the alignment between the previous two intensity contours. Figure 6.4 shows the intensity contour for the sound after alignment which shows what happens to the intensity contour of the synthesised sound if we apply the shifts that were implied by the DTW.
6.2. ESTIMATING PHONE LENGTH

Figure 6.2: Intensity curve for a real speech

Figure 6.3: Intensity curve for a synthesised speech
Figure 6.4: Intensity curve for a realigned synthesised speech

The alignment process is carried out as an iterative process. First of all, we started with a flat set of phone lengths, and then we synthesised these sequences of phones. After that, we aligned the synthesised speech with the real one. Then, we resynthesised the speech followed by the alignment again and again until a certain point which is specified by 5 iterations. Figure 6.5 shows the phone length of the first phone in a flat sound (in sec) which is indicated above the selected portion of the signal (pink rectangle), and Figure 6.6 shows the phone length of the same phone after the alignment.
6.2. **ESTIMATING PHONE LENGTH**

![Flat Sound Extracted from Praat](image1)

**Figure 6.5:** Phone length in a flat sound extracted from Praat

![Improved Sound Extracted from Praat](image2)

**Figure 6.6:** Phone length in an improved sound extracted from Praat
We performed a contextual analysis of the length of phones in order to obtain more accurate estimation of the length. To do that, we stored all phone lengths in three tables. The first table contained phone lengths without any conditions, and this table was called ‘unconditional’ table. The second table contained phone lengths when the phone is followed by a specific phone, and this table was called ‘suffix table’. The third table contained phone lengths when the phone is preceded by a specific phone, and this table was called ‘prefix table’. So, the first table is an unconditional table, while the other two tables are conditional ones. After that, we normalised all the three tables by taking the average length for each phone. For example, in our code suffix_Table = {'S': {'b': [155]}, ...} means that the phone ‘S’ has a length of 155 if it is followed by the phone ‘b’. Similarly, prefix_Table = {'S': {'a': [160]}, ...} means that the phone ‘S’ has a length of 160 if it is preceded by the phone ‘a’. In the unconditional table, the code Unconditional_Table = {'S': [165], ...} says that the phone ‘S’ has a length of 165 without any conditions.

Since we are attempting to produce speech that an Arabic learner can use as a model and try to copy, some tests on the output quality of the speech were carried out. A set of 10 Arabic words is synthesised in three different versions of how we assign the length values. The first version was performed by setting the phone length to the average value of the following: the value in the suffix table for this phone with its current suffix, the value in the prefix table for this phone with its current prefix, and the value of its length in the unconditional table. This version was called Version A. The second version was done by setting all phone lengths to a fixed value, which is equal to the average length of all phones in the unconditional table, and this version was named Version B. The last version was made by assigning a phone length to the average value of the phone’s length in the suffix table with its current suffix and its length in the prefix table with its current prefix, and this version was called Version C. These three versions of the ten Arabic words were played to two Arabic native speakers in order to select which version is the best in terms of clarity and naturalness. The results show that Version A was selected 5 times, Version B was selected 4 times, and Version C was selected 11 times. This indicated that Version C was the best one to be used in our experiment.
6.3 Improving Pitch Contour

Pitch is debatably the most expressive feature of speaker-dependent prosody, which also includes other factors like phone length, loudness and pause locations \[\text{[HAH01]}\]. Different speakers have different pitch ranges; for example, women are more likely to have higher pitches than men.

A pitch contour is related to the highness or lowness of a sound over time. In general, speech has both a global pitch contour and a local pitch contour. The global pitch contour expresses emotions and linguistic features such as the sentence type. For example, most declarative sentences have a decreasing pitch contour. In contrast, most questions have an overall increasing pitch contour. The global pitch contour is actually meaningless for our task because our CALL tool is designed to be an isolated words system, and the isolated words do not have a global pitch contour.

The local pitch contour is driven by stressed syllables and it is extremely important for our task. The stress can be identified from the standpoints of production and perception. From a production standpoint, stressed syllables are produced with more muscular energy than unstressed syllables. From a perceptual standpoint, all stressed syllables have one feature in common which is prominence. There are four different factors which make the syllable prominent: pitch, length, loudness and vowel quality \[\text{[Als14]}\].

Many researchers suggest that stress is an essential feature for any language. For example, Lea et al. \[\text{[Lea80]}\] declares that “stressed syllables can form the anchors around which a search for occurrences of words can be attempted”. In the field of NLP, researchers have addressed the importance of locating stressed syllables and have used this information as a strong feature to enhance the acoustic signal of speech.

It is fairly easy to identify the stressed syllable of a word in most languages. However, in Arabic this identification is not as easy due to the existence of semi-vowels. This means an effort should be made to determine carefully whether a phoneme is a consonant or a vowel before the attempt of finding syllables and where the stress is. For instance, the Arabic letter \(\text{\textdegree}\) (Waw) is sometimes treated
as a consonant (e.g. /wa/) and in other times, it is treated as a vowel when it is preceded by a Damma (i.e., one of the Arabic diacritical marks; a short vowel) [AEAG+08]. This kind of effort was made by Alsharhan [Als14] and Ramsay et al. [RAA14], culminating in defining the necessary stress rules for Arabic.

The allowed syllables in Arabic are: CV, CVC, and CVCC where V represents a short vowel, C represents a consonant [Alo07], and VV represents elongation (i.e., long vowels). Arabic words can only start with a consonant (C). Ramsay et al. [RAA14] defined the Arabic stress rules which can be described briefly as follows:

1. If the word contains a super heavy syllable (CVVC or CVCC), this syllable must be stressed. For example, consider the word علام ‘knowledgable’ /'a-laam/ (i.e., contains CVVC), and the word كتبت ‘I wrote’ /ka-tabt/ (i.e., contains CVCC). This type of syllable can only be found once in a word.

2. If there is no super heavy syllable, the stress is placed on the last open heavy syllable (CVV) if there is one. For instance, consider the word كتبت ‘writer’ /ka-tib/ and the word كتبتون ‘writers’ /ka-ti-bu-na/.

3. In the case where the word has neither a super heavy syllable nor an open heavy syllable, the place of the stressed syllable depends on the number of syllables in the word:
   - In disyllabic words, the stress falls on the first syllable (e.g., منكبت ‘a ride’ /mar-kab/).
   - In multisyllabic words, the stress is applied on the antepenultimate syllable (i.e., two before the last; on the third from last syllable). For example, كتبت ‘she wrote’ /ka-ta-bat/.

We used the rules above to make use of the observation that stressed syllables tend to have slightly higher pitch than their neighbours. The stress rules are

\[\text{The VV notation means elongation, not two single vowels.}\]
6.4. CONCLUSION

built in the locating stress program (developed by Alsharhan \textsuperscript{Als14}) which we used to help find the stress of a given word and add the right pitch values. The assigning of local pitch contours according to the stress program leads to better pitch variations and improves the quality of the synthesis.

Just as with the tests on phone length in the previous section for the output quality of the speech, we have carried out tests on the pitch contour. The target speech was synthesised in four different versions of how we assign the pitch values. The first version was done by setting pitch values of the stress syllables to be high (i.e., 150) and pitch values of the unstressed to be low (i.e., 120), and this version was called Version A. The second version was a slight modification to Version A where pitch values of stress syllables were changed to be 180, and of unstressed ones were changed to be 110, and this version was called Version B. The third version was performed by setting the vowel in the middle with a high pitch value (i.e., 150) and the extreme ends with low pitch values (i.e., 120). After that, the phones in-between were provided with interpolated intermediate values, and this version was called Version C. The last version was called Version D which is similar to Version C but with higher pitch values. Ten Arabic words were synthesised using these versions. After that, they were played in random order to two Arabic native speakers, and who were asked to listen and determine which was the best version in terms of realistic and natural speech. The results showed that Version A was chosen 6 times to be the best, Version B was chosen only once, Version C was chosen 8 times, and Version D was chosen 5 times. This meant that Version C was the best one, and hence we used it in our experiment.

The above tests in both this section and the previous one (Section 6.2) helped us to obtain the right values of both phone length and pitch. These values were added to the MBROLA script leading the generated speech from MBROLA to be more realistic, which would help Arabic learners to listen to the correct pronunciation clearly.

6.4 Conclusion

In this chapter, we attained the completion of the research task RT3 “To design and implement a tool for synthesising what the speaker’s utterance sounds like (i.e., the synthesis tool)” which is the third research task mentioned in Chapter 1 (i.e., Section 1.3). At this point, our CALL tool has been developed completely
with all of its versions. We then applied the tool in an Arabic class in order to evaluate its effectiveness (i.e., the fourth research task). This evaluation will be presented in the next chapter (Chapter 7).
Chapter 7
Classroom Experiments and Evaluation

This chapter presents the classroom experiments which were carried out with Arabic learners in order to assess the effectiveness of our CALL tool. This assessment is actually assessing the effectiveness of its use and how it works in order to help students learn rather than assessing the CALL itself as a piece of software [Zha03]. In other words, the evaluation is based on measuring human performance using our CALL tool rather than measuring system performance [Kin05]. Moreover, this chapter shows the results of these experiments with regard to the improvement of learners’ pronunciation. The usability and learner behaviour using five different versions of the CALL tool are also discussed in this chapter.

7.1 Experimental Setup

In general, CALL tools are very difficult to evaluate because learning a language is a long, slow process and measuring the effect of a tool requires a large scale longitudinal study. However, experiments with Baldi [Mas04] suggest that people’s pronunciation can in fact improve quickly if they are given a tool like this. Therefore, we talked to people in some schools where they teach Arabic, and let them use our tool with each student for a period of half an hour. We took all the students in the Arabic class who were available to us and willing to do the experiments. The students were presented with laptop computer, microphone, and headphones. The students were provided with a number of pronunciation tasks.
CHAPTER 7. CLASSROOM EXPERIMENTS AND EVALUATION

After producing their pronunciation, they had a number of choices: they could repeat their utterance, go to the next utterance, access some kind of feedback, and after each action they could still access the full range of these choices. So, they made as many attempts as they want to make in the 30 minutes. Moreover, we made some hypotheses about sounds which we know that people who are learning Arabic have difficulty with. These are the sounds that non-native speakers find difficult to produce which we can diagnose and give some information about.

We had 50 Arabic students in total who used five different versions of our CALL tool. These versions were: the full version (i.e., includes all facilities), the animation version, the synthesis version, the instruction version, and the language lab version (more detail in Section 7.3.1 for the language lab version). We divided the students equally into five groups, which means that each 10 students would use one of the previous five versions. Each student had a thirty minute session with the tool, working their way through a set of pronunciation exercises at their own pace.

7.2 Results

Our measure of performance is what percentage of Arabic students in each sample performed well. The measured samples were consecutive Arabic words taken at the beginning and consecutive ones taken at the end. The window size taken at both sides (i.e., beginning and end) was chosen to be not too small to avoid using insufficient samples as we needed a sensible measure, and at the same time, not too big because this would lead to overlapping between both beginning and end samples.

The performance of students was computed at the beginning and at the end of the session as shown in Table 7.1. We measured learners’ performance on some Arabic pronunciation tasks, and how often they carried them out right. So, the numbers in the table represent how accurate learners were at both the beginning and the end of the session and the average improvement for these learners. The results show that the pronunciation of the students improved when they used any of the different versions of our tool apart from the instruction version. Those students who used the synthesis version had an improvement in their pronunciation of about 27%, and 17% improvement was recorded for those who used the full version, and 18% improvement for those who used the animation version.
version. These are the highest percentages of pronunciation improvement which provide an indication that these versions (i.e., synthesis, full, and animation) are the most useful ones for learners.

A low percentage of improvement at 6% was documented with the language lab version. This version gives all facilities to students, but without providing any indication of whether their pronunciation is correct. This means that they simply practise saying words, with the option of listening to their own recordings and idealised recordings and of looking at animations but with no information about whether their own pronunciation was correct. This version mirrors the facilities of a standard language lab, with the extra possibility of looking at animations as well as listening to recordings. This version is included in order to measure the practice effect so that when the improvements produced by other versions have been measured, we can have a benchmark against which to measure them.

The results, which are presented across all students, suggest that the different versions of our CALL tool did produce some beneficial effect beyond that achieved through practice with the language lab set-up.

<table>
<thead>
<tr>
<th>Version</th>
<th>At the beginning</th>
<th>At the end</th>
<th>% Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>0.39</td>
<td>0.56</td>
<td>17%</td>
</tr>
<tr>
<td>Animation</td>
<td>0.44</td>
<td>0.62</td>
<td>18%</td>
</tr>
<tr>
<td>Synthesis</td>
<td>0.35</td>
<td>0.62</td>
<td>27%</td>
</tr>
<tr>
<td>Instruction</td>
<td>0.44</td>
<td>0.44</td>
<td>0%</td>
</tr>
<tr>
<td>Language Lab</td>
<td>0.56</td>
<td>0.62</td>
<td>6%</td>
</tr>
</tbody>
</table>

Table 7.1: Classroom results

In order to see whether this result is statistically significant, we broke down the data into individual numbers. The following table (Table 7.2) contains more details which show the performance for each student with each version. In this table, we have 5 different versions (animation, synthesis, instruction, full, and language lab) and our 10 observations of how much each student improved with each version. These observed numbers represent a ratio between the number of mistakes made at the beginning and the number of mistakes made at the end for individual students.

The statistical model called ANOVA (Analysis of Variance) was used with these results. So, the p-value was computed to measure how likely the data for each version were to have occurred by chance, assuming the null hypothesis is true. The null hypothesis for our test is that the tool provides no more
CHAPTER 7. CLASSROOM EXPERIMENTS AND EVALUATION

Table 7.2: Individual student improvement

<table>
<thead>
<tr>
<th>Animation</th>
<th>Synthesis</th>
<th>Instruction</th>
<th>Full</th>
<th>Language Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>1.5</td>
<td>2</td>
<td>1.5</td>
<td>1.67</td>
<td>1.67</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.3</td>
<td>1.67</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1.3</td>
<td>1.67</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>3</td>
<td>0.67</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1.25</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>1.25</td>
<td>1.5</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>0.67</td>
<td>1.5</td>
<td>0.67</td>
<td>1.67</td>
<td>1</td>
</tr>
<tr>
<td>0.5</td>
<td>4</td>
<td>0.80</td>
<td>3</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 7.2 shows that when we compare the results for students who use the full version of the tool to those for students who just used the language lab version (i.e., control group), we obtained a p-value of 0.046 which means there is a probability of about 1 in 20 of this result occurring by chance. Similarly, when we attempted to look at the other versions of the tool, we found that the p-values are 0.09 (i.e., about 1 in 10 chance occurrence) for the animation, 0.008 (i.e., about 1 in 100 chance occurrence) for the synthesis version, 0.87 (i.e., high value meaning no improvement recorded) for the instruction version. This means that the only statistically significant figures are for the full and the synthesis versions, assuming a significance level of 0.05.

benefits than an equivalent session in a standard language lab which says ‘nothing’s happening’ [Cra05]. In other words, a learner’s performance does not improve more than what we would expect if he simply practised saying Arabic words for half an hour with the option of listening to his own recorded voice and idealised recordings and of looking at animations but with no feedback about whether his own pronunciation was correct or not. Table 7.3 shows the p-value for each version.

Table 7.3 shows the p-value for each version vs. language lab

<table>
<thead>
<tr>
<th>Version Vs. Language Lab</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>0.046</td>
</tr>
<tr>
<td>Animation</td>
<td>0.09</td>
</tr>
<tr>
<td>Synthesis</td>
<td>0.008</td>
</tr>
<tr>
<td>Instruction</td>
<td>0.87</td>
</tr>
</tbody>
</table>

Table 7.3 shows that when we compare the results for students who use the full version of the tool to those for students who just used the language lab version (i.e., control group), we obtained a p-value of 0.046 which means there is a probability of about 1 in 20 of this result occurring by chance. Similarly, when we attempted to look at the other versions of the tool, we found that the p-values are 0.09 (i.e., about 1 in 10 chance occurrence) for the animation, 0.008 (i.e., about 1 in 100 chance occurrence) for the synthesis version, 0.87 (i.e., high value meaning no improvement recorded) for the instruction version. This means that the only statistically significant figures are for the full and the synthesis versions, assuming a significance level of 0.05.
We were only able to have a sample of 50 students, 10 in each group who took a session for half an hour each, which means our individual groups were quite small. Therefore, it is really hard to perform very reliable statistics. The ANOVA analysis suggests that the effects produced with the synthesis and full versions were fairly convincing.

7.3 User Behaviour

In this section, we attempt to gain an understanding of the behaviours of Arabic learners using our CALL tool. In particular, we wish to correlate the user behaviours to our results by looking at the state transition diagram of their behaviours. We drew this diagram to represent each step done by the learner during a half-hour session.

The learners used one version out of five, and the state transition diagram will differ from one version to another because the options available to the user differ from one version to another. These versions are: the full version, animation version, synthesis version, instruction version, and language lab version with 9 nodes, 8 nodes, 7 nodes, 6 nodes, and 9 nodes, respectively.

In the discussion below, we discuss various types of behaviours that were observed, and show individual instances of those behaviours. Several of these behaviours are examples of more general patterns of behaviour. Appendix K shows all the behaviour traces grouped into these more general patterns.

7.3.1 Using the Language Lab Version

This version of the tool is designed to estimate the practice effect so that we can eliminate this effect as a contribution to the other versions. When a learner sits for half an hour pronouncing Arabic words and listening to himself, this will lead to an improvement in his pronunciation due to the practice effect. With this version, we are attempting to construct a baseline in order to determine whether learners improve more when they use the other versions of the toolkit than they would have performed with half an hour of practice.

This version is shown first because it provides the baseline of our tool. In this version, we replicated a traditional language lab where a learner says prompt words, listens to his own voice, listens to the instructor’s voice, says the words
again, listens to his own voice again, listens to the instructor’s voice again, and then he moves on.

The results show that there was just 6% improvement of students’ pronunciation as a result of using the language lab version. This version does not provide the student with a feedback of result for his pronunciation whether it is correct or incorrect while the other versions do. This result could be the reason for the low percentage of improvement seen when using the language lab version.

As we said above, the students who used this version did not know whether they pronounced the sample correctly or not. Therefore, they behaved according to what they thought was the best way of improving their pronunciation. Most students tended to listen to their voice several times as shown by the loop from ‘Play Me’ node to the ‘Play Me’ node and then compare it to the correct synthesised voice as shown by the thickness of the arrow going from ‘Play Me’ node to the ‘Synthesise Correct’ node in Figure 7.1. This behaviour was a very common pattern for students using the language lab version. Other examples of this behaviour are shown in Section K.1 of Appendix K (Figures K.1 - K.7).

![Behavior 1 in the Language Lab version](image)

Figure 7.1: Behaviour1 in the Language Lab version (Figure K.1 in Appendix K)

The sole exception is the student whose behaviour is shown in Figure 7.2, where the student preferred to display the explanatory text of how to pronounce the sample correctly (‘Display Instruction’ button). After that, he looked at the animation of articulators for pronouncing this sample (‘Display Correct’ button). This behaviour is reasonable as the student can link what he gained from the text to the animated head.
7.3. USER BEHAVIOUR

7.3.2 Using the Animation Version

The results show that learners’ pronunciation improved by 18% after using the animation version. Figure 7.3 shows an example of the behaviour of a group of students who used the animation version. These students have pronounced most samples correctly, which is shown by the thickness of the arrow going from the ‘Record’ node to the ‘Correct’ node. The figure also shows that the user used the ‘Display Correct’ button of our tool to correct his mispronunciation. In more detail, when the learner pronounced the Arabic word incorrectly, he tended to see what he did first (i.e., ‘Display Me’ button), and then he displayed frequently what he should have done (i.e., ‘Display Correct’ button), and finally recorded it again trying to pronounce it correctly. Other examples of this behaviour are shown in Section K.2 of Appendix K (Figures K.9 - K.11).

A second group kept trying the same sample in case of mispronunciation until they pronounced it correctly. Figure 7.4 is an example of this behaviour. Other examples of this behaviour are shown in Section K.2 of Appendix K (Figures K.12 - K.16).

7.3.3 Using the Synthesis Version

The results show that there was an improvement of about 27% in pronunciation for those who used the synthesis version. Most learners who used this version had a common behaviour. They tended to focus on listening to the correct synthesised voice and then record their voices again attempting to pronounce
the words correctly. They also tended to play their recording several times, and sometimes they tended to listen to both their synthesised voice and the correct synthesised one in order to see the difference if any. Figure 7.5 shows an example of such behaviour. All other examples of this behaviour are shown in Section K.3 of Appendix K (Figures K.17 - K.20).

Figure 7.6 gives an example of different behaviour of another group of students. When the learner mispronounced the sample word, he struggled many times until he pronounced it correctly. Moreover, when he finished practising all samples, he went back to the beginning to practise again and make sure of his rectified pronunciation as shown by the thickness of the ‘Previous’ node. Other examples of this behaviour are shown in Section K.3 of Appendix K (Figures K.21 - K.24).
7.3. USER BEHAVIOUR

There was no improvement recorded using the instruction version as mentioned in Table 7.1. Most learners of this version behaved as follows: they recorded an Arabic word, then if they were told they had pronounced it correctly, they went on to the next word. If they had pronounced it wrongly, they mainly re-recorded it. An example of this mode of behaviour is illustrated in Figure 7.7 which shows that learners did not like reading the textual information. Other examples of behaviours are shown in Section K.4 Appendix K (Figures K.25 - K.32).
7.3.5 Using the Full Version

In this version, learners have all the facilities available in other versions to exploit in order to improve their pronunciation in the designated period of half an hour. The results showed an improvement of 17% for those who used this version.

Different behaviours of learners were recorded with this version. For instance, one learner attempted to rectify his mispronunciation by using both animation and instruction facilities. Moreover, he ascertained his correct pronunciation by using synthesis and other facilities. Such behaviour can be seen in Figure 7.8.

Another learner chose to correct his pronunciation only via the animation facility. He preferred to confirm his correct pronunciation by using both the correct
animation and synthesised voice. He benefited from the instruction facility only when he listened to the correct synthesised voice and this was followed by displaying the correct animation. Figure 7.9 shows the behaviour of this learner. This version has different behaviours due to the different types of provided facilities. Other examples of behaviours are shown in Appendix K (Section K.5).

![Figure 7.9: Behaviour2 in the full version (Figure K.34 in Appendix K)](image)

### 7.4 Conclusion

In this chapter, we have finished the research task RT4 “To carry out classroom evaluation of the tool” which is the last research task mentioned in Chapter 1 (i.e., Section 1.3). The results showed that there is a general improvement in the pronunciation of learners who use such a tool (RQ1). It was, however, not possible to reliably assess which of the tools was most effective (RQ2).
Chapter 8

Conclusion and Future Work

8.1 Research Questions and Research Tasks Revisited

In this thesis, we sought to investigate an effective way of helping non-native Arabic speakers with their pronunciation of Arabic. The work aimed at teaching non-native Arabic speakers how to sound like native speakers by providing them with a CALL tool to be a diagnostic tool and pronunciation support containing multiple types of feedback.

In the introduction of this thesis, we raised and set out to address two research questions:

RQ1 Does such a tool contribute to improving Arabic learners’ pronunciation?

RQ2 If so, which form of feedback is the most effective for helping Arabic learners with their pronunciation?

The classroom experiments in Chapter 7 showed conclusively that the performance of students who used different forms of our CALL tool (i.e., synthesis, animation, full) versus the students who used the language lab version was significantly better. Therefore, the answer to RQ1 “Does using our CALL tool improve Arabic learner’s pronunciation?” is yes. Unfortunately, we could not answer RQ2 in sufficient detail “which formal feedback is the most effective for helping Arabic learners with their pronunciation?” because we do not have enough data to answer this question reliably. Despite having some suggestive answers from the classroom results, we would prefer to have more data. We believe that if we had
more students, we might have been able to see more evidence in the pattern of usage.

In order to answer these questions (i.e., RQ1 and RQ2), we had to carry out the following tasks:

**RT1** To ensure that the ASR system we are using can be made accurate enough to support the principal aim

**RT2** To design and implement an animation of the vocal tract (i.e., the animation tool)

**RT3** To design and implement a tool for synthesising what the speaker’s utterance sounds like (i.e., the synthesis tool)

**RT4** To carry out classroom evaluation of the tool

**RT1** The overall project required, first of all, a fundamental component which is having a high-performance speech recognition system. A large amount of work and effort was needed to resolve the many problems arising from different sources that hindered developing an accurate recognition system. The speech recogniser (i.e., the HTK) is used to identify the properties of acoustic speech signals for both native and non-native speakers. The HTK is trained by supplying it with native Arabic sounds as a standard in order to compare them with what a learner says. *Chapter 4* laid out the experimental dimensions of the developed systems by introducing the main component for the acoustic analysis (i.e., HMMs), describing how the recognition toolkit (the HTK) works, introducing the amendments made into the HTK processing stages to accommodate the developed phoneme level rules, and finally describing the collected datasets that were used in running the research experiments. These experiments were focused on the sounds which most non-native Arabic learners have difficulty with. We investigated many factors (speech units: phoneme level or word level, gender type, and better training data) in order to see their effectiveness on the recognition accuracy of the HTK. Finally, we used the model that resulted from the experiment for which we had the highest accuracy. We found that using the completely constraining grammar over minimal pairs led to have 100% recognition accuracy, which gave us the confidence we needed in order to use it for our task.
RT2  Our CALL tool includes three forms of feedback to Arabic learners about their pronunciation. The first form of feedback is an animation of the vocal tract, in which a learner is given a graphical representation of both the way that his sound is produced, and the way that the sound is produced correctly. We developed this source of feedback to learners (i.e., the animation tool) as described in Chapter 5.

RT3  We designed and implemented the second form of feedback (i.e., the synthesis tool), where a learner can play his sound, listen to a synthesis version of his articulation, and listen to a synthesis version of the correct articulation. Many experiments were conducted to improve the naturalness of the synthesised speech for this source of feedback as described in Chapter 6.

RT4  Finally, we assessed the effectiveness of our CALL tool. We applied the tool in an Arabic class and started conducting the classroom experiments which were performed with Arabic learners to investigate the improvement of their pronunciation. When we compared the results for learners who used the full or synthesis version of our CALL tools versus those for learners who just used the language lab version, there was a significant improvement. This classroom evaluation, the usability and learners’ behaviour related to using different versions of the CALL tool were discussed in Chapter 7.

8.2 Future Work

There are a number of possible directions for future research and also some remaining open questions related to the field that are worth further investigation. For example, the speech recognition system introduced in this research was created with the help of the HTK as a portable toolkit for building and manipulating HMMs. Our CALL tool would benefit from further work on speech recognition accuracy. As part of our effort to make sure that we have the most accurate recognition we could, it will be worth trying other available speech recognition toolkits such as CMU Sphinx because such tool may perform better on this kind of data and to ensure that our approach is robust even when applied in different recognition engines. CMU Sphinx toolkit is of particular interest to us due to the fact that it is open source, well-documented and well-maintained.
8.2. **FUTURE WORK**

Additionally, the tool would benefit considerably from further work on improving the speech synthesiser. Although participants studying Arabic seemed to like the speech synthesiser based on their traced behaviour, and although their pronunciation seemed to improve when they used the speech synthesiser, we are aware the speech synthesiser does not actually sound very natural, so improving it is worth doing.

In addition, in order to put this CALL tool in a practical classroom setting, especially in order to use the tool as certified summative assessment, further experiments need to be done. Moreover, it would be sensible to include many more words and vocabulary in the test sample, and to do directed lessons.

Improving the system would allow it to not only provide learners with guiding hints on how to improve their pronunciation but should also include the possibility of preparing exercises that are specifically tailored to their current needs. Also, in the error feedback, in addition to listening to the target word, the learner can also listen to the realisation of the target phone and the substituted one in that word.

One interesting future work would be applying our CALL tool to be a MALL (Mobile Assisted Learning Language) tool, or what is sometimes called MALU (Mobile Assisted Language Use). Jarvis and Achilleos [JA13] defined MALU as “non-native speakers using of a variety of mobile devices in order to access and/or communicate information on an anywhere/anytime basis and for a range of social and/or academic purposes in an L2”. Yamaguchi [Yam05] concluded: “a computer is better than a mobile phone for handling various types of information such as visual, sound, and textual information, but mobile phone is superior to a computer in portability. And some students don’t have their own computer”.

Recently, widespread ownership of mobile devices that can access wireless networks has led to the popularity of using these devices in order to support language learning. Therefore, learners are increasingly in a position to control activities motivated by their personal needs and circumstances [KHTP07]. MALL differs from CALL in its use of personal, portable devices that provide new ways of language learning, focusing on continuity of access and interaction [KHS08]. Mobile technology can promote learner independence, or the capacity to control one’s own learning. There has been a recent explosion in the availability of mobile apps for language learning. Mobile learning technologies such as iPhone, iPad, MP3/MP4 players, PDAs (Personal Digital Assistants), palmtop computers, and
others, are rapidly gaining popularity as an effective way to improve foreign language skills, such as the pronunciation skill. A reviewed study of mobile learning projects confirmed that mobile phones are the most frequently used devices in these projects [PK07]. The developers of Baldi, for instance, have moved into MALL technology as they developed an iPhone application called “iBaldiArabic” as shown in Figure 8.1. Using such apps in such devices (e.g., iPhones and iPads) can alter the dynamics of classrooms for the better: providing a wider range of learning activities; inspiring alternative forms of homework, assessment and feedback; and increasing chances for cooperation and creative expression [BHM+12].

Figure 8.1: MALL tool - iBaldiArabic
Bibliography


Project in Learning and Teaching, Heriot-Watt University, Edinburgh, UK, 2010.


135


Appendix A

Ethical Approval

Prior to commencing our experiments and collecting data or involving our human participants to use in our research, we must obtain ethical approval which is included in this appendix.
Dear Majed,

I am writing to you to confirm that your request for ethical approval for the study:

**Pronunciation Support for Arabic Learners**  
*Ref: CS111*

was considered by the School Ethics Committee, and judged to satisfy the requirements of our UREC-approved templates. It was hence granted ethical approval on the 29th August 2013 (passed by Simon Harper, CS Ethics Committee at that time).

This is also to confirm that CS111 is now recorded on the School of Computer Science Ethics Admin system as having been successfully completed (7th June 2014), with no alterations and no recorded problems.

Yours Sincerely

Dr Carole J Twining  
CS Ethics Committee 2014
Appendix B

Python Script for the HTK training

This appendix contains the python script which we wrote it to carry out the HTK commands needed for its training.

```
# this function (train htk) is to carry out the HTK commands needed for the training (10 steps)
def train_hawk(srcdir0,exptdir0, problems=[]):
    sys.stdout.write('STARTING/uni2423TRAINING:/uni2423srcdir0=%s,/uni2423exptdit0=%s

    srcdir = home + srcdir0 + '"
    exptdir = home + exptdir0 + '"

    # HTK commands for Data Preparation (Steps 1 - 5)
    # manually: gram file (your grammar)
    gramfile = srcdir+"gram"
    shutil.copy(gramfile,exptdir+"gram")
    test(['HParse',exptdir+"gram",exptdir+"wdnet"])
    create_global_file(exptdir)

    # manually: prompts file (audio file name + text transcription)
    if problems == []:
        shutil.copy(srcdir+"prompts",exptdir+"prompts")
    if not problems == []:
        prompts = open(exptdir+"prompts","r").readlines()
        prompts = remove_problems(prompts, problems)
        pfile = open(exptdir+"prompts","w")
        for p in prompts:
            pfile.write(p)
        pfile.close()
        create_prompts2wlist(exptdir)
        test(['perl',exptdir+"prompts2wlist",exptdir+"prompts",exptdir+"wlist1",exptdir+"wlist1"])
        lines = readAndAddLines(exptdir+"wlist1", ['SENT-START', 'SENT-END'],
                                exptdir+"wlist")
        shutil.copy(srcdir+"lexicon.ded",exptdir+"lexicon.ded")
        test(['HDMan','-m','-w',exptdir+"wlist",exptdir+"dict1",exptdir+"lexicon.ded",exptdir+"dlog"])
```

139
```python
check_short_pause(exptdir+'monophones1',exptdir)
add_sps(exptdir+'dict1',exptdir)
# NB: monophones1 = monophones0 + sp; monophones1 file is auto-generated from
# HDMan
create_directories(exptdir)
shutil.copy(srcdir+'testprompts',exptdir+'recfiles\'+"testprompts.txt")
create_prompts2mlf(exptdir)
test(['perl',exptdir+'prompts2mlf',exptdir+'words.mlf',exptdir+'prompts'])
create_mkphones0_file(exptdir)
test(['HLEd','-l','-1','*','-d',exptdir+'dict1','-i',exptdir+'phones0.mlf',
      exptdir+'mkphones0.led',exptdir+'words.mlf'])
create_wav_config(exptdir)
# manually: codetr.scp (paths for wav and mfc files)
if problems == []:
    shutil.copy(srcdir+'codetr.scp',exptdir+'codetr.scp')
if not problems == []:
    codetr = open(exptdir+'codetr.scp',r').readlines()
    codetr = remove_problems(codetr, problems)
    cfile = open(exptdir+'codetr.scp','w')
    for c in codetr:
        cfile.write(c)
    cfile.close()
create_temp_dirs(home,srcdir)
test(['HCopy','-A','-D','-V','-T','1','-C',exptdir+'config','-S',exptdir+'
      codetr.scp'])
if os.path.isdir(exptdir+'wav'):
    shutil.rmtree(exptdir+'wav')
shutil.move(home+'wav',exptdir+'wav')
# HTK commands for Creating Monophones HMMs (Steps 6 - 8)
create_proto_file(exptdir)
# manually: train.scp ( location of training files - mfc files)
if problems == []:
    shutil.copy(srcdir+'train.scp',exptdir+'train.scp')
if not problems == []:
    train = open(exptdir+'train.scp',r').readlines()
    train = remove_problems(train, problems)
    tfile = open(exptdir+'train.scp','w')
    for t in train:
        tfile.write(t)
    tfile.close()
create_config(exptdir)
test(['HCompV','-C',exptdir+'config1','-f','0.01','-m','-S',exptdir+'train.
      scp','-M',exptdir+'hmm0',exptdir+'proto'])
# create macros file
ps = open(exptdir+'hmm0\proto','r').readlines()
vs = open(exptdir+'hmm0\vFloors','r').readlines()
ms = genMacrosfromProto(ps,vs)
mf = open(exptdir+'hmm0\macros','w')
mf.write(ms)
mf.close()
# monophones0 file is done by deleting sp from monophones1 file
create_monophones0_file(exptdir+'monophones1',exptdir)
# get hmmdefs file in hmm0 automatically
```
import subprocess

phones = genHMMfromMPhones(exptdir+'monophones0')
genHMMFromProto(phones, exptdir, exptdir+'hmm0/hmmdefs')

test(["HERest", '-C', exptdir+'phones0.mlf', '-t',
                 '250.0', '150.0', '1000.0', '-S', exptdir+'train.scp', '-H',
                 exptdir+'hmm0/macros', '-H', exptdir+'hmm0/hmmdefs', '-M',
                 exptdir+'hmm1', exptdir+'monophones0'])

test(["HERest", '-C', exptdir+'phones0.mlf', '-t',
                 '250.0', '150.0', '1000.0', '-S', exptdir+'train.scp', '-H',
                 exptdir+'hmm1/macros', '-H', exptdir+'hmm1/hmmdefs', '-M',
                 exptdir+'hmm2', exptdir+'monophones0'])

test(["HERest", '-C', exptdir+'phones0.mlf', '-t',
                 '250.0', '150.0', '1000.0', '-S', exptdir+'train.scp', '-H',
                 exptdir+'hmm2/macros', '-H', exptdir+'hmm2/hmmdefs', '-M',
                 exptdir+'hmm3', exptdir+'monophones0'])

# contents of hmm3 copied to hmm4, and hmmdefs file modified by adding "sp"
model
shutil.copy(exptdir+'hmm3/hmmdefs', exptdir+'hmm4/hmmdefs')
shutil.copy(exptdir+'hmm3/macros', exptdir+'hmm4/macros')
s0 = open(exptdir+'hmm4/hmmdefs', "r").read()
s1 = modifySilencePattern(s0)
f = open(exptdir+'hmm4/hmmdefs', "w")
f.write(s1)
f.close()
create_sil_file(exptdir)

test(["HHEd", '-H', exptdir+'hmm4/macros', '-H', exptdir+'hmm4/hmmdefs', '-M',
          exptdir+'hmm5', exptdir+'s1.hed', exptdir+'monophones1'])
create_mkphones1_file(exptdir)

test(["HLEd", '-l', '*', '-d', exptdir+'dict1', '-i',
          exptdir+'phones1.mlf',
          exptdir+'mkphones1.led', exptdir+'words.mlf'])

test(["HERest", '-C', exptdir+'phones1.mlf', '-t',
                 '250.0', '150.0', '1000.0', '-S', exptdir+'train.scp', '-H',
                 exptdir+'hmm5/macros', '-H', exptdir+'hmm5/hmmdefs', '-M',
                 exptdir+'hmm6', exptdir+'monophones1'])

test(["HERest", '-C', exptdir+'phones1.mlf', '-t',
                 '250.0', '150.0', '1000.0', '-S', exptdir+'train.scp', '-H',
                 exptdir+'hmm6/macros', '-H', exptdir+'hmm6/hmmdefs', '-M',
                 exptdir+'hmm7', exptdir+'monophones1'])

lines = readAndAddLines(exptdir+'dict1', ['silence_sil'], exptdir+'dict')
x = test(["HVite", '-l', '*', '-o', 'SWT', '-b', 'silence', '-C', exptdir+'config1',
          '-a', '-H', exptdir+'hmm7/macros', '-H',
          exptdir+'aligned.mlf', '-m', '-t', '250.0', '-y', 'lab', '-I',
          exptdir+'words.mlf', '-S',
          exptdir+'train.scp', exptdir+"\"dict\"", exptdir+'monophones1'],
          stdout = subprocess.PIPE)
problemCases = findProblemCases(x)
if not problemCases == []:
    sys.stdout.write('The following files caused problems at this point:

' % (problemCases))
sys.stdout.flush()
train_htk(srcdir0, exptdir0, problems=problemCases)
return
APPENDIX B. PYTHON SCRIPT FOR THE HTK TRAINING

```python
# HTK commands for Creating Tied-State Triphones (Steps 9 - 10)
create_mktriled_file(exptdir)
test(["HLEd","-n","triphones1","-l","*","-i",exptdir+'wintri.mlf',exptdir+'mktri.led',exptdir+'aligned.mlf'])
create_maketrihed(exptdir)
test(["perl",exptdir+'maketrihed',exptdir+'monophones1','triphones1'])
if os.path.isfile(exptdir+'mktri.hed'):
    os.remove(exptdir+'mktri.hed')
shutil.move(home+'mktri.hed',exptdir)
test(["HHEd","-B","-H",exptdir+'hmm9/macros',"-H",exptdir+'hmm9/hmdefs',"-M",exptdir+'hmm10',exptdir+'mktri.hed',exptdir+'monophones1'])
if os.path.isfile(exptdir+'triphones1'):
    os.remove(exptdir+'triphones1')
shutil.move(home+'triphones1',exptdir)
test(["HERest","-B","-C",exptdir+'config1',"-I",exptdir+'wintri.mlf',"-t","250.0","150.0","1000.0","-S",exptdir+'train.scp',"-H",exptdir+'hmm10/macros',"-H",exptdir+'hmm10/hmdefs',"-M",exptdir+'hmm11',exptdir+'triphones1'])
test(["HERest","-B","-C",exptdir+'config1',"-I",exptdir+'wintri.mlf',"-t","250.0","150.0","1000.0","-S",exptdir+'train.scp',"-H",exptdir+'hmm10/macros',"-H",exptdir+'hmm10/hmdefs',"-M",exptdir+'hmm12',exptdir+'triphones1'])

test(["HRest","-C",exptdir+'config1',"-I",exptdir+'aligned.mlf',"-t","250.0","150.0","1000.0","-S",exptdir+'train.scp',"-H",exptdir+'hmm7/macros',"-H",exptdir+'hmm7/hmdefs',"-M",exptdir+'hmm8',exptdir+'monophones1'])
test(["HERest","-C",exptdir+'config1',"-I",exptdir+'aligned.mlf',"-t","250.0","150.0","1000.0","-S",exptdir+'train.scp',"-H",exptdir+'hmm8/macros',"-H",exptdir+'hmm8/hmdefs',"-M",exptdir+'hmm9',exptdir+'monophones1'])
```

143
os.remove(os.path.join(exptdir, 'mkclscript.prl'))
shutil.move(os.path.join(home, 'mkclscript.prl'), exptdir)

# append fixed lines to the end of tree.hed file
append_tree_tail(exptdir)

shutil.copy(os.path.join(exptdir, 'stats'), home + '/stats')
shutil.copy(os.path.join(exptdir, 'fulllist'), home + '/fulllist')
test(['HHEd', '-B', '-H', os.path.join(exptdir, 'hmm12/macros'), '-H', exptdir + 'hmm12/hmmdefs', '-M', exptdir + 'hmm13', exptdir + 'tree.hed', exptdir + 'triphones1'])

148
os.remove(os.path.join(home, 'stats'))
if os.path.isfile(os.path.join(exptdir, 'trees')):
    os.remove(os.path.join(exptdir, 'trees'))
shutil.move(os.path.join(home, 'trees'), exptdir)
if os.path.isfile(os.path.join(exptdir, 'tiedlist')):
    os.remove(os.path.join(exptdir, 'tiedlist'))
shutil.move(os.path.join(home, 'tiedlist'), exptdir)
test(['HERest', '-B', '-C', os.path.join(exptdir, 'config1'), '-I', os.path.join(exptdir, 'wintri.mlf'), '-t', 250.0, 150.0, '1000.0', '-s', os.path.join(exptdir, 'stats'), '-S', os.path.join(exptdir, 'train.scp'), '-H', os.path.join(exptdir, 'hmm13/macros'), '-H', os.path.join(exptdir, 'hmm13/hmmdefs'), '-M', os.path.join(exptdir, 'hmm14'), os.path.join(exptdir, 'tiedlist1')])

test(['HERest', '-B', '-C', os.path.join(exptdir, 'config1'), '-I', os.path.join(exptdir, 'wintri.mlf'), '-t', 250.0, 150.0, '1000.0', '-s', os.path.join(exptdir, 'stats'), '-S', os.path.join(exptdir, 'train.scp'), '-H', os.path.join(exptdir, 'hmm14/macros'), '-H', os.path.join(exptdir, 'hmm14/hmmdefs'), '-M', os.path.join(exptdir, 'hmm15'), os.path.join(exptdir, 'tiedlist1')])
if os.path.isdir(os.path.join(exptdir, 'mfcc')):
    shutil.rmtree(os.path.join(exptdir, 'mfcc'))
shutil.move(os.path.join(home, 'mfcc'), os.path.join(exptdir, 'mfcc'))
Appendix C

Dynamic Time Warping

This appendix contains the python code of the Dynamic Time Warping (DTW) algorithm which is used to perform the alignment process for both morphing in Chapter 5 and speech synthesis in Chapter 6.

dtw.py

```python
import math

EXCHANGE = 1.5
INSDEL = 20

def isnumber(x):
    t = x.__class__.__name__
    return t in ['int', 'float']

def istuple(x):
    return x.__class__.__name__ == 'tuple'

class link:
    def dist(self, x, y):
        if isnumber(x) and isnumber(y):
            return abs(x-y)
        if istuple(x) and istuple(y):
            t = 0.0
            for i in range(0, len(x)):
                t = t+(x[i]-y[i])**2
            return math.sqrt(t)
        if x == y:
            s = 0
        else:
            s = EXCHANGE
        return s

    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.link = False
        self.value = '/uni2423'
```

144
def extend(self, dx, dy, a):
    array = a.array
    x = self.x+dx
    y = self.y+dy
    if x >= len(a.v1) or y >= len(a.v2):
        return
    if dx == 1 and dy == 1:
        s = self.dist(a.v1[x], a.v2[y])
    else:
        s = INSDEL
    other = array[x][y]
    if other.value == '\uni2423' or self.value+s < other.value:
        other.value = self.value+s
        other.link = self

class array:
    def __init__(self, v1, v2):
        self.v1 = v1
        self.v2 = v2
        self.array = [[link(i, j) for j in range(0, len(v2))] for i in range(0, len(v1))]

    def findPath(self):
        self.array[0][0].value = 0
        for j in range(0, len(self.array[0])-1):
            for i in range(0, len(self.array)):
                self.tabular()  # print self.tabular()
                l = self.array[i][j]
                l.extend(0, 1, self)
                l.extend(1, 1, self)
                l.extend(1, 0, self)
        return self.array[-1][-1]

    def show(self):
        s = ''
        for j in range(0, len(self.array[0])):
            for i in range(0, len(self.array)):
                s = s+str(self.array[i][j].value)+'/uni2423
        return s

    def tabular(self):
        s = r'\BREAK\VPARA\begin{tabular}{'+('c'*(len(self.array)+1))+'}
        for i in range(0, len(self.v1)):
            s = s+'~~&~~'+str(self.v1[i])
            s = s+'\\
        for j in range(0, len(self.v2)):
            s = s+r'/uni2423&/uni2423\Rnode{n%s%s}{\texttt{%s}}'%(i, j, str(self.array[i][j].value))
            s = s+('&'*len(self.v1))+'\\
        s = s+'\end{tabular}\n'
        for j in range(0, len(self.array[0])):
            for i in range(0, len(self.array)):
                l = self.array[i][j]
                if l.link:
APPENDIX C. DYNAMIC TIME WARping

```
s = s+'\ncline[linewidth=2pt,nodesep=5pt]{->}{n%s%s}{n%s%s}
'%(l.x, l.y, l.link.x, l.link.y)

return s

def showPath(p, a):
v1 = a.v1
v2 = a.v2
path = []
while p:
    l = p.link
    if l:
        dx = p.x-l.x
dy = p.y-l.y
        if dx == 1 and dy == 1:
            path = [(v1[p.x], v2[p.y])]+path
        elif dx == 1:
            path = [(v1[p.x], '*')]+path
        else:
            path = [('*', v2[p.y])]+path
        p = l
    path = [(v1[0], v2[0])]+path
return path

def javapath(p, feature, image1, image2):
s = ""
    int %s%s%s[] = new int{""%(feature, image1, image2)
for x in p:
    a = x[0]
b = x[1]
    if not istuple(a):
        a = b
        s = s+"{%s,%s},\n"%(a[0], a[1])
    s = s[:-2]+'""
    int %s%s%s[] = new int{""%(feature, image2, image1)
for x in p:
    a = x[0]
b = x[1]
    if not istuple(b):
        b = a
        s = s+"{%s,%s},\n"%(b[0], b[1])
    s = s[:-2]+'""
Morph morph%s%s%s = new Morph(%s%s%s, %s%s%s);
""%(feature, image1, feature, image2, feature, image1, image2, feature, image2, image1)
return s
```
Appendix D

Morphing Code

This appendix contains the Java code of the morphing from one drawing to another as described in Section 5.1.1 in order to animate the vocal tract as feedback to Arabic learners:

```java
public Hashtable makeMorphs(){
    Hashtable table = new Hashtable();
    String names[] = {"Q","l","q","p","Z","D","h","T","d","S","b","i","fa","E","*","a",
                     "k","m","H","n","s","r","t","w","y","z","tha"};
    int segments[] = {
        /* upperLip */ 14,
        /* upperJaw */ 44,
        /* lowerJaw */ 28,
        /* lowerTeeth */ 8,
        /* upperTeeth */ 7,
        /* nose */ 81,
        /* throat */ 33,
        /* tongue */ 19,
        /* lowerLip */ 31};

    // reading the coords from ‘txtPile.txt’ generated from dtw.py
    float[][][] coords = readCoords();
    // End of reading coords in ‘txtPile.txt’

    for(int i=0; i<names.length; i++){
        GeometryArray tableValue = createGeomArray(segments, coords[i]); // i <=> a picture
        table.put(names[i],tableValue);
    }
}
```
public BranchGroup makeMorph(String testPics[], Hashtable table) {
    GeometryArray[] geomArray = new GeometryArray[testPics.length];
    for (int i = 0; i < testPics.length; i++) {
        geomArray[i] = (GeometryArray)table.get(testPics[i]);
    }
    Transform3D t3d = new Transform3D();
    TransformGroup translate = new TransformGroup();
    t3d.setScale(1.0);
    translate.setTransform(t3d);
    TransformGroup objTrans[] = new TransformGroup[testPics.length+1];
    for (int i = 0; i < testPics.length+1; i++) {
        objTrans[i] = new TransformGroup();
        translate.addChild(objTrans[i]);
    }
    Transform3D tr = new Transform3D();
    Transform3D rotY15 = new Transform3D();
    rotY15.rotY(15.0 * Math.PI / 180.0);
    for (int i = 0; i < testPics.length+1; i++) {
        objTrans[i].getTransform(tr);
        tr.setTranslation(new Vector3d(-1.0, 1.0, -3.0));
        tr.mul(rotY15);
        objTrans[i].setTransform(tr);
    }
    Appearance app = createAppearance();
    Morph morphObj = new Morph((GeometryArray[]) geomArray, app);
    morphObj.setCapability(Morph.ALLOW_WEIGHTS_READ);
    morphObj.setCapability(Morph.ALLOW_WEIGHTS_WRITE);
    objTrans[testPics.length].addChild(morphObj);
    // create alpha object – to perform timing our morphing operation
    // Alpha converts time values to alpha values between 0 and 1 (generate a single ramp from 0 to 1).
    int dur = 1500*testPics.length;
    Alpha alpha =
        new Alpha(1, // loop count, -1 ==> continuously loop
            Alpha.INCREASING_ENABLE, // the mode
            1000, // trigger time
0, // phase delay time, to make sure that j3d has enough time to bring up the
window
dur, // the ramp takes 2 sec (for each pic) to complete =
increasingAlphaDuration
0, // increasingRampDuration = acceleration
0, // AlphaAtOneDuration
0, // decreasingAlphaDuration
0, // decreasingRampDuration = acceleration
0); // AlphaAtZeroDuration

// create morph driving behavior
morphBehav = new MorphBehavior(morphObj, alpha, testPics.length); // , testPics.length
morphBehav.setSchedulingBounds(new BoundingSphere()); // region where action takes place
BranchGroup scene = createSceneGraph(morphObj, morphBehav, translate);
double w[] = morphBehav.weights;
return scene;
}
Appendix E

Top-Level System Integration

This appendix contains the code that implements the integration between Java, Python, and the HTK.

```java
import java.lang.Runtime;
import java.io.*;

public void actionPerformed(ActionEvent ae){
    if(record_stopB.getLabel().equals("Stop")){
        // save wav file
        record_stopB.setLabel("Record");
        capture.stop(sentenceCounter);
        new RunTheHTK(); // call the Python to run the HTK
        System.out.println("AFTER/uni2423HTK");
        DisplayResult();
    } else {
        startRecording();
    }
}

public class RunTheHTK {
    RunTheHTK(){
        Runtime rt = Runtime.getRuntime(); // to access the current runtime environment
        try{
```
Process p = rt.exec("python_scripts.py src_exp"); // Call Python to run the HTK
{
}

catch(Exception e){
    System.out.println(e);
}

def integWithJava(args):
    gen_mfc_test_files(args[0],args[1])  # generate the standard format of the HTK (mfc files)
        # from wav files
    recout4animation(args[0],args[1])  # generate a text file ('phonemes.txt') containing the
        # recognised phones
    correctPhones(args[0])  # Put the target phones in a file called 'correct.txt'
    sys.stdout.close()
    sys.exit()
Appendix F

Buckwalter Arabic transliteration

This appendix contains the Buckwalter Arabic transliteration which is used in our research. This transliteration was developed at Xerox by Tim Buckwalter in the 1990s. It is an ASCII only transliteration scheme, representing Arabic orthography strictly one-to-one.
Table F.1: Buckwalter Arabic transliteration table

<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>Transliteration</th>
<th>Letter Name</th>
<th>Unicode Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ALEF</td>
<td>u0627</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ALEF WITH HAMZA ABOVE</td>
<td>u0623</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ALEF WITH HAMZA UNDER</td>
<td>u0625</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>ALEF WITH MADDA ABOVE</td>
<td>u0622</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>BEH</td>
<td>u0628</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>TEH</td>
<td>u062A</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>THEH</td>
<td>u062B</td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>JEEM</td>
<td>u062C</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>HAH</td>
<td>u062D</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>KHAH</td>
<td>u062E</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>DAL</td>
<td>u062F</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>THAL</td>
<td>u0630</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>REH</td>
<td>u0631</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>ZAIN</td>
<td>u0632</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>SEEN</td>
<td>u0633</td>
<td></td>
</tr>
<tr>
<td>$</td>
<td>SHEEN</td>
<td>u0634</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>SAD</td>
<td>u0635</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>DAD</td>
<td>u0636</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>TAH</td>
<td>u0637</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>ZAH</td>
<td>u0638</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>AIN</td>
<td>u0639</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>GHAIN</td>
<td>u063A</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>FEH</td>
<td>u0641</td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>QAF</td>
<td>u0642</td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>KAF</td>
<td>u0643</td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>LAM</td>
<td>u0644</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>MEEM</td>
<td>u0645</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>NOON</td>
<td>u0646</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>HEH</td>
<td>u0647</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>TEH MARBUTA</td>
<td>u0629</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>WAW</td>
<td>u0648</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>YEH</td>
<td>u064A</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>FATHA</td>
<td>u064E</td>
<td></td>
</tr>
<tr>
<td>u</td>
<td>DAMMA</td>
<td>u064F</td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>KASRA</td>
<td>u0650</td>
<td></td>
</tr>
<tr>
<td>aN</td>
<td>TANWIN ALFATH</td>
<td>u064B</td>
<td></td>
</tr>
<tr>
<td>uN</td>
<td>TANWIN ALDAM</td>
<td>u064C</td>
<td></td>
</tr>
<tr>
<td>iN</td>
<td>TANWIN ALKASER</td>
<td>u064D</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SHADDA</td>
<td>u0651</td>
<td></td>
</tr>
</tbody>
</table>
Appendix G

Modified Buckwalter Arabic transliteration

This appendix contains Buckwalter Arabic transliteration including our modifications. As compared to the default Buckwalter Arabic transliteration, we modified four characters ‘>’ (i.e., ُّ), < (i.e., ۸), | (i.e., ٰ), and $ (i.e., ﷩) into Q, ’i, ’A, and ֵ, respectively. These modifications were made in order to avoid causing a contradiction with the special characters used in the grammar file of the HTK.
<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>Transliteration</th>
<th>Letter Name</th>
<th>Unicode Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>ALEF</td>
<td>ALEF</td>
<td>u0627</td>
</tr>
<tr>
<td>Q</td>
<td>ALEF WITH HAMZA ABOVE</td>
<td>ALEF WITH HAMZA ABOVE</td>
<td>u0623</td>
</tr>
<tr>
<td>i</td>
<td>ALEF WITH HAMZA UNDER</td>
<td>ALEF WITH HAMZA UNDER</td>
<td>u0625</td>
</tr>
<tr>
<td>'A</td>
<td>ALEF WITH MADDA ABOVE</td>
<td>ALEF WITH MADDA ABOVE</td>
<td>u0622</td>
</tr>
<tr>
<td>b</td>
<td>BEH</td>
<td>BEH</td>
<td>u0628</td>
</tr>
<tr>
<td>t</td>
<td>TEH</td>
<td>TEH</td>
<td>u062A</td>
</tr>
<tr>
<td>v</td>
<td>THEH</td>
<td>THEH</td>
<td>u062B</td>
</tr>
<tr>
<td>j</td>
<td>JEEM</td>
<td>JEEM</td>
<td>u062C</td>
</tr>
<tr>
<td>H</td>
<td>HAH</td>
<td>HAH</td>
<td>u062D</td>
</tr>
<tr>
<td>x</td>
<td>KHAH</td>
<td>KHAH</td>
<td>u062E</td>
</tr>
<tr>
<td>d</td>
<td>DAL</td>
<td>DAL</td>
<td>u062F</td>
</tr>
<tr>
<td>*</td>
<td>THAL</td>
<td>THAL</td>
<td>u0630</td>
</tr>
<tr>
<td>r</td>
<td>REH</td>
<td>REH</td>
<td>u0631</td>
</tr>
<tr>
<td>z</td>
<td>ZAIN</td>
<td>ZAIN</td>
<td>u0632</td>
</tr>
<tr>
<td>s</td>
<td>SEEN</td>
<td>SEEN</td>
<td>u0633</td>
</tr>
<tr>
<td>`s</td>
<td>SHEEN</td>
<td>SHEEN</td>
<td>u0634</td>
</tr>
<tr>
<td>S</td>
<td>SAD</td>
<td>SAD</td>
<td>u0635</td>
</tr>
<tr>
<td>D</td>
<td>DAD</td>
<td>DAD</td>
<td>u0636</td>
</tr>
<tr>
<td>T</td>
<td>TAH</td>
<td>TAH</td>
<td>u0637</td>
</tr>
<tr>
<td>Z</td>
<td>ZAH</td>
<td>ZAH</td>
<td>u0638</td>
</tr>
<tr>
<td>E</td>
<td>AIN</td>
<td>AIN</td>
<td>u0639</td>
</tr>
<tr>
<td>g</td>
<td>GHAIN</td>
<td>GHAIN</td>
<td>u063A</td>
</tr>
<tr>
<td>f</td>
<td>FEH</td>
<td>FEH</td>
<td>u0641</td>
</tr>
<tr>
<td>q</td>
<td>QAF</td>
<td>QAF</td>
<td>u0642</td>
</tr>
<tr>
<td>k</td>
<td>KAF</td>
<td>KAF</td>
<td>u0643</td>
</tr>
<tr>
<td>l</td>
<td>LAM</td>
<td>LAM</td>
<td>u0644</td>
</tr>
<tr>
<td>m</td>
<td>MEEM</td>
<td>MEEM</td>
<td>u0645</td>
</tr>
<tr>
<td>n</td>
<td>NOON</td>
<td>NOON</td>
<td>u0646</td>
</tr>
<tr>
<td>h</td>
<td>HEH</td>
<td>HEH</td>
<td>u0647</td>
</tr>
<tr>
<td>p</td>
<td>TEH MARBUTA</td>
<td>TEH MARBUTA</td>
<td>u0629</td>
</tr>
<tr>
<td>w</td>
<td>WAW</td>
<td>WAW</td>
<td>u0648</td>
</tr>
<tr>
<td>y</td>
<td>YEH</td>
<td>YEH</td>
<td>u064A</td>
</tr>
<tr>
<td>a</td>
<td>FATHA</td>
<td>FATHA</td>
<td>u064E</td>
</tr>
<tr>
<td>aN</td>
<td>TANWIN ALFATH</td>
<td>TANWIN ALFATH</td>
<td>u064B</td>
</tr>
<tr>
<td>uN</td>
<td>TANWIN ALDAM</td>
<td>TANWIN ALDAM</td>
<td>u064C</td>
</tr>
<tr>
<td>iN</td>
<td>TANWIN ALKASER</td>
<td>TANWIN ALKASER</td>
<td>u064D</td>
</tr>
<tr>
<td>[]</td>
<td>SHADDA</td>
<td>SHADDA</td>
<td>u0651</td>
</tr>
</tbody>
</table>
Appendix H

IPA for Arabic

The table below shows the way in which the International Phonetic Alphabet (IPA) represents the form of Arabic.
Table H.1: IPA symbols for Arabic

<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>IPA</th>
<th>English approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ا</td>
<td>æ</td>
<td>as in at</td>
</tr>
<tr>
<td>أ</td>
<td>?</td>
<td>Glottal Stop</td>
</tr>
<tr>
<td>ب</td>
<td>b</td>
<td>as in book</td>
</tr>
<tr>
<td>ت</td>
<td>t</td>
<td>as in step</td>
</tr>
<tr>
<td>ث</td>
<td>e</td>
<td>as in thing</td>
</tr>
<tr>
<td>ج</td>
<td>dʒ</td>
<td>as in jam</td>
</tr>
<tr>
<td>ح</td>
<td>fi</td>
<td>NA</td>
</tr>
<tr>
<td>خ</td>
<td>x</td>
<td>as in loch (Scottish)</td>
</tr>
<tr>
<td>د</td>
<td>d</td>
<td>as in deed</td>
</tr>
<tr>
<td>ذ</td>
<td>ð</td>
<td>as in this</td>
</tr>
<tr>
<td>ر</td>
<td>r</td>
<td>as in rule</td>
</tr>
<tr>
<td>ز</td>
<td>z</td>
<td>as in zoo</td>
</tr>
<tr>
<td>س</td>
<td>s</td>
<td>as in see</td>
</tr>
<tr>
<td>ج</td>
<td>ḟ</td>
<td>as in she</td>
</tr>
<tr>
<td>ر</td>
<td>sʰ</td>
<td>roughly as in message</td>
</tr>
<tr>
<td>ض</td>
<td>dʰ</td>
<td>roughly as in dark</td>
</tr>
<tr>
<td>ت</td>
<td>tʰ</td>
<td>roughly as in star</td>
</tr>
<tr>
<td>ظ</td>
<td>ḍʰ</td>
<td>roughly as in thus</td>
</tr>
<tr>
<td>ف</td>
<td>f</td>
<td>NA</td>
</tr>
<tr>
<td>ي</td>
<td>y</td>
<td>as in French R</td>
</tr>
<tr>
<td>ق</td>
<td>q</td>
<td>roughly as in good</td>
</tr>
<tr>
<td>ك</td>
<td>k</td>
<td>as in king</td>
</tr>
<tr>
<td>ل</td>
<td>l</td>
<td>as in bell</td>
</tr>
<tr>
<td>م</td>
<td>m</td>
<td>as in man</td>
</tr>
<tr>
<td>ن</td>
<td>n</td>
<td>as in no</td>
</tr>
<tr>
<td>ه</td>
<td>h</td>
<td>as in he</td>
</tr>
<tr>
<td>و</td>
<td>w</td>
<td>as in we</td>
</tr>
<tr>
<td>ي</td>
<td>j</td>
<td>as in yes</td>
</tr>
</tbody>
</table>
Appendix I

Modified IPA for Arabic

The table below shows the modified IPA which we use it in our research. The modifications include the emphatic sounds of Arabic (i.e., ظ, ط, ض, ص) and other sounds having symbols that are not accepted by the HTK.
<table>
<thead>
<tr>
<th>Arabic Letter</th>
<th>IPA</th>
<th>English approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ا</td>
<td>aa</td>
<td>as in at</td>
</tr>
<tr>
<td>أ</td>
<td>Q</td>
<td>Glottal Stop</td>
</tr>
<tr>
<td>ب</td>
<td>b</td>
<td>as in book</td>
</tr>
<tr>
<td>ت</td>
<td>t</td>
<td>as in step</td>
</tr>
<tr>
<td>ؤ</td>
<td>e</td>
<td>as in thing</td>
</tr>
<tr>
<td>دج</td>
<td>d3</td>
<td>as in jam</td>
</tr>
<tr>
<td>ح</td>
<td>h^</td>
<td>NA</td>
</tr>
<tr>
<td>خ</td>
<td>x</td>
<td>as in loch (Scottish)</td>
</tr>
<tr>
<td>د</td>
<td>d</td>
<td>as in deed</td>
</tr>
<tr>
<td>ح^</td>
<td>D^</td>
<td>as in this</td>
</tr>
<tr>
<td>ر</td>
<td>r</td>
<td>as in rule</td>
</tr>
<tr>
<td>ز</td>
<td>z</td>
<td>as in zoo</td>
</tr>
<tr>
<td>س</td>
<td>s</td>
<td>as in see</td>
</tr>
<tr>
<td>ف</td>
<td>f</td>
<td>as in she</td>
</tr>
<tr>
<td>سˆ</td>
<td>sˆ</td>
<td>roughly as in message</td>
</tr>
<tr>
<td>دˆ</td>
<td>dˆ</td>
<td>roughly as in dark</td>
</tr>
<tr>
<td>تˆ</td>
<td>tˆ</td>
<td>roughly as in star</td>
</tr>
<tr>
<td>زˆ</td>
<td>zˆ</td>
<td>roughly as in thus</td>
</tr>
<tr>
<td>إˆ</td>
<td>E^</td>
<td>NA</td>
</tr>
<tr>
<td>ى</td>
<td>y</td>
<td>as in French R</td>
</tr>
<tr>
<td>ق</td>
<td>q</td>
<td>roughly as in good</td>
</tr>
<tr>
<td>ك</td>
<td>k</td>
<td>as in king</td>
</tr>
<tr>
<td>ل</td>
<td>l</td>
<td>as in bell</td>
</tr>
<tr>
<td>م</td>
<td>m</td>
<td>as in man</td>
</tr>
<tr>
<td>ن</td>
<td>n</td>
<td>as in no</td>
</tr>
<tr>
<td>ه</td>
<td>h</td>
<td>as in he</td>
</tr>
<tr>
<td>و</td>
<td>w</td>
<td>as in we</td>
</tr>
<tr>
<td>ي</td>
<td>y</td>
<td>as in yes</td>
</tr>
</tbody>
</table>
Appendix J

Arabic Minimal Pairs

This appendix contains the Arabic words that have been used as minimal pairs in our experiments. These words contain confusables phonemes that form a difficulty to pronounce by students learning Arabic. The HTK is trained to ensure that it can recognise these words and differentiate each one with its counterpart.

<table>
<thead>
<tr>
<th>Arabic Word</th>
<th>Transliteration</th>
</tr>
</thead>
<tbody>
<tr>
<td>يصبر</td>
<td>yaSbir</td>
</tr>
<tr>
<td>يسبر</td>
<td>yasbir</td>
</tr>
<tr>
<td>ضرب</td>
<td>Darb</td>
</tr>
<tr>
<td>درب</td>
<td>darb</td>
</tr>
<tr>
<td>طبن</td>
<td>Tyn</td>
</tr>
<tr>
<td>تين</td>
<td>tyn</td>
</tr>
<tr>
<td>طل</td>
<td>Zal</td>
</tr>
<tr>
<td>ذل</td>
<td>*al</td>
</tr>
<tr>
<td>يأمر</td>
<td>yaQmur</td>
</tr>
<tr>
<td>يعمر</td>
<td>yaEmur</td>
</tr>
<tr>
<td>همزه</td>
<td>hamzah</td>
</tr>
<tr>
<td>حمزه</td>
<td>Hamzah</td>
</tr>
<tr>
<td>كله</td>
<td>kul</td>
</tr>
<tr>
<td>قبل</td>
<td>qul</td>
</tr>
</tbody>
</table>
Appendix K

Behavioural Traces

This appendix contains the behavioral traces for participants in the classroom experiments (Chapter 7). These traces were grouped according to the version type: language lab (Section K.1), animation (Section K.2), synthesis (Section K.3), instruction (Section K.4), full (Section K.5) as shown below.

K.1 User behaviours of language lab version

Figure K.1: User behaviour in the language lab version
Figure K.2: User behaviour in the language lab version

Figure K.3: User behaviour in the language lab version

Figure K.4: User behaviour in the language lab version
K.1. USER BEHAVIOURS OF LANGUAGE LAB VERSION

Figure K.5: User behaviour in the language lab version

Figure K.6: User behaviour in the language lab version

Figure K.7: User behaviour in the language lab version
Figure K.8: User behaviour in the language lab version
K.2 User behaviours of animation version

Figure K.9: User behaviour in the animation version

Figure K.10: User behaviour in the animation version
Figure K.11: User behaviour in the animation version

Figure K.12: User behaviour in the animation version

Figure K.13: User behaviour in the animation version
K.2. USER BEHAVIOURS OF ANIMATION VERSION

Figure K.14: User behaviour in the animation version

Figure K.15: User behaviour in the animation version

Figure K.16: User behaviour in the animation version
K.3 User behaviours of synthesis version

Figure K.17: User behaviour in the synthesis version

Figure K.18: User behaviour in the synthesis version
K.3. USER BEHAVIOURS OF SYNTHESIS VERSION

Figure K.19: User behaviour in the synthesis version

Figure K.20: User behaviour in the synthesis version

Figure K.21: User behaviour in the synthesis version
Figure K.22: User behaviour in the synthesis version

Figure K.23: User behaviour in the synthesis version

Figure K.24: User behaviour in the synthesis version
K.4 User behaviours of instruction version

Figure K.25: User behaviour in the instruction version

Figure K.26: User behaviour in the instruction version
Figure K.27: User behaviour in the instruction version

Figure K.28: User behaviour in the instruction version

Figure K.29: User behaviour in the instruction version
Figure K.30: User behaviour in the instruction version

Figure K.31: User behaviour in the instruction version

Figure K.32: User behaviour in the instruction version
K.5 User behaviours of full version

Figure K.33: User behaviour in the full version

Figure K.34: User behaviour in the full version
Figure K.35: User behaviour in the full version

Figure K.36: User behaviour in the full version
Figure K.37: User behaviour in the full version

Figure K.38: User behaviour in the full version
K.5. USER BEHAVIOURS OF FULL VERSION

Figure K.39: User behaviour in the full version

Figure K.40: User behaviour in the full version