Speech Verification for Computer Assisted Pronunciation Training

Dissertation

zur Erlangung des akademischen Grades eines
Doktors der Philosophie
der Philosophischen Fakultät
der Universität des Saarlandes

vorgelegt von
Renlong Ai
aus Liaoning, China

Saarbrücken, 2018

Dekan: Prof. Dr. Roland Marti

Berichterstatter: Prof. Dr. Hans Uszkoreit, Prof. Dr. Bernd Möbius

Tag der letzten Prüfungsleistung: 26.06.2018

昔者庄周梦为胡蝶，栩栩然胡蝶也，自喻适志与，不知周也。俄然觉，则蘧蘧然周也。不知周之梦为胡蝶与，胡蝶之梦为周与？周与胡蝶，则必有分矣。此之谓物化。

《庄子·齐物论》

Abstract

Computer assisted pronunciation training (CAPT) is an approach that uses computer technology and computer-based resources in teaching and learning pronunciation. It is also part of computer assisted language learning (CALL) technology that has been widely applied to online learning platforms in the past years. This thesis deals with one of the central tasks in CAPT, i.e. speech verification. The goal is to provide a framework that identifies pronunciation errors in speech data of second language (L2) learners and generates feedback with information and instruction for error correction. Furthermore, the framework is supposed to support the adaptation to new L1-L2 language pairs with minimal adjustment and modification.

The central result is a novel approach to L2 speech verification, which combines both modern language technologies and linguistic expertise. For pronunciation verification, we select a set of L2 speech data, create alias phonemes from the errors annotated by linguists, then train an acoustic model with mixed L2 and gold standard data and perform HTK[1] phoneme recognition to identify the error phonemes. For prosody verification, FD-PSOLA[2] and Dynamic time warping are both applied to verify the differences in duration, pitch and stress. Feedback is generated for both verifications. Our feedback is presented to learners not only visually as with other existing CAPT systems, but also perceptually by synthesizing the learner's own audio, e.g. for prosody verification, the gold standard prosody is transplanted onto the learner's own voice.

The framework is self-adaptable under semi-supervision, and requires only a certain amount of mixed gold standard and annotated L2 speech data for bootstrapping. Verified speech data is validated by linguists, annotated in case of

[1] Hidden Markov Toolkit
[2] Pitch Synchronous Overlap and Add

wrong verification, and used in the next iteration of training. Mary Annotation Tool (MAT) is developed as an open-source component of MARYTTS for both annotating and validating. To deal with uncertain pauses and interruptions in L2 speech, the silence model in HTK is also adapted, and used in all components of the framework where forced alignment is required.

Various evaluations are conducted that help us obtain insights into the applicability and potential of our CAPT system. The pronunciation verification shows high accuracy in both precision and recall, and encourages us to acquire more error-annotated L2 speech data to enhance the trained acoustic model. To test the effect of feedback, a progressive evaluation is carried out and it shows that our perceptual feedback helps learners realize their errors, which they could not otherwise observe from visual feedback and textual instructions. In order to improve the user interface, a questionnaire is also designed to collect the learners' experiences and suggestions.

Zusammenfassung

Computer Assisted Pronunciation Training (CAPT) ist ein Ansatz, der mittels Computer und computergestützten Ressourcen das Erlernen der korrekten Aussprache im Fremdsprachenunterricht erleichtert. Dieser Ansatz ist ein Teil der Computer Assisted Language Learning (CALL) Technologie, die seit mehreren Jahren auf Online-Lernplattformen häufig zum Einsatz kommt. Diese Arbeit ist der Sprachverifikation gewidmet, einer der zentralen Aufgaben innerhalb des CAPT. Das Ziel ist, ein Framework zur Identifikation von Aussprachefehlern zu entwickeln für Menschen, die eine Fremdsprache (L2-Sprache) erlernen. Dabei soll Feedback mit fehlerspezifischen Informationen und Anweisungen für eine richtige Aussprache erzeugt werden. Darüber hinaus soll das Rahmenwerk die Anpassung an neue Sprachenpaare (L1-L2) mit minimalen Adaptationen und Modifikationen unterstützen.

Das zentrale Ergebnis ist ein neuartiger Ansatz für die L2-Sprachprüfung, der sowohl auf modernen Sprachtechnologien als auch auf corpuslinguistischen Ansätzen beruht. Für die Ausspracheüberprüfung erstellen wir Alias-Phoneme aus Fehlern, die von Linguisten annotiert wurden. Dann trainieren wir ein akustisches Modell mit gemischten L2- und Goldstandarddaten und führen eine HTK-Phonemerkennung[3] aus, um die Fehlerphoneme zu identifizieren. Für die Prosodieüberprüfung werden sowohl FD-PSOLA[4] und Dynamic Time Warping angewendet, um die Unterschiede in der Dauer, Tonhöhe und Betonung zwischen dem Gesprochenen und dem Goldstandard zu verifizieren. Feedbacks werden für beide Überprüfungen generiert und den Lernenden nicht nur visuell präsentiert, so wie in anderen vorhandenen CAPT-Systemen, sondern auch perzeptuell vorgestellt. So wird unter anderem für die Prosodieverifikation die Goldstandardprosodie auf die eigene Stimme des Lernenden übergetragen.

[3] Hidden Markov Toolkit
[4] Pitch Synchronous Overlap and Add

Zur Anpassung des Frameworks an weitere L1-L2 Sprachdaten muss das System über Maschinelles Lernen trainiert werden. Da es sich um ein semi-überwachtes Lernverfahren handelt, sind nur eine gewisse Menge an gemischten Goldstandard- und annotierten L2-Sprachdaten für das Bootstrapping erforderlich. Verifizierte Sprachdaten werden von Linguisten validiert, im Falle einer falschen Verifizierung nochmals annotiert, und bei der nächsten Iteration des Trainings verwendet. Für die Annotation und Validierung wurde das Mary Annotation Tool (MAT) als Open-Source-Komponente von MARYTTS entwickelt. Um mit unsicheren Pausen und Unterbrechungen in der L2-Sprache umzugehen, wurde auch das sogenannte Stillmodell in HTK angepasst und in allen Komponenten des Rahmenwerks verwendet, in denen Forced Alignment erforderlich ist.

Unterschiedliche Evaluierungen wurden durchgeführt, um Erkenntnisse über die Anwendungspotenziale und die Beschränkungen des Systems zu gewinnen. Die Ausspracheüberprüfung zeigt eine hohe Genauigkeit sowohl bei der Präzision als auch beim Recall. Dadurch war es möglich weitere fehlerbehaftete L2-Sprachdaten zu verwenden, um somit das trainierte akustische Modell zu verbessern. Um die Wirkung des Feedbacks zu testen, wird eine progressive Auswertung durchgeführt. Das Ergebnis zeigt, dass perzeptive Feedbacks dabei helfen, dass die Lernenden sogar Fehler erkennen, die sie nicht aus visuellen Feedbacks und Textanweisungen beobachten können. Zudem wurden mittels Fragebogen die Erfahrungen und Anregungen der Benutzeroberfläche der Lernenden gesammelt, um das System künftig zu verbessern.

Acknowledgments

I want to thank all my colleagues, friends and relatives who have given me support and encouragement during my work on this dissertation.

Above all, I am deeply grateful to my supervisor Hans Uszkoreit who gave me the chance to research in this cutting edge theme. I have been enlightened by his integral view on research and have learned many things from the discussions and meetings with him, which will be of great benefit to me in the future.

I am especially indebted to my supervisor Feiyu Xu at the DFKI. She has guided and helped me in the past years with extreme enthusiasm and patience. She was always there when I needed her advice and I was always enlightened in a joyful atmosphere. She also directed me in forming the idea of the framework as shown in this thesis. It was a great pleasure to conduct this thesis under her supervision. I have learned a lot from her, especially the courage to come up with new ideas as well as constant skepticism toward both mainstream approaches and one‘s own research directions and results.

I am also deeply grateful to Sven Schmeier, who provided me with a large amount of software and hardware resources which made my research much easier. His vision in discovering potential applications was a very valuable inspiration of the work presented in this thesis.

My special thanks to Marcela Charfuelan for her great guidance and fruitful co-development, particularly in the implementation of integrating MaryTTS components into our system.

My sincere thanks go to Sandra Gasber, Philip Gienandt and Sabine Buckbesch at LinguaTV for their great cooperation. They provided the L2 learner speech data essential to my research.

My special thanks to Hans Uszkoreit, Feiyu Xu, Mary van Genabith and Sven Schmeier, who helped to proofread the dissertation and generously offered numerous corrections and suggestions.

I also want to thank all my colleagues in the DFKI language lab for making it an enjoyable working environment. They helped me a lot in the past eight years and I have learned so much from them.

A final word of gratitude is dedicated to my beloved wife for her love and support and for accompanying me through the late nights during the thesis writing phase, and also to my parents for their eternal understanding and encouragement. They were the sources of love and trust from which I have drawn the energy for managing this task and for coping with critical challenges in the past years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis aims to develop and apply language technologies for computer assisted pronunciation training (CAPT). In recent years, second language (L2) learning has become more and more popular to meet the need of communicating and integrating with a foreign community or society in the globalized world. However, learning a second language takes time and dedication, not only from learners, but also from teachers, in particular, both face-to-face and 24/7 personal online language learning are very expensive. A large and still growing number of computer assisted language learning (CALL) applications in the market has shown a clear trend: language learning is going to be web-based, interactive, multimedia and personalized, so that learners can be flexible as to times and places for learning.

Modern technologies allow software programs to provide comparable performance of human teachers in many aspects of language teaching such as validation of vocabulary and grammars, but as yet not in training pronunciation. Some industrial CALL applications have applied automatic speech recognition (ASR) to the learners' speech and tried to infer the existence of errors from the confidence value in the recognition result. This yields results with low accuracy because no specific model is trained to deal with all possible errors, and furthermore, they could not provide feedback for explanation of the errors. Hence ASR based speech verification is far less effective than traditional classroom teaching. To raise the accuracy of pronunciation error detection, several approaches have been discussed and presented in the literature, including building classifiers with Linear Discriminant Analysis or Decision Tree (Truong et al. 2004),

or using Support Vector Machine (SVM) classifier based on applying transformation on Mel Frequency Cepstral coefficients (MFCC) of the learners' audio data (Picard et al. (2010) and Ananthakrishnan et al. (2011)). However, these methods either involve complex training processes or are limited in scalability, e.g. only feasible for one specific second language, hence these methods haven't been integrated in the current CAPT systems.

Traditional CALL systems have focused on detecting phoneme errors in pronunciation, e.g. mispronouncing /ə/ to /ɔ/, but paid less attention to prosody error detection, or performed primitive prosody verification by simply checking amplitude and pitch value in speech data. Although some systems provide visual feedback like a pitch curve graph, it is far from conveying information to the learners about how to pronounce correctly, because it's hard for learners to perceive the differences in prosody when their own voices are different from what they hear (Flege, MacKay, and Meador 1999).

We develop our method by studying the most common usecase in CAPT: A learner firstly listens to the gold standard version of a sentence read by a native speaker, then tries to imitate what s/he has heard, and at last is reported how good s/he has spoken, and with which kinds of errors and why. This means that the sentence and also the correct phoneme sequence are known to the system. The system should also know all possible errors that could happen in this sentence, if such information is previously given to or is continuously learned by the system. In our approach, we firstly gather the learners' data which are then annotated by linguists with error types and correction tips. After the analysis of the annotated data, we set up classifiers to distinguish not only correct and incorrect phonemes, but also in which way a phoneme is falsely pronounced. Thus, by applying a model trained with the gold standard plus learners' data, our HMM network produces fine-grained classified results, which contain information for generating corrective feedback. Furthermore, we verify prosody by comparing parameters in the frequency domain, and develop novel feedback using prosody transplantation.

1.1 Major Contributions

We provide a rich set of methods for improving speaking skills in second language learning of pronunciation, including detecting phoneme and prosody errors, and

also providing corrective feedback via graphics and sounds. Our system is designed as a pronunciation training subsystem for commercial CALL applications, which have the option to access large amount of the gold standard speech data. But more importantly is that our system can integrate user speech data and thus improve its performance when more data are generated.

Since L2 can be strongly affected by learners' L1, our system trains separate models for different learner groups, e.g. for English as a second language, German and French native speakers will be offered a different systems, one with acoustic model trained only with German learners' speech data and the other only with French. With a few adaptations, our system can be applied to any learner-target language pairs, including tonal and non-tonal languages.

Besides mispronunciation, the uncertainties in L2 pronunciation, such as pause and extra phoneme caused by hesitation, also bring challenges to verification. Therefore we adapt the popular speech recognition toolkit HTK, modify the silence model to make it more suitable for recognizing pauses and also delivering more precise forced alignment. This will be done in the preparation work phase and applied to each following phase.

1.1.1 Core Methods

Our core algorithms can be divided into two tasks: pronunciation verification and prosody verification. They work separately from each other but share internally the same data format, which extends MaryXML (Schröder and Breuer 2004) Schema and stores all kinds of errors in speech. In runtime, both subsystems are fed with the same input, processed from the learner's speech data, and run in parallel. Hence our system runs in real time despite the huge CPU time cost in performing transplantation in frequency domain.

1.1.1.1 Pronunciation Error Classification

In general, teachers are able to identify pronunciation errors in the learners' speech and can show them how to correct these, but this is only feasible if teachers are available, however, most existing commercial CALL applications classify a phoneme only as either right or wrong, hence they do not provide additional explanations for error corrections.

We divide pronunciation errors into insertion, deletion, substitution and distortion, meaning a phoneme is inserted, removed, replaced with another, or pronounced in the wrong way. For distortion, errors are classified by the tongue, lip and mouth movements of the learners perceived by linguists. Under this type of classification, each phoneme can have different potential errors due to wrong articulation, and for each type of error, information provided in the annotation can be offered for correction. In this way, our approach combines expertise from linguists and modern language technologies successfully and efficiently.

1.1.1.2 Phoneme Error Detection and Feedback

To detect phoneme errors in L2 speech, we utilize the HTK (Hidden Markov Toolkit) to perform phoneme recognition, which is similar to speech recognition, except that the dictionary contains phonemes not words, and the grammar describes how phonemes combine into words, rather than words into sentences. In order to classify distortion effectively, we create different aliases for each type of distortion of a single phoneme, and add them to the dictionary. Grammars are then generated by considering all possible combinations of correct and incorrect phonemes, including inserted, removed, substituted and distorted ones. Since the texts that learners read are already known to CALL systems, we generate grammars on the fly while displaying sentences to the learners to read. One grammar is limited to recognizing only the phoneme sequence of the text from which it is generated. The acoustic model required by the recognition is trained with mixed speech data from the gold standard and the L2 learners, and all the data are forced aligned and error-annotated. The result of the recognition is a sequence of phoneme. By comparing the result sequence to the correct one, we can identify the pronunciation errors from extra, missing or different phonemes. When distortion is found, we retrieve corrective information from the annotations and show the learners how to articulate correctly. To help learners perceive and understand their pronunciation errors more easily, we show not only textual instructions but also audio feedback by playing clips of the learners' own speech data that contain the correct phonemes.

1.1.1.3 Prosody Verification

Our prosody verification methods implement state of the art signal processing and manipulating algorithms to locate prosody differences in accent, duration and pitch, and visualize them to learners. Analysis results are based on correct forced alignment of learners' and gold standard speech data. To achieve this, we train a new acoustic model that takes L2 pronunciation errors into consideration while aligning. As a result, labels are still correctly aligned if they contain pronunciation errors, even when new phoneme is added or existing ones removed. Knowing at what time each phoneme is spoken, i.e. aligned labels, we can compare each prosody aspect separately: stress misplacement can be detected by checking energy distribution; duration difference can be retrieved directly from labels; and pitch differences are calculated via pitch marks of each phoneme per window fragment. Learners' speech data is compared to gold standard of the same gender, and in the comparison, we focus on not the pitch values but the variation tendency, this successfully avoids false positive results when learner's *f0* deviates a lot from gold standard. Differences are visualized via highlighting the corresponding phonemes or words, for pitch differences specifically, pitch curves with time warping are rendered below phoneme lists to provide more direct comparison.

1.1.1.4 Prosody Transplantation as Feedback

In addition to the visualization of the differences in pitch, we provide a new type of feedback: prosody transplantation, which transplants the correct prosody from the gold standard to the learners' voices. This not only enables learners to perceive the differences better, but also provides target utterance templates, which learners can easily imitate. We set threshold parameters based on gender information and forced alignment results to guarantee the transplanted voices does not sound distorted, in case they are affected by pronunciation errors. Learners' feedback from subjective evaluation shows that prosody transplantation contributes a lot in prosody correction.

Prosody verification methods are developed based on modern signal processing technologies and are hence fully language independent (Huang et al. 2001). They apply for both tonal and non-tonal languages, although in tonal language like Chinese, error in pitch (tone) is usually categorized as phoneme error rather

than prosody error (Broselow et al. 1987). But prosody transplantation as perceivable feedback might have critical requirements for some language pairs, and threshold values for error identification in each target language would also need minimal moderation.

1.1.2 Overall System

We elaborately design a system framework to realize our core algorithms, including an annotation tool, which collects L2 speech data by having linguists annotate them efficiently; and a speech verification system, that verifies learners' speech at both phoneme and prosody level.

1.1.2.1 Annotation Tool

MARY Annotation Tool (MAT) (Ai and Charfuelan 2014) is developed in Java, as a component in the open-source MARYTTS project. It is designed specially for L2 speech error annotation. MAT provides an intuitive interface with no dependency to other software or libraries. MAT aligns speech data automatically, so that annotators can play audio clip of each phoneme, syllable and word individually. Moreover, it allows annotators to easily define new error types, add information for error explanation, and correction suggestions. Annotations are validated against predefined annotating principles and stored as extended MaryXML files. MAT also inherits signal processing methods from MARY (Schröder and Trouvain 2003) and is capable of displaying various kinds of processing result as images, such as waveform, spectrogram, pitch contours and energy graph. In our work, we use MAT for annotating phoneme level errors including insertion, deletion, substitution and distortion, but if configured differently, syllable, word and sentence level errors can also be annotated.

1.1.2.2 Online Pronunciation Training System

A revolutionary pronunciation training system is developed by integrating our novel methods and algorithms. Each time a learner reads a sentence, the pronunciation and prosody are verified at the same time, and visual and audio feedback are generated and presented to the learner. In this way, the learner

is not only informed where and how an error occurs, but is also provided with corrective information, which they can easily perceive and follow to correct the error. The system is able to update itself under semi-supervision, and requires only a certain amount of the gold standard and the L2 speech data for bootstrapping. After deployed online, verified speech data from L2 learners will be stored in the same MARYXML format as the annotations. Linguists can open these in the annotation tool and check the verifications and correct false ones. The new annotations are then used for training a more precise acoustic model to perform better verification. A general picture of the system and the interactions between its components are depicted in Figure 1.1.

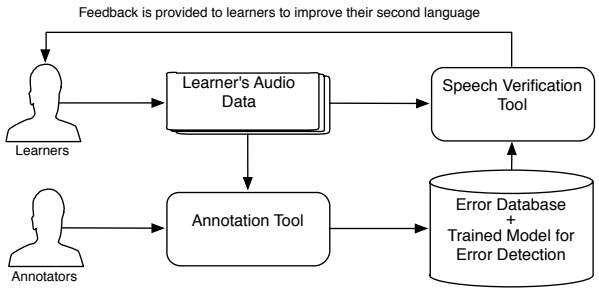


Figure 1.1: Bootstrapping and self-learning architecture.

This thesis makes relevant contributions to the evaluation of perceivable feedback in CAPT systems. One of our experiments shows that learners' own pronunciation can be used to correct their mistakes in substitution and distortion. We also designed another progressive experiment, which proves that learners indeed have problem imitating the prosody in the gold standard utterances, but when provided with perceivable feedback of their own voice and transplanted prosody, learners can replicate native prosody much easier.

1.2 Research Context and Support

The thesis idea reported here was formed in the project SPRINTER (Speech Technology for Interactive, Multimedia Language Course)[1]. It was funded by the German Federal Ministry of Education and Research (BMBF)[2], and car-

[1] http://sprinter.dfki.de
[2] http://www.bmbf.de/

ried into execution by DFKI and LinguaTV[3], an online platform for learning languages, focused on producing videos and games for language learning. The project aimed at integrating modern language technologies into computer assisted language learning platforms, with application areas including: automatic generation, processing and analysis of learning material, interactive learning methods, and personalized and individualized learning. The main research goals of the SPRINTER were:

- Speech verification for enhancing pronunciation of second language learners
- Dialog technologies for a new type of language learning game
- Paraphrasing for generation of language learning material.

The author's main effort in the project was to realize the first goal, with inspiration and support from MARYTTS, which was developed in an earlier project NECA (A Net Environment for Embodied Emotional Conversational Agents), funded by European Union. MaryTTS is an open-source, multilingual Text-to-Speech Synthesis platform written in Java. It is now maintained by the Multimodal Speech Processing Group[4] in the Cluster of Excellence MMCI[5] and DFKI. The schema MaryXML was extended and used as data storage format across different modules in the system. Many MARY components and GUIs were utilized within the SPRINTER project such as training acoustic models and preprocessing speech signals. The prosody transplantation method described in this thesis is also inspired by MARY's prosody manipulation methods.

The fruitful research results in this work were stepwise presented in different conferences, including

- Perceptual feedback in computer assisted pronunciation training: A survey (RANLPStud 2013[6])
- MAT: a tool for l2 pronunciation errors annotation (LREC 2014[7])

[3] www.linguatv.com
[4] http://www.mmci.uni-saarland.de/en/irg/msp
[5] http://www.mmci.uni-saarland.de/
[6] http://lml.bas.bg/ranlp2013/
[7] http://lrec2014.lrec-conf.org/

- Sprinter: Language technologies for interactive and multimedia language learning (LREC 2014)
- A system demonstration of a framework for computer assisted pronunciation training (ACL-IJCNLP 2015[8])
- Automatic Pronunciation Error Detection and Feedback Generation for CALL Applications (HCI International 2015[9]),

where the author discussed with other researchers, received many suggestions and upgraded the system gradually. (Ai (2013); Ai and Charfuelan (2014); Ai et al. (2014); Ai and Xu (2015); Ai (2015))

1.3 Thesis Structure

The remainder of this thesis is organized in five chapters:

- Chapter 2 presents background information on CALL research. It starts by reviewing a brief history of CALL, from which the general accepted principles in the CALL system design are summarized. Afterwards, the relevant methods and technologies applied to CALL are collected. Finally, the current and future work of CALL, which brings together artificial intelligence and natural language processing, is discussed.
- Chapter 3 introduces a special area in CALL: Computer Assisted Pronunciation Training (CAPT), that is also the scope in which our research resides. State of the art CAPT technologies are briefly discussed, including: methods for pronunciation error detection and automatic feedback generation, with focus on perceptual feedback and ways from which they can be created, forced alignment and also MaryTTS.
- Chapter 4 presents the preparatory work, which is done prior to the system deployment. After evaluating existing forced alignment toolkits, we choose the better one and adapt it to fit the special task of aligning L2 speech data. An acoustic model is trained with the mixed gold standard and L2 speech data.

[8] http://acl2015.org/
[9] http://2015.hci.international/

- Chapter 5 describes our novel L2 speech error annotation tool, MAT (MARY Annotation Tool). The advantages of MAT are shown by comparing it to other speech data annotation tools. The schema for storing annotations is also introduced.

- Chapter 6 shows our solution for phoneme level pronunciation error detection. We describe each step deliberately and evaluate the result with human judgment results. Feedback is generated at the same time that errors are detected. We also perform subjective evaluation stepwise to check if perceptual feedback really works.

- Chapter 7 presents our solution for prosodic error detection and feedback. After explaining the importance of prosody in speech, we firstly introduce force alignment as the basic technology. Then we present detailed methods on how differences in word stress, duration and pitch can be discovered. These differences are not only shown visually, but are also calculated and contributed to perceptual feedback generation, in which we adapt the existing FD-PSOLA method to perform prosody transplantation.

- Chapter 8 closes with a conclusion discussing the essential components of our approach. Furthermore, open problems and opportunities for future research is also presented.

Chapter 2

Computer Assisted Language Learning

This chapter describes the global research context in which our work is embedded. We firstly inspect historical developments in CALL to access the technological progress and scientific insights of the last decades. Then we review the generally accepted design principles of CALL applications and collect useful methods and technologies. At last, we introduce language technology and discuss the new future of CALL by integrating language technology and other modern technologies.

2.1 Traditional CALL

2.1.1 A Brief History

Teaching and learning is one of the key aspects of why human beings evolve and recreate the world as it is now, so new methods of teaching and learning always find their connection to modern technologies of the time. Before computers, radio and broadcasting have already been used in educational purposes. And it is not surprising that computer assisted language learning (CALL) has a history almost as long as computer itself. The term *applied linguistics*, which is the main research area that modern CALL belongs to, was raised in the U.S. almost at the same time as the first computer ENIAC (Eckert and Mauchly 1964) was

brought to the world. In the following 1950s, ideas of how language and other knowledges should be taught and learned in the future were shared by different authors, among them are: *linguistics across cultures: applied linguistics for language teachers* (Lado 1957), *the science of learning and the art of teaching* (Skinner 1954), *a new technique of education* (Ramo 1957), and *the problem of programming communication with changing machines: a proposed solution* (Strong et al. 1958). So it was only a matter of time before the computer was used for educational purpose too, and it didn't take too long.

In 1960, the PLATO (Programmed Logic for Automatic Teaching Operations) (Bitzer et al. 1961), which is commonly regarded as the first computer assisted instruction system and also an important milestone in early CALL development (Marty 1981), was deployed on the ILLIAC 1 computer at the university of Illinois. By the end of 1969, PLATO III after several re-designs was already capable of allowing third party lesson modules developed using TUTOR programming language (Sherwood 1974), brought PLATO to a wilder range of usage where both developers and users were involved. This feature also brought the PLATO team more funding that enabled the development of PLATO IV in 1972. Even from today's point of view, it was already a very mature multi-user system since it included functionalities such as instant messaging, chat rooms, forum and message boards. Its multimedia features were more impressing: it could connect to plasma display to render high resolution bitmap, and also to programmable synthesizer to produce complex sounds. Soon after PLATO IV was released, Curtin et al. (1972) developed a program for translating Russian to English, which started the use of PLATO in language teaching. By the end of the seventies PLATO laboratory was delivering over 100,000 student hour courses every semester, and half of them were for language teaching (Hart 1995). A broad range of languages were already covered at that time, including Chinese, English, French, German, Hindi, Latin, Russian, Greek and Swedish (Ahmad 1985), and many outstanding programs were developed, e.g. a Sentence Judger program that can check misspelling and word order in the student's answer (Curtin et al. 1972). Despite the fact that it still ran on mainframe computers, PLATO was regarded as a pioneer in CALL, and according to Levy (1997), "it was the first project to engage language teachers and technical staff in the development of CALL materials in a co-ordinated way."

The other system that shared the same public attention was TICCIT (Time-shared, Interactive, Computer-Controlled Information Television), a cable tele-

vision system developed by MITRE Corporation, which was later refined for computer assisted instruction in cooperation with the University of Texas in Austin and Brigham Young University (Bunderson 1973). Different from PLATO, which allowed developers to design lesson freely, TICCIT has a "learner-centered" framework that limits the design of the lessons, that is: learners could always navigate through the displays and control the content to be displayed, with a special designed keyboard (Sanders 1995). This limit was very prophetic and later allowed TICCIT to be quickly adapted onto personal computers and gained much more users than in community universities originally. Similar to PLATO, the courses for TICCIT are mainly languages too. In fact, both were usually compared and evaluated together (Alderman et al. 1978), and regarded as the landmarks of early CALL, also known as behavioristic CALL (Warschauer and Healey 1998) or restricted CALL (Bax 2003).

Communicative CALL (Warschauer and Healey 1998), which is generally accepted as the second phase, began with the arrival of personal computers in the late seventies. It was also an era in which a variety of computer software boomed. Earlier in this decade, CALL exercises were dominated by those focused on direct training, such as gap-filling, multiple choice and sentences reordering (Jones and Fortescue 1987). Although many of them still exist today in paper materials and examinations, they are gradually abandoned by language teachers since in such exercises, learners only provide responses to the presented prompts, which easily gets boring. CALL however, has gone over this long before. In the eighties, CALL typology was enriched by various new kinds of exercises, among them are games, simulations and adventures. These exercises focused on how to use languages, moreover, they were able to handle flexible input and react correspondingly. In this way, the learners' interest could be held and grammars taught implicitly. *French on the run* was a famous exercises at that time and also one of the first adventures made for language learners. The learner played a role of a solider in World War II who tried to strive a way out of the enemy area. Another example is *Granville*, which simulated the environment of the French town, in which students learned not only French but also everyday life of French people and local information (Cameron 1989). But in the later years, as the processing abilities of computers rose to new levels, these programs were updated and developed into serious games and deviated further away from the purpose of language learning, e.g. *Sim City*.

Then came the 1990s, a booming decade for multimedia and Internet. CALL

nourished again from the top technologies and evolved greatly. Thanks to multimedia support in text, graphic, audio and video, it was for the first time that the four essential skill in language: reading, listening, writing and speaking can be observed at the same level and trained cooperatively. This has significant meaning to CALL because according to Stepp-Greany (2002), the skills are interrelated and interacted, and training them simultaneously helps learners to improve their language in better ways that old CALL applications never could. On the other hand, the appearance of CD-ROM and DVD also allowed portable applications that spread more widely among users, such as *Montevidisco* (Schneider and Bennion 1983). Internet also brought new life to CALL. Since the year 2000, Web 2.0 allowed web applications to behave just like desktop ones, and the larger bandwidth avoided poor audio/video quality and slow reaction time. Besides, Internet provided convenient tools such as social networking and video sharing, that got more people involved in both teaching and learning. Above all, Internet granted learners access to large amount of resources whenever and wherever they wanted, i.e. personalized learning flexible to time and places. CALL of this phase (1990 - present) was summarized by Warschauer (1996) as *Integrated CALL*, applications in this phase tended to utilize all resources and technologies available, for example the method published by Ai et al. (2015) was able to parse any article on the web with enough relationships mentioned into multiple-choice exercises for language learners.

2.1.2 CALL System Design

The design of a CALL system is a large topic because CALL applications have different training focus and can be visualized in various ways. However, there are some common conditions that CALL applications should meet in order to provide learners a helpful environment. Such conditions are firstly concluded by Egbert and Jessup (1996) as follows:

- *Opportunities for learners to interact and negotiate meaning with an authentic audience.* Before the age of CALL, learning is basically interaction between students and others (Kelman 1990). For language learning with a communicative purpose, interaction is more important. Hence the involvement of an active audience is a necessary environment CALL should also provide (Vygotsky 1980).

- *Learners involved in authentic tasks, which promote exposure to and production of varied and creative language.* This means learners should be provided with interesting tasks with a real purpose behind it (Johnson 1991). The "grammar drill" multiple-choice exercises are very common counterexamples, on the contrary, Roen (1989) provided a good example in which tasks were set up for learners and could only be solved with creative language skills.

- *Learners have opportunities to formulate ideas and thoughts and intentional cognition is promoted.* Learners should not only be given chances to formulate ideas and thoughts, but also be engaged to do so. (Salomon 1990)

- *An atmosphere with ideal stress/anxiety level in a learner-centered classroom.* Some pedagogists believe certain feelings of apprehension, as those learners can feel in traditional classroom, are necessary. (Kanters et al. 2009) (Brown 1987)

Egbert et al. (2007) later refined them into eight necessary conditions, namely: interaction, authentic audience, authentic task, exposure and production, time and feedback, intentional cognition, atmosphere, and autonomy. But even under clear guidance, designing CALL applications is still no simple task. Designers need to keep the learners interested in the teaching material and exercises, and give them enough control to choose what they want to learn. This requires cooperation of specialists in different fields, such as linguists, pedagogist, programmer, graphic designer, video technician, sound engineer and content designer (Gimeno-Sanz et al. 2002).

2.1.3 Methods and Technologies

To realize the environment conditions, many methods and technologies were applied.

2.1.3.1 Authoring Systems

Authoring Systems showed the cooperation of programmers and padagogists at a very early stage. They were developed to ease the generation of CALL

course-ware and exercises, in the way that padagoists or linguists were hidden from program complexity and operated directly on the content, mostly in a WYSIWYG (what you see is what you get) way. For example, to generate a gap-filling exercise, the author could directly mark gaps on a given text; likewise, a multiple-choice exercise could be generated via button-clicks and other tools.

Early authoring systems mainly dealt with text only, for example *The Authoring Suite*[1] and *Gapkit*[2], since CALL at that time were text-only too. Other early CALL systems provided authoring languages for a second level programmer to develop exercises and course-ware, such as TUTOR language (Sherwood 1974) in PLATO. As multimedia arrived in the late 80s, it was embraced by authoring systems quickly. Authoring systems in the new era allowed authors to manipulate all available resources including graphics, audio, video and animations, in a much more user-friendly way, such as *Authorware*[3] and *TenCORE*[4]. Starting from the 90s when more and more CALL turned to web applications, authoring systems also included various Internet functionalities, such as managing websites, hence were widely used in creating CALL course-ware. Most of these authoring tools are still popular today, such as *Dreamweaver*[5] and *FrontPage*[6].(Bangs 2008)

Authoring systems are able to produce not only multiple-choice and gap filling exercises, but also other popular "old style" CALL exercises such as crosswords, jumbled sentences and Clozes. However, these systems also have disadvantages. One of the critical issues is that exercises generated by such systems have lack of variation and learners easily get bored. To keep them interested in the exercise, CALL has to do more. Moreover, learners are passive acceptors when they work on the exercises, and this is a far less effective role in learning compared to active creators (Stepp-Greany 2002).

2.1.3.2 Automatic Speech Recognition

In the first decades of CALL it was basically "deaf". The course-ware and exercises provided learning and training for mostly grammar and vocabulary. Al-

[1] http://archive.ecml.at/projects/voll/our_resources/Graz_2001/authentic/wida/menu_wida.htm
[2] http://www.camsoftpartners.co.uk/gapkit.htm
[3] http://www.adobe.com/products/authorware/
[4] http://www.tencore.com/
[5] http://www.adobe.com/products/dreamweaver.html
[6] http://www.microsoftfrontpage.com/

though pronunciation was also taught in the way that accent free audio material was played to learners, there was no interaction, i.e. CALL software couldn't react to the learners' pronunciation and tell them if their pronunciations were correct or not, good or bad. Hence when Automatic Speech Recognition (ASR) was well developed and in the market in the 90s, it really excited CALL developers. It was also around the same time that the concept Computer Assisted Pronunciation Training (CAPT) was first mentioned as a specialized area of CALL (Johnson et al. 1991) (Chun 2013).

Generally, Speech Recognition is a technology that allows computers to analyse audio data of human speech and generate readable text. ASR adds the feature that the recognition is processed on the fly. It is a multileveled pattern recognition task and can be described in the following steps:

1. Human speech is captured by audio input system and translated to analogous electric signals. These signals are then sent to analog-to-digital converter (ADC), which produces filtered and normalized digital data by taking precise measurements of the wave at frequent intervals.

2. The outcome is matched to the smallest segmental units of speech, i.e. phonemes.

3. Phonemes are mapped to words and sentences based on a dictionary and grammar rules defined in the language model. Over time, there has been different specifications to express grammar, such as Java Speech Grammar Format[7] (JSGF) and Speech Recognition Grammar Specification[8] (SRGS). Hidden Markov Model (HMM) and Artificial Neural Network (ANN) are most widely used for language models (Rabiner 1989).

Early ASR systems were almost all speaker-dependent, hence they always required a certain training phase. They were suitable for such tasks as voice controlling, but not for CALL. In the 90s, there were several successful ASR SDKs that allowed developers to build up ASR applications, among them are Fonix, Loquendo, SVOX and Nuance.[9] Applications using these SDKs are speaker-independent but still couldn't meet the requirement of CALL, as they were

[7] http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html
[8] http://www.w3.org/TR/speech-grammar/
[9] Loquendo and SVOX were acquired by Nuance in 2011, and in the same year Fonix changed its name to SpeechFX.

designed for native speakers, whereas CALL was facing second language learners with possible heavy accent. As was evaluated by Derwing et al. (2000), the accuracy of ASR on non-native speech is much lower than on native. Hence several adaptations were made to traditional ASR engines to make it suitable for recognizing non-native speech, such as involving non-speakers in the training process and extending the phoneme set with L2 phonemes (Morton 1999). However, this brought ASR to a dilemma for its usage in CALL: if learners' speech with strong accent or even wrong pronunciation were recognize accurately, it will be hard to detect these accents and pronunciation errors, which means that although ASR finishes its recognition job successfully, it can't help learners with improving their pronunciation.

Another approach to achieving high recognition accuracy in CALL system is via task design. Traditional ASR were developed for dictations, which means they were able to recognize free spoken text, but this also requires complex processing based on large training data because the result could contain thousand of words and their combinations, hence usually done on the server side. In CALL, the requirement of recognition could be narrowed down to a single sentence via careful design. This could significantly raise the recognition accuracy to an acceptable level, even when processing speech input with an accent (Witt and Young 2000).

Early usage of ASR in CALL was to simply pass the learner's speech to standard ASR engines and check the recognition result. Ideally, most words in the results should be identical to what the learner has read, but some were recognized differently. In this case, the system would tell the learner something like "*You sound like <recognition result>, please try again.*", or better: "*Please pay attention to the word <wrong word>*", where <wrong word> can be retrieved by comparing the recognition result and the original sentence. Although this method couldn't provide feedback on how to pronounce correctly (other than replaying the gold standard), it did point out where pronunciation errors occured. However, a more usual case is that when one word was recognized as being wrong, the words behind it were also recognized as wrong due to the affection of the first wrong word, and in the worst case, half the sentence was harmed. In this situation it would be hard to tell if some words were indeed mispronounced or affected by other wrong ones. Hence the performance of this backward error tracing was very poor. Implementation of this method could only be found in

the early CAPT systems, as in the first version of *Tell Me More*[10].

The other method was realized with the scores generated by ASR systems as a judgment of the learners' speech. A score is calculated per recognition result with a complex algorithm to represent the plausibility of the recognition. For ASR engines with n-best ability, a recognition result with a higher score means it's more likely which speaker has just spoken. While in CALL programs, scores are usually used as an indication of how similarly a learner has spoken in comparison to the gold standard. In this way, ASR engine was no longer in dictation mode, which would try to recognize any speech input, but was limited to recognizing a single sentence. This means that no matter what and how the learner has read, the recognition result would always be the same, however, with different scores for each word. A lower score for a certain word would indicate a possible pronunciation error.

Although scores were more credible and straightforward than comparing words between recognition result and text to read, they were still regarded as vague feedback. The scores might differ a lot even if the learner tried to read the same sentence twice in an identical way. Furthermore, there was nothing more to dig behind the score, and a score couldn't tell if a phoneme was pronounced wrong, or a silent letter was pronounced. The only information it provided was: something was wrong with this word. But this only information was not true all the time. Mackey and Choi (1998) and Wildner (2013) have pointed out strange scores during their evaluation, for example the scores of native speakers were sometimes lower than non-native learners. Miura (2004) compared the automatic scores and the normalized teachers' rating, and found that they didn't agree in many words. Due to this evidence it was suggested that a more heuristic scoring algorithm should be proposed. It should be trained using a L2 pronunciation corpus and reflect the analysis result of pronunciation verification, not of the speech recognition. For example, Franco et al. (2000) combined different types of scores such as segment duration scores and timing scores into an overall score, which could predict the pronunciation quality and this was consistent with human grading.

As a conclusion, ASR was a technology that once was used commonly among CAPT applications. It could locate the pronunciation errors in a predictive way and provide limited feedback. Methods and adaptations have been tried to

[10]http://tellmemore.com/home, was acquired by Rosetta Stone in 2013.

raise the accuracy of error detection but the results weren't promising. Hence nowadays ASR has lost its popularity compared to the beginning of the century.

2.1.3.3 Virtual World

Virtual world evolved from early simulation and adventure games in the 70s, such as *Granville* and *Colossal Cave Adventure*. Back at that time, they were simply text-only applications, which accepted command inputs from the keyboard and reacted correspondingly. Today, virtual world has benefited from Internet and multimedia technologies and provide users with experiences that no one in the 70s could ever imagine: language learners find themselves in a 3D virtual environment of a special time period, in which they learn languages by controlling their avatars to perform tasks or communicating with others, such as in *Immerse Learning*[11] and *Avatar Languages*[12]. Many popular virtual world games also integrated CALL to provide users with a learning environment, for example, one of the most common education-based activity in *Second Life*[13] was language learning, and over 200 universities and academic institutions were involved (Cooke-Plagwitz 2013).

A generally accepted pedagogical outline for virtual world was proposed by Bronack et al. (2008), in which they summarized the concepts as follows:

- Learning occurs through participation in a Community of Practice;
- Knowledge is socially constructed and learning is social in nature in a Community of Practice;
- Learners proceed through stages of development from Novice to Expert under the guidance of more experienced and knowledgeable mentors and among like-minded peers in the Community of Practice;
- An identifiable knowledge base that is both general in nature and also specific to specialties emerges from focused activity within the Community of Practice;

[11] https://immerselearning.com/
[12] http://www.avatarlanguages.com/
[13] http://secondlife.com/

- All professional educators develop a set of Dispositions reflecting attitudes, beliefs, and values common to the Community of Practice.

Enlightened by these guidelines, virtual world language learning applications applied a general pattern called task-based language learning. The core of this pattern was to provide virtual tasks analog to real life ones for learners to practice the language they had learned. The tasks covered almost all kinds of real life behaviors, such as bargaining with a seller in a store, making an appointment with a doctor, or even flirting with someone in a bar. In this way, learners not only put what they learned directly into usage, but also continued learning while exercising. In these applications, the main challenge is to create tasks. A proper task should on one hand have a realistic meaning in which a real life problem needs to be solved, and on the other hand, contain certain "gaps" that learners are required to solve, e.g. information gap needs exchange of information with NPC or other learners, while opinion gap requires learners to digest the information and formulate their own ideas (Ellis 2003). A task needs to be elaborately designed so that learners are not distracted from the task, e.g. focusing onto linguistic content such as unknown words or grammar structure. And during the task, the teacher or instructor should only play the role of observer and give the learners full control, any instruction and feedback should be given pre and afterwards. This also conforms the "learner centered policy" that has been applied to CALL since the early stage.

Another concept that shares a similar idea is a *Virtual Learning Environment* (VLE), which is also a web-based multimedia platform. The main goal of a VLE is to create a virtual learning environment analog to real life ones, in which learners and teachers interact with each other. Different to virtual world, VLE focuses on virtual courses instead of tasks, or in another way, courses are the only tasks that learners are asked to participate in a VLE. Hence almost all major components in a VLE are related to courses, such as administrative information (e.g. prerequisites) of all available courses, a notice board of ongoing courses, and communication media within and outside courses, e.g. forum, chat room and video conference. Similar to real courses, learners are also asked to take part in exams after attending courses, and automatically generated scores serve as feedback of how good learners have conquered the courses. Moreover, a VLE is also used as an Information System, since it provides large amount of resources related to courses, which can be accessed without constraints of time

and location.

A VLE also follows a "learner centered" pattern. Learners are encouraged to make their own timetable of selected courses. They could take their time to do the exercises and review previous courses if needed. Exams could also be taken whenever learners are well prepared.

In a word, virtual world and virtual environments opened a new window for computer assisted language learning by utilizing the most modern technology in video, graphics and computer mediated communication.

2.2 ICALL - Next Generation CALL

Telecommunication and computer technologies evolved fast in the past decades. Users' experiences were enriched while interacting with CALL software and other learners, while other technologies, such as authoring systems, eased the way in which teachers and programmers developed and deployed CALL programs. However, there is one technology closely related to CALL but hasn't gained full attention among developers and users – language technology. By starting to adapt and utilize this technique, a new era has arrived: Intelligent CALL (ICALL).

2.2.1 Overview of Language Technology

Language is one of the oldest information medium and communication system humans ever developed and used. Since early hominins started to share intentionality millions of years ago, languages have evolved into different language families, and it was estimated that the number of languages in the world today was up to 7000 (Hauser et al. 2002). In most of the time in human history, languages were only "spoken". The first written language appeared after it was spoken for more than ten thousand years (Zimerle 2010). Written languages also evolved with time, for example, in less than 3500 years, Chinese written language has evolved from oracle bone script into today's simplified [14]. Besides, population shift and national amalgamation also played important roles in language formation. With text, human knowledge was able to be maintained, even

[14] http://www.omniglot.com/chinese/evolution.htm

when the language in which the text was written, was no longer spoken.

Different technologies, including algorithms, computer programs and electric devices have been developed to analyze speech and text, or even to produce and modify them efficiently. Over the years, these technologies evolved and became more specialized in solving corresponding issues, as listed in Table 2.1 and 2.2.

Examples of Speech Technologies	
Speech Recognition	Transform speech into human readable text. This technology is realized differently between tasks. Recognition of lists of commands or small sets of sentences can be done by pattern matching; while in order to recognize whatever is spoken (dictation), large amount of corpus and training is required.
Speaker Identification	Identify the speaker from a set of known speakers, or verify if the speaker is known to the system. A typical use case is in the authentication part of the security process. Also known as voice recognition.
Speech Synthesis	The process to produce human speech, from either readable text, or other representation of words and sentences, for example phonetic transcription. Speech synthesis is widely used in today's computer software and electrical devices, as both stand-alone product and accessory functionalities.
Speech Coding	Compress speech data while maintaining important features such as emotion, intonation and identity of speakers. Speech coding provides fundamental support for other speech technologies and boosts their efficiency.

Table 2.1: Common Speech Technologies

However, language technologies are not simple summarization of speech and text technologies. On one side, there was never a clear boundary between the roles that spoken and written languages played. Traditionally, people use speech to communicate and text to record. But text has also been used in communication for thousands of years between people in distance that can't be reached via speech, e.g. mails or note tied onto pigeons' legs, whereas speech were able to be recorded and served as maintenance and transmission of knowledge too. As a result, technologies earlier applied to text are now found in speech, such as audio indexing and retrieval (Makhoul et al. 2000), and vice versa. On the other side, more technologies need to be involved when processing language. For example, We attach facial expression to convey our feelings as we speak. Writing styles also reflect writers' emotion. To analyze such features, technologies like

Examples of Text Technologies	
Text Categorization	Analyze texts and categorize them. Filtering is a special case of text categorization when only two categories are involved.
Text Summarization	Generate a much shorter version of given texts that summarize the main ideas. Different summarizations are generated when requirements like length and writing style vary.
Text Indexing and Retrieval	Store texts in indexed database for efficient retrieval. This technique is widely used for full text search among all search engines. Performance is enhanced when combining with text categorization and text summarization.
Information Extraction	Extract relevant information pieces, such as topics, named entities and relations between entities, from given texts.
Text Mining	Analyze extracted information pieces from coherent sources, in order to formulate conclusion or discover new information.
Question Answering	Automatically answer questions in natural language. Answers are created by querying knowledge bases, which could be either a structured database or an unstructured collection of texts.
Translation Technologies	Translate texts between languages automatically (machine translation) or assist humans while translating (computer-aided translation). Depending on the closeness between source and target language, different methods can be applied.

Table 2.2: Common Text Technologies

pattern recognition need to be applied. For disabled people, languages are not only spoken and written, but also signed and brailled. Assistive technologies are required to handle the communication and knowledge access of deaf and blind people. Furthermore, modern multimedia technologies have mixed language with more elements like pictures and videos, and wrapped them together with speech and text as a whole, thus speech and text technologies overlapping with each other are employed together with other technologies to handle multimodal communication and multimedia documents (Uszkoreit 2002).

Different methods and algorithms are employed to realize language technologies, especially non-discrete mathematical methods, e.g. statistical techniques

and neural networks. Moreover, linguistic knowledge and formalisms are also utilized, among them are dictionaries, morphological and syntactic grammars, and rules for semantic interpretation. For training statistical models and testing purposes, large amount of corpus are also collected or produced, in both speech and text form.

It's not surprising that many applications and tools with human language knowledge already existed and changed our lives. We use automatic correction, completion and grammar checking tools when we type. We also subscribe channels in news applications to read only what we are interested in. Modern operating systems are equipped with speech input to support the operations. And digital intelligent personal assists, which are pre-installed on many mobile phones, could answer to the users' query and provide information searched from the Internet, or react to the users' commands and generate proper feedback. But language technologies can do better. With more advanced natural language understanding technology and multimodal technologies including facial expression and gesture simulation, computers are expected to become truly communicative partners, from which we can access global knowledge using natural interaction. Apart from this, language technology applications also help humans communicate with each other, particularly between people with different mother tongues. Although automatic translation of unrestricted texts still has a long way to reach acceptable accuracy, prepared translation is enough to deal with daily communications when traveling aboard. For example, mobile applications YoChina (Xu et al. 2014) has lists of useful phrases and sentences, in which the syntactics are fixed but the information are empty (e.g. *I'm _ years old*). By translating, the information, that the users provide, are applied to the target language templates stored in the database. In this way, translations are guaranteed correct in both syntactic and content (Uszkoreit 2002).

In a word, language technologies have changed our lives and will keep changing them, since the ultimate goal of language technologies is barrier free communication among all human beings and machines.

2.2.2 Language Technology in CALL

To deal with the increasing requirement on performance and usability of CALL applications, EUROCALL (European Association for Computer-Assisted Lan-

guage Learning) [15] has opened special interest group on Intelligent CALL (ICALL) and Nature Language Processing CALL (NLPCALL). Both groups work together to discuss and share innovative research, practice and development in integrating computational linguistics and artificial intelligence technologies and methodologies into computer-aided language learning. Before this, some language technologies had already been used in CALL, e.g. speech recognition, but they served mainly as toolkits in modules and never affected the workflow of the whole framework: the system always behaved the way it was programmed to. But this is about to change, as the next generation CALL applications will provide the following features:

- Capable of producing or selecting customized learning material for given tasks.
- Able to determine learners' levels of skill and knowledge, manage their learning progress, and generate appropriate tasks for next levels. (Beinborn et al. 2012)
- The ability to interact with learners through conversational agents which simulate the linguistic facility of human interlocutors, in the way that agents analyze learners behavior and models of their proficiency, and respond accordingly.

But most importantly among all, by applying nature language processing in CALL, applications no longer treat languages as a list of tokens but can understand language at a deeper level. In *ROBO-SENSEI: Personal Japanese Tutor* (Nagata 2009), learners were asked to compose texts in nature language, e.g. answers to certain questions or description of images. To detect L2 errors in writing, the texts were processed with a pipeline containing several NLP components, namely: lexicon checker, morphological generator, word segmentor, morphological parser and syntactic parser. Errors detected were categorized and summarized in Table 2.3. Depending on error type, corrective feedback is also generated automatically by filling information of wrong words and error types into corresponding template, e.g. "The word 'rise' is not in the causative form. Change it to causative form. Try it again!". The main advantage of using NLP in *ROBO-SENSEI* is that only a simple answer schema is needed per exercise,

[15] http://www.eurocall-languages.org/

instead of a long list of possible answers. Relevant correct answers are generated in runtime, and parsed with the learners' answers together to find out any different grammatical natures between them.

Error Type	Description
Unknown Word	A word (or its morphological form) can't be found in the lexicon.
Missing Word	A word is missing to form the correct grammar, typically auxiliary verbs.
Unexpected Word	A word is in the lexicon but not in the correct answer schema. E.g. using "Mum" in an official situation.
Predicate Form Error	Including tense, negation, style and all auxiliary forms such as causative, passive etc.
Modifier Error	Wrong or inappropriate modifier before a noun is used, checked with a syntactic parser.
Word Order Error	Words in a sentence are in the wrong order, e.g. the position of a verb in the relative clause.

Table 2.3: L2 Writing Error Types Detected by ROBO-SENSEI.

Apart from enabling application types, which are previously impossible or less feasible, language technologies also enhance traditional applications, and deliver more robustness and usability. "Jumbled order exercises" is a very famous exercise in which the learner should formulate a grammatically correct sentence using a given set of words. The task could be difficult if the learner does not recognize certain words or has no idea about the meaning of the target sentence. Such exercises usually have defined answers, and in most cases only one correct sentence per exercise, even if there might be several possible solutions, which could frustrate learners. To solve this, Ai et al. (2014) adopted a "parse and generate" approach that is based on dependency analysis and subsequent generation of sentence variants as different linearizations of the dependency structures. Dependency structures provide a functional representation of a sentence, in general without implying a specific word order, in contrast to phrase structure representations. For the dependency analysis they employed DFKI dependency parsers trained on dependency treebanks (Volokh and Neumann 2012). The resulting dependency tree is fed into a generator (ZHANG 2012) that produces possible linearizations for the dependency tree as sentence variants. To reduce over-generation of ungrammatical variants, in addition, a deep parser based on HPSG grammars is used for grammatical verification of the generated sentences. The advantage of this approach is that the number of generated variants can be controlled by various parameters, such as beam size and probability thresholds.

Furthermore, user feedback can be collected through exercise manager and used to improve and adapt the models via re-training.

A handful of CALL applications with NLP ability are already on the market, as listed in Table 2.4, and more are under research and development, as artificial intelligence and language technology grow more mature, the gap between research prototypes and commercial products could be even smaller, and we can expect ICALL applications powerful enough to replace tradition classroom language teaching and learning.

2.3 Conclusion

This chapter attempted to give a concise overview of CALL. From the 1950s till the late 20th century, research and development of CALL has undergone huge changes. Early CALL projects tried to find out the best way for learners to digest knowledge and practice exercises, hence a large amount of applications were created, and at the same time, a lot of authoring systems were developed to allow language teachers to build applications with little or no programming knowledge. In the 80s, more thoughts of pedagogy and heuristic were given to CALL applications and in the mean time, newest technologies at that time were applied to CALL immediately, e.g. speech recognition, with the hope to make applications not only judge right or wrong answers but also recognize the learners' errors and provide feedback. But later researchers and developers realized that it was impossible to deal with various kinds of L2 errors, especially in pronouncing. Starting from the 90s, as the technologies of the Internet and multimedia boomed, new elements were also added to CALL and integrated training of reading, speaking, writing and listening was possible. Web applications brought new customers to CALL, which also promoted the research in customization and personalization in CALL programs. In the 21st century, language technology and artificial intelligence were applied in CALL to pinpoint errors and provide comprehensive feedback. With the improvements in these and other modern technologies, a new future in CALL can be expected.

System	**Language**	**Technologies**	**Skills**	Reference
RECALL	English	NLP, ES	Writing	Krüger and Hamilton (1997)
Web.Pass.Voice	English	NLP	Writing	Virvou and Tsiriga (2001)
ISLE	English	ASR	Speaking	Menzel et al. (2001)
L2tutor	English	NLP	Reading, writing, speaking	Price et al. (1999)
ILTS	English	NLP, MT	Writing	Tokuda and Chen (2001)
Subarashii	Japanese	ASR	Listening, speaking	Bernstein et al. (1999)
FreeText	French	NLP	Writing	Vandeventer (2001)
Herr Kommisar	German	NLP, NLG	Reading, writing, speaking	DeSmedt (1995)
The Spanish Verb	Spanish	NLG, ES	Writing	Soria (1997)
TLS/CATL	Thai	NLP	Writing	Dansuwan et al. (2001)
MILT	English, Spanish, Arabic	NLP, ASR	Reading, writing, speaking	Holland et al. (1999)
Athena	Language independent	NLP, NLG, ASR	Reading, writing, speaking	Murray (1995)

Table 2.4: Examples of ICALL apps applied for different languages. Multiple artificial intelligence and language technologies are used, including: ASR: automatic speech recognition; NLG: natural language generation; NLP: natural language processing; MT: machine translation; ES: expert system.

Chapter 3

State of the Art

In this chapter, we narrow our research scope down to a specific topic in CALL: Computer Assisted Pronunciation Training (CAPT). We will firstly present a brief introduction and then review the related work that has provided us with relevant and inspiring methods and experiment results. Since we reuse and further develop some of MARY components, we will also briefly introduce MARY at the end and focus on its text processing components related to our work.

3.1 Computer Assisted Pronunciation Training

Compared to reading, writing and listening, speaking is regarded as the most difficult 2nd language skill, not only because pronunciation could be strongly affected by the learners' mother tongue, but also due to the enormous manpower required in training, including supervised exercise and interaction with native speakers (Lenneberg et al. 1967). Hence the advantage of CAPT is obvious: it allows learners to access theoretically unlimited resources as the gold standard to follow, i.e. there are large amounts of language resources on the web, and at the same time machines will not be tiring to guide learners in practicing.

Because of the requirement on applying different methods and technologies, especially audio signal processing, CAPT has been technologically a separate research and development area in CALL, although the general pedagogical concepts and design principles still work. In fact, pronunciation training was treated slightly differently than in other areas in language teaching. There were miscon-

ceptions about the effect of pronunciation teaching and learning in earlier times, because adults' phonetic-phonological systems were adapted to their mother tongue and could hamper the pronunciation of another language (Flege 1995), and the effect increases with age (Kuhl et al. 1992), and there were even arguments that accent-free pronunciation for 2nd language learners was almost impossible (Hill 1970) and teaching L2 pronunciation was counterproductive (Kresan 1981). This led to less interest in research, few authoritative studies and insufficient pedagogical guidelines in this area. As a result, pronunciation training systems could not be found in early CALL applications, as either commercial system or research project, whereas at the same time a large amount of applications that trained other language skills, e.g. grammar and vocabulary already existed. It was not until the 80s when speech recognition technology was available, that CAPT gained the attention of researchers and went commercial, although as discussed in the previous chapter, ASR was far from being able to pinpointing pronunciation errors.

Inherited from the design principles of CALL, CAPT developers also generally followed some rules, which were learned from pedagogical experience, as summarized below:

- Learners should be supplied with a large amount of L2 speech data from different native speakers. In this way, learners can perceive variations of phonemes and categorize them correctly (Lively et al. 1993).
- Learners should be given the opportunities and encouraged to practice (Kendrick 1997) with teachers. Only in this way they can compare their pronunciation with the gold standard (Swain 1985).
- Learners should be provided with feedback, which not only pinpoint errors in the learners' utterances but also include corrective information that can be perceived (Flege 1995).
- Learners should not be over-corrected. Different from written language which has clear boundary between right and wrong, one can speak with a strong accent but still be understood. The goal of pronunciation training should be communicable, and then accent free. Learners' engagement should not be discouraged by always telling them to repeat the same pronunciation just because they don't sound native (Derwing and Munro 1997).

Among all these design principles, the most challenging one is to identify pronunciation error and generating corresponding feedback. The intelligibility of an utterance can be damaged by not only segmental errors such as phoneme substitution and deletion, but also suprasegmental factors including stress, duration and pitch. Furthermore, intonation serves as "the glue that holds a message together" (Brown 1991) and also affects intelligibility.

In the following sections, we present and discuss existing methods and technologies that detect pronunciation errors and provides feedback automatically. But first, we introduce the fundamental one that make all other technologies possible: forced alignment.

3.2 Forced Alignment

Generally speaking, forced alignment is the process of determining the duration and time of occurrence for each word and phoneme in a speech signal based on its text transcription. Since forced alignment is essential to speech synthesis and speech recognition, it is implemented in almost all such systems and toolkits, such as HTK[1], Kaldi[2], Julius[3] and CMU Sphinx[4] and Festvox[5]. Apart from these, there were also standalone forced alignment tools, as developed by Prahallad et al. (2006), Rosenfelder et al. (2011) and Gorman et al. (2011).

Two phases are involved in forced alignment: training and recognition. The training phase can be generally divided into the following steps:

1. Generating phoneme sequences from text transcriptions with phonemization tools.
2. Iterating all phoneme sequences and producing a set which contains all appeared phonemes in the sequences.
3. Extracting cepstral coefficients of the audio files. Different coefficients are extracted depending on the systems and tools, including Mel Fre-

[1] http://htk.eng.cam.ac.uk/
[2] http://kaldi-asr.org/
[3] http://julius.osdn.jp/
[4] http://cmusphinx.sourceforge.net/
[5] http://festvox.org/

quency Cepstral Coefficient (MFCC), Linear Prediction Cepstral Coefficient (LPCC) and Bark frequency Cepstral coefficient (BFCC).

4. Calculating the deltas and optionally delta-delta features of the coefficients and performing scaling on the feature vectors.

5. Model training. Most forced alignment methods train Hidden Markov Model (HMM) with Baum-Welch algorithm (Welch 2003) and/or Viterbi Path Counting algorithm (Davis and Lovell 2004). Several systems use hybrid models of HMM and ANN (Artificial Neural Network) to boost alignment accuracy.

The recognition phase is roughly equal to HMM decoding, meaning to find out the most likely state sequence that produced the observations, i.e. the acoustic vectors extracted from to-be-aligned audio files using the same methods as in training. Most forced alignment systems implement the Viterbi Algorithm (Viterbi 1967) for decoding, and generate time stamps from the results. For example, HTK would generate the result shown in Figure 3.1 for the sentence "He left his number for you.", in which the first two columns shows the start and end of each phoneme (unit: $0.1\mu s$), the third column shows the recognized phonemes, in HTK default phone set derived from CMU phone set[6], and the last phoneme represents the log likelihood scores of each phoneme. Silences beyond sentence boundary and pauses in between are marked as "sil" and "sp".

3.3 Speech Error Annotation

Error-annotated L2 speech data has been an important type of resource for training automatic error detection and also for testing and evaluating. However, the acquisition of such data is difficult due to the annotation work, which has to be manually carried out by linguists or phoneticians.

Annotation requires several steps of preparation, like text collection and preprocessing, sentence selection according to the desired criteria, and recording by appropriate speakers. Beyond these, the most significant step is to align the recording with a phonetic transcription of the speech. In earlier times when phonetic models were formulated based on limited data, alignment could still

[6]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

```
#!MLF!#
"/path/to/file.mlf"
0 200000 SIL -373.604706
200000 300000 SIL -179.028198
300000 500000 HH -379.245514
500000 1100000 IY -1138.754761
1100000 1100000 sp -0.138478
1100000 2600000 L -2487.238770
2600000 3700000 AE -1763.409180
3700000 3850000 F -296.773773
3850000 4550000 T -1240.916016
4550000 4550000 SP -0.138478
4550000 4800000 HH -475.973877
4800000 5400000 IH -974.963135
5400000 6400000 S -1711.515259
6400000 6900000 SP -815.157898
6900000 7050000 N -315.243958
7050000 7900000 AH -1270.570679
7900000 8400000 M -773.567688
8400000 9000000 B -1006.816589
9000000 9700000 ER -1070.720703
9700000 11100000 SP -2385.134277
11100000 11250000 F -265.891968
11250000 11900000 AA -1056.531494
11900000 11900000 SP -0.138478
11900000 12050000 Y -290.408295
12050000 14650000 UH -3960.716064
14650000 18400000 SIL -6024.947754
18400000 19150000 SIL -1122.620728
.
```

Figure 3.1: Example of forced alignment results from HTK, for the sentence "He left his number for you.".

be accomplished by phoneticians, although very laboriously, with software like Transcriber (Barras et al. 2001) or Wavesurfer (Sjölander and Beskow 2000). But nowadays, the footprint of recordings has grown from megabytes to gigabytes, with duration from a few minutes to several hours, which makes the transcription work unrealistic for humans. In recent years, forced-alignment has been introduced to adopt this job, in the way that it produces automatic aligned speech data with transcription. Forced-alignment is implemented in many widely distributed toolkits such as HTK (Young and Young 1993) and ehmm, which is part of the Festvox framework (Anumanchipalli et al. 2011).

It's not surprising that many next generation annotation tools have been developed with the ability to align speech data automatically by utilizing forced-alignment middleware. With these tools, phoneticians and linguists can work on

well aligned speech data and produce annotation in the desired format. Popular tools among them are SPPAS (Bigi and Hirst 2012), EasyAlign (Goldman 2011) and Train&Align (Brognaux et al. 2012a). Their features are summarized in Table 3.1:

Name	Features
SPPAS	• Python based standalone program. • GNU public license. • Language independent algorithms for tokenization, phonemization and alignment. • Supports Linux, MacOS and Windows. • Generates TextGrid files which can cooperate with Praat.
EasyAlign	• Supports multi-tier annotation in word, syllable and phoneme tiers. • Works as a Praat[7] plug-in, hence very user-friendly for phoneticians and linguists. • Available in French, Spanish, Portuguese, English and Mandarin. • Supports Windows system only. • Relies on HTK, which is under GPL-incompatible license.
Train&Align	• Acoustic models are trained from the corpus to be aligned. • Web interface accessible from all platforms. • Configurable training parameters. • Compatible with Praat formats.

Table 3.1: Features and highlights of popular annotation tools

3.4 Pronunciation Error Detection

In the early years of Computer Assisted Pronunciation Training (CAPT), validation of learners' pronunciation was mostly done with scores from Automatic Speech Recognition (ASR), such as in the work of Hamada and NAKATSU (1993) and Hiller et al. (1994). The calculation of score is based on a set of speech attributes that can be easily retrieved, e.g. articulation rate, segment duration and speech rate (Cucchiarini et al. 2000). The best known example among these scoring methods is Goodness of Pronunciation (GOP), developed by Witt and Young (1997). Although it was later improved (Witt and Young 2000) and adapted (Kanters et al. 2009) (Neri et al. 2006), and also integrated into CAPT systems (Mak et al. 2003) (Luo et al. 2008), this method provides only quantitative measurement of how good learners' pronunciations are, i.e. the difference between a learner and a native speaker. The refined GOP method could detect possible pronunciation error at the phoneme level, but cannot specify what kind of errors the learners have made, nor tell them how to correct these errors.

Many approaches have tried to solve this. Stouten and Martens (2006) mapped the MFCCs to phonological features, performed a segmentation and labeling of the utterance on the basis of these features and computed phonological goodness scores to characterize the pronunciation proficiency of speakers. Although the system mainly focused on generating scores matching human judgment, it also considered different types of errors such as insertion and substitution. Others try to handle the phonemes with low score and apply different classifiers for error determination. Tsubota et al. (2002) introduced pair-wise verifiers that can distinguish confusing pair of phonemes. They generate discriminative features using linear discriminant analysis (LDA) of segments from each pair of phonemes. In this way the previously predicted pronunciation errors from speech recognition can be finely classified. Ito et al. (2007) used a decision-tree-based algorithm in their method. According to their experiment, the accuracy of pronunciation error detection was improved significantly by applying different threshold for each cluster. Truong et al. (2005) adopted LDA and decision tree in their study, and set heuristic threshold on ROR (Rate Of Rise) values to classify phoneme pairs (Weigelt et al. 1990). Specific classifiers for eastern language with tone were also developed, for example, Jo et al. (1998) set up classifiers for articulatory features like manner and place of articulation to identify mis-

pronunciation in Japanese. Wei et al. (2009) proposed a method for detecting mispronunciation in Chinese. They employed the log-likelihood ratios between all the acoustic models and the model corresponding to the given text as features and used Support Vector Machine (SVM) as the classifier. Further more, they used pronunciation space models to enhance the discriminative capability for pronunciation variations.

3.5 Automatic Feedback Generation

Feedback nowadays has been playing a much more significant role than simply telling the learner "You have done right!" or "This doesn't sound good enough". Thanks to the newer technologies in signal processing, it can pinpoint specific errors and even provide corrective information (Crompton and Rodrigues 2001). One of the earliest types of feedback, which is still used in modern CAPT systems like *tellmemore*[8], is to show the waveform of both the L1 (the teacher's or native's) speech and the L2 learner's one. Although the difference of the two curves can be perceived via comparison, the learner is still left with the question why they are different and what he should do to make his own curve similar to the native one. He might then try randomly many times to produce the right pronunciation, which may lead to reinforcing bad habits and result in fossilisation (Eskenazi 1999). To solve this, forced alignment was introduced. It allowed to pinpoint the wrong phoneme, and give suggestion to increase or decrease the pitch or energy, like in *EyeSpeak*[9], or mark the wrong pronounced phoneme to notify the learner, like in *FonixTalk SDK*[10].

Another common type of feedback among CAPT systems is to provide a score. A score of the overall comprehensibility of the learner's utterance is usually acquired via Automatic Speech Recognition (ASR), like in SpeechRater Engine (Zechner et al. 2007), which is part of TOFEL (Test of English as a Foreign Language) since 2006. Many CAPT systems also provide word-level or even phoneme-level scoring, like in *speexx*[11]. Although scoring is appreciated among language students due to the immediate information on the quality it provides (Atwell et al. 1999) , it is regarded merely as an overall feedback, because if no

[8] http://www.tellmemore.com
[9] http://www.eyespeakenglish.com
[10] http://www.speechfxinc.com
[11] http://speexx.com/en/

detail follows, the number itself will not show any information for the learner to improve his speech.

To provide more pedagogical and intuitive feedback, the situation of classroom teaching is considered. Imagine if a student makes a wrong pronunciation, the teacher would then show him how exactly the phoneme is pronounced, maybe by slowing down the action of the mouth while pronouncing or pointing out how the tongue should be placed (Morley 1991). After investigating such behaviours, Engwall et al. (2006) presented different levels of feedback implemented in the ARTUR (the ARticulaton TUtoR) pronunciation training system. With the help of a camera and knowledge of the relation between facial and vocal tract movements, the system can provide feedback on which part of the human vocal system did not move in the correct way to produce the correct sound, the tongue, the teeth or the palate, and shows in 3D animations how to pronounce the right way.

These types of feedback are known as visual feedback and automatic diagnoses (Bonneau and Colotte 2011) that show information with a graphical user interface. Besides these, perceptual feedback, which is provided via speech and/or speech manipulations, is also used more and more commonly in modern CAPT systems.

Simple playback of the native and the learner's speech and leaving the work of comparing them to the learners will not help them to perceive the difference between the sound they produced and the correct targets sound because of their L1 influence (Flege 1995), hence, the importance of producing perceivable feedback has been increasingly realized by CAPT system vendors and many ways of enhancing learners' perception have been tried.

3.5.1 Speech Synthesis for Corrective Feedback

Meng et al. (2010) implemented a perturbation model that resynthesizes the speech to convey focus. They modified the energy, max and min f0 and the duration of the focused speech, and then use STRAIGHT (Kawahara 2006), a speech signal process tool, for the re-synthesizing. This perturbation model was extended later to provide emphasis (Meng et al. 2012). A two-pass decision tree was constructed to cluster acoustic variations between emphatic and neutral

speech. The questions for decision tree construction were designed according to word, syllable and phone layers. Finally, Support vector machines (SVMs) were used to predict acoustic variations for all the leaves of the main tree (at word and syllable layers) and the sub-trees (at phone layer). In such a way, the learner's attention can be drawn onto the emphasized segments so that they can perceive the feedback in the right way.

The study of De-La-Rosa et al. (2010) shows that English text-to-speech may be good enough for providing feedback. A similar study for French language was presented in (Handley 2009), where four French TTS systems are evaluated to be used within CALL applications. In these last two cases speech synthesis is used more as a complement to reinforce the learning process, that is, in most of the cases as a way of listen and repeat, without further emphasis.

3.5.2 Emphasis and Exaggeration

Yoram and Hirose (1996) presented a feedback in their system which produces exaggerated speech to emphasis the problematic part in the learner's utterance, as a trial to imitate human teachers, e.g. if the learner placed a stress on the wrong syllable in a word, the teacher would use a more extreme pitch value, higher energy and slower speech rate at the right and wrong stressing points to demonstrate the difference. As feedback, the system plays a modified version of the learner's speech with exaggerated stress to notify him where his problem is. A Klatt formant synthesizer was used to modify the f0, rate and intensity of the speech.

Lu-2012 looked into the idea of exaggeration further by investigating methods that modified different parameters. They evaluated duration-based, pitch-based and intensity-based stress exaggeration, and in the end combined these three to perform the final automatic stress exaggeration, which, according to their experiment, raised the perception accuracy from 0.6229 to 0.7832.

3.5.3 Prosody Transplantation or Voice Conversion

In the previous sections we have seen that speech synthesis techniques can be used to provide feedback to the learner by modifying some prosody parameters

of the learner's speech in order to focus on particular problems or to exaggerate them. Other forms of feedback intend to modify the learner's voice by replacing or "transplanting" properties of the teacher's voice. The objective is then that the learner can hear the correct prosody in his/her own voice. This idea has been motivated by studies that indicate that learners benefit more from audio feedback when they can listen to a voice very similar to their own (Eskenazi 2009) or when they can hear their own voice modified with correct prosody (Bissiri et al. 2006) (Felps et al. 2009).

Prosody transplantation tries to adjust the prosody of the learner to the native's, so that the learner can perceive the right prosody in his own voice. According to the research of Nagano and Ozawa (1990), learners' speech sounds more like native after they tried to mimic their own voice with modified prosody rather than to mimic the original native voice. The effect is more remarkable if the L1 language is non-tonal, e.g. English and the target language is tonal, e.g. Mandarin (Peabody and Seneff 2006). Pitch synchronous overlap and add (PSOLA) (Moulines and Charpentier 1990a) has been widely used in handling pitch modifications. Many different approaches, namely time-domain (TD) PSOLA, linear prediction (LP) PSOLA and frequency-domain (FD) PSOLA, have been applied to generate effective and robust prosody transplantation.

Felps et al. (2009) provided prosodically corrected versions of the learners' utterances as feedback by performing time and pitch scale before applying FD PSOLA to the user and target speech. Latsch and Netto (2011) presented in their PS-DTW-OLA algorithm a computationally efficient method that maximizes the spectral similarity between the target and reference speech. They performed dynamic time warping (DTW) algorithm to the target and reference speech signals so that their time-warping become compatible to what the TD PSOLA algorithm requires. By combining the two algorithms, pitch-mark interpolations was avoided and the target was transplanted with high frame similarity. Cabral and Oliveira (2005) modified the standard LP-PSOLA algorithm, in which they used smaller period instead of twice of the original period for the weighting window length to prevent the overlapping factor to increase above 50%. They also developed a pitch synchronous time-scaling (PSTS) algorithm, which gives a better representation of the residual after prosodic modification and overcomes the problem of energy fluctuation when the pitch modification factor is large.

Vocoding, which was originally used in radio communication, can be also utilized in performing prosody transplantation and/or voice conversion. By passing the f0, bandpass voicing and Fourier magnitude of the target speech and the Mel-frequency cepstral coefficients (MFCCs) of the learner's speech, the vocoder is able to generate utterance with the L2 learner's voice and the pitch contours of the native voice. Recently, vocoder techniques have been also used in flattening the spectrum for further processing, as shown in the work of Felps et al. (2009).

An overview of the different types of perceptual feedback, the acoustic parameters they changed and the techniques they used, is summarized in Table 3.2.

Perceptual Feedback	**Reference**	**Modify/replaced parameters**	**Method or technique**
Speech synthesis	Meng et al. (2010)	F0, duration	STRAIGHT
	Meng et al. (2012)	F0, duration	decision tree, support vector machines
Emphasis and exaggeration	Yoram and Hirose (1996)	F0, rate and intensity	Klatt formant synthesizer
	Lu et al. (2012)	F0, duration and intensity	PSOLA
Voice conversion or prosody transplantation	Felps et al. (2009)	duration, pitch contour, spectrum	FD-PSOLA, spectral envelope vocoder
	Latsch and Netto (2011)	duration, pitch contour	TD-PSOLA, DTW
	Cabral and Oliveira (2005)	pitch and duration	LP-PSOLA, time-scaling

Table 3.2: Perceptual feedback, acoustic parameters modified or replaced and the techniques used.

3.5.4 Pros, Cons and Challenges of Perceptual Feedback

Compared to other feedback, the most obvious advantage of perceptual feedback is that the corrective information is provided in a most comprehensive way: via the language itself. To overcome the problem that it is hard for L2 learners to perceive the information in an utterance read by a native speaker, methods can

be applied to their own voice so that it is easier for them to tell the difference. However, the most directly way to tell the learners where the error is located is to show them via graphic or text. Hence, the ideal feedback that a CAPT system should provide is a combination of visual and perceptual feedback in the way that automatic diagnoses identify the errors and show them, while perceptual feedback helps to correct them.

One argument about perceptual feedback is: in most works, only prosodic errors like pitch and durations are taken care of, and in most experiments that prove the feasibility of perceptual feedback, the native and L2 speech that are used as input differ only prosodically. Although the results of these experiments show the advantage of perceptual feedback, e.g. the learners did improve their prosody after hearing modified version of their own speech than simply hearing the native ones, it is not the real case in L2 language teaching, at least not for the beginners, who might usually change the margins between syllables or delete the syllables depending on their familiarity to the syllables and their sonority (Carlisle 2001). These add difficulties to the forced alignment or dynamic time warping procedure, which is necessary before the pitch modification, and hence the outcome will also not be as expected (Brognaux et al. 2012b).

Perceptual feedback has been widely discussed and researched but not yet fully deployed in commercial CAPT systems. In order to provide more reliable feedback, the following considerations should be taken into account:

- For the moment, perceptual feedback should be applied to advanced learners who focus on improving their prosody, or to the case that only prosodic errors are detected in the learner's speech, i.e. if other speech errors are found, e.g. phoneme deletion, the learner gets notified via other means and corrects it; if only a stress is misplaced by the learner, he will hear a modified version of his own speech where the stress is placed right so that he can perceive his stress error.
- More robust forced alignment tool for non-native speech has been under development for years. In the near future, it should be able to handle pronunciation errors and provide right time-alignment even if the text and audio do not 100% match. Until then, an L1 independent forced alignment tool, which is one of the bottlenecks in speech technology nowadays, will be open to researchers, so in the near future, more accurate perceptual

feedback can be generated.

3.6 MaryTTS

The MARY (Modular Architecture for Research on speech sYnthesis) text-to-speech system (Schröder and Trouvain 2003) is an open-source text-to-speech synthesis platform written in Java. MARY was originally developed for German, and currently supports German, British and American English, French, Italian, Swedish, Russian, Turkish, and Telugu in its latest version 5.1.2. More languages can be added using the tools MARY provides. MARY uses MaryXML (Schröder and Breuer 2004) as its internal data representation, which not only guarantees the flexibility of MARY TTS by defining a unique data representation among all modules, but also provides data structure to store rich phonetic and linguistic information. Over the years, MARY has grown into an essential toolkit for speech processing with contributors from all over the world.

MARY is implemented in a server-client architecture, in order to handle multiple requests in parallel. MARY is highly modularized, and processes results step by step. New modules can be inserted into the workflow, or replace existing ones, as long as they implement the required interface. Likewise, modules in MARY can be easily reused and re-configured into a new work flow. Figure 3.2 (Schröder et al. 2008) shows a brief architecture of the MARY framework.

MARY server is equipped with a set of voices from different languages, and also several audio effects which can be applied to the synthesized output via signal processing tools. When the server starts, it reads the configuration files and decides on the language to be loaded and also the modules to be used in the workflow. Then the modules are initialized with parameters specified in their own configuration files. Typical input from the client is a text file, plus some settings to specify which voice should be used for synthesizing and what kinds of audio effect should be applied. The text is then processed by MARY text analytics tools, in the following steps:

1. Tokenization: a rule based tokenizer breaks the input text into tokens, i.e. words, numbers, special characters and punctuations. MARY uses *JTok*[12] in its latest version.

[12] https://github.com/DFKI-MLT/JTok

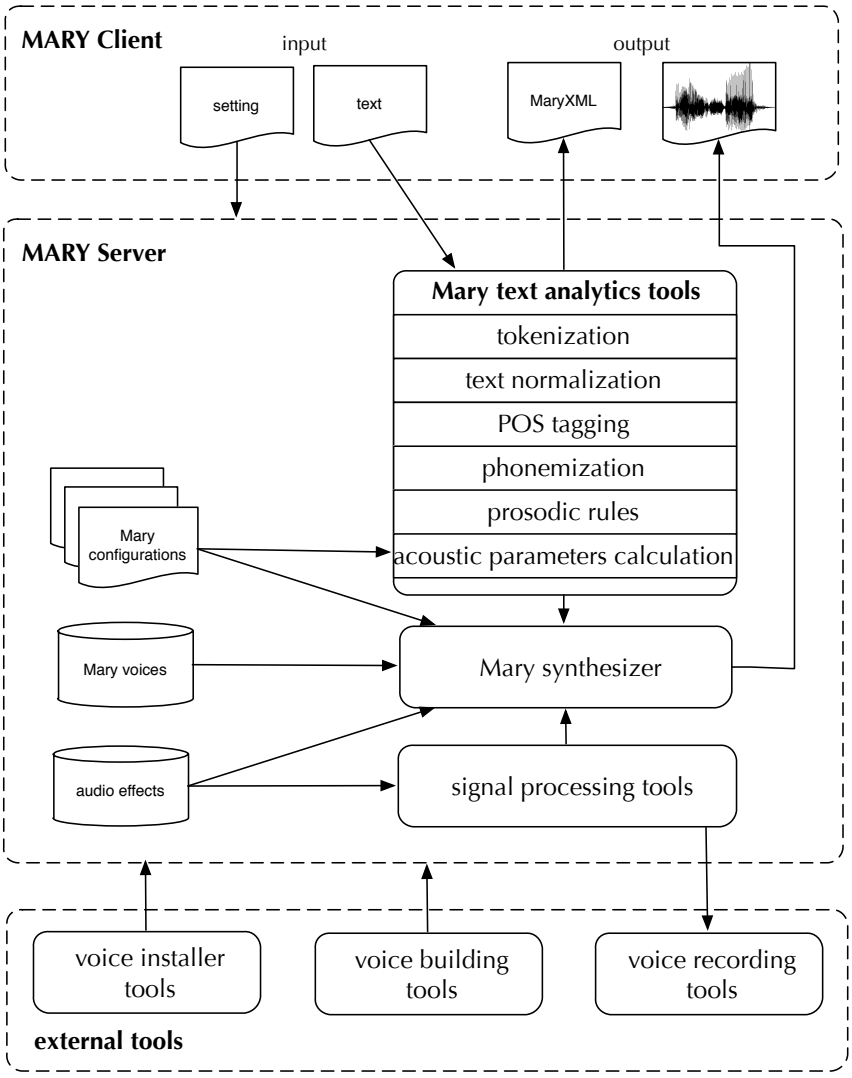


Figure 3.2: The architecture of MARY TTS system

2. Text normalization: in this step, the text is converted to its "spoken" form, including changing numbers to texts, e.g. "3" to "three", and also handling abbreviations, e.g. "etc." to "et cetera".

3. Part-of-Speech tagging: in order to determine the correct prosody, tokens need to be assigned with parts of speech, e.g. noun, verb, adjective etc.

Currently MARY uses POS tagger from OpenNLP[13].

4. Phonemization: this module generates phonemes from normalized tokens with POS informations. Known words are dealt with via extensive lexicons, while unknown words are processed with letter-to-sound conversion algorithms. The current phonemic transcription used in MARY is SAMPA[14].

5. Prosodic rules: This module provides prosody information of the to be synthesized speech, including prosodic boundaries, pitch accents and tones. The rules are generated from corpus analysis. Boundaries are expressed with break indexes assigned to each token, from value 2 to 6, meaning respectively potential boundary, intermediate phrase break, intra-sentential phrase break, sentence and paragraph final boundary. Tones including pitch accent tones, intermediate phrase boundary tones and intonation boundary tones, are assigned according to sentence type.

6. Acoustic parameter calculation: This module calculates the duration and *f0* parameters of the target speech, using normally classification and regression trees (CART). It converts MaryXML to a list of individual segments with duration and pitch features, which is ready for synthesizers to process.

MARY is equipped with a set of synthesizers using different technologies but sharing the same interface, including: an MBROLA[15] diphone synthesizer (the earliest integrated synthesizer), an LPC-based diphone synthesizer provided by FreeTTS[16], a unit selection synthesizer (Schröder and Hunecke 2007), and an HMM-based synthesizer ported from HTS project[17]. They generate synthesized audio files from text analytics results. MARY also provides various signal processing tools which are able to handle synthesized audio in different ways, including applying audio effect specified in client input.

In addition, MARY has a set of external tools that prepare voices for the system, they are:

[13]https://opennlp.apache.org/
[14]http://www.phon.ucl.ac.uk/home/sampa
[15]http://tcts.fpms.ac.be/synthesis/mbrola.html
[16]http://freetts.sourceforge.net/
[17]http://hts.sp.nitech.ac.jp/

- Voice installer tools: This module manages voices from existing sources, e.g. downloads and installs new voices and removes existing voices.
- Voice recording tools: A recorder designed to build speech synthesize database. It is equipped with some basic filters for noise reduction to ensure high quality recordings.
- Voice building tools: New voices for synthesizing can be created from this module. Depending on the synthesizing technology, e.g. unit selection or HMM-based, the methods for voice generation also differ. Voice building tools also integrates state of the art speech technologies toolkits to serve its purpose, among them are Sphinx, Snack[18], Praat, Festvox, HTK and HTS.

In general, results from all component are visible to both machine and humans, and can be adapted to serve special purposes and fed back to the system. For example, after text analytics workflow processes the text "2nd chance", a MaryXML file in Figure 3.3 is generated at the end, in which each component contributes the following:

- Tokenizer splits the phrase into tokens and marks them with *<t>* tag. It also checks paragraph, sentence and phrase boundaries and encloses tokens in *<p>*, *<s>* and *phrase* tags. Furthermore it creates *<mtu>* tag for multi-token unit like "2nd". *Boundaries* are added after punctuation tokens.
- Text normalization module converts special tokens to text in speech, e.g. "2nd" to "second".
- POS tagger analyze part of speech for each token and write to *pos* attribute in *<t>* tag. *Boundaries* for phrases are also determined in this module.
- Phonemizer generate phonemes for given tokens using informations above, and write to *ph* attribute in *<t>* tag. Phonemes can be derived from lexicon or user defined dictionary, or generated from letter in case of unknown words or strange symbols, as specified in *g2p_method* attribute. It also creates *<syllable>* and *<ph>* tags inside tokens to denote single phonemes.
- At last, the prosodic rules module applies *accent* to *<t>* and *<syllable>* tokens and assign *breakindex* and *tone* to *boundary*.

[18] http://www.speech.kth.se/snack/

3.7 Conclusions

This chapter firstly introduced CAPT as a special area in CALL. Different from other areas, CAPT relies heavily on signal processing methods to perform pronunciation error detection and also feedback generation. As a key challenge, different methods of pronunciation error detection were discussed and existing research on providing feedback via multiple means was summarized. Literatures show that perceptual feedback is more effective compared to visual feedback. We categorize perceptual feedback in 3 groups:

- via speech synthesis
- introducing emphasis and exaggeration
- performing prosody transplantation

All three methods modify or replace prosody parameters like *f0* and duration. The most used speech signal processing technology is PSOLA. Subsequently, the pros and cons of perceptual feedback were analyzed taking into consideration the difficulties of its implementation in commercial CAPT systems. Challenges in implementing prosody transplantation were also discussed.

This chapter also briefly presented forced alignment as the key technology in speech synthesis and recognition. Furthermore, since L2 speech data is necessary for training a language model, we also discussed several popular annotation tools and their main features.

MARYTTS has been released for over 10 years and has become a popular open source toolkit for speech synthesis. At the end of this chapter, we briefly introduced MARYTTS and focused on its text analytics components, including tokenization, text normalization, POS tagging, phonemization, prosodic rules and acoustic parameter calculation. We also presented step by step how a MaryXML file is generated by these components in a workflow.

```
<?xml version="1.0" encoding="UTF-8"?><maryxml
 xmlns="http://mary.dfki.de/2002/MaryXML"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
 " version="0.5" xml:lang="en-GB">
<p>
<voice name="dfki-spike">
<s>
<phrase>
<mtu orig="2nd">
<t accent="L+H*" g2p_method="lexicon" ph="' s E - k @ n d"
pos="JJ">
second
<syllable accent="L+H*" ph="s E" stress="1">
<ph p="s"/>
<ph p="E"/>
</syllable>
<syllable ph="k @ n d">
<ph p="k"/>
<ph p="@"/>
<ph p="n"/>
<ph p="d"/>
</syllable>
</t>
</mtu>
<t accent="!H*" g2p_method="lexicon" ph="' tS { n s" pos="NN">
chance
<syllable accent="!H*" ph="tS { n s" stress="1">
<ph p="tS"/>
<ph p="{"/>
<ph p="n"/>
<ph p="s"/>
</syllable>
</t>
<t pos=".">
.
</t>
<boundary breakindex="5" tone="L-L%"/>
</phrase>
</s>
</voice>
</p>
</maryxml>
```

Figure 3.3: Example of MARY text analytics tools output for given text "2nd chance".

Chapter 4

Forced Alignment Adaptation

In order to be able to pinpoint errors in the learners' speech data, and also to compare phonemes between learners' speech and the gold standard, we need to know when exactly each phoneme is pronounced. Forced alignment is ideal for this task since the texts, which learners are supposed to read is also available. However, the challenge in aligning L2 speech data is that L2 learners could produce various kinds of pronunciation errors as well as interruptions caused by hesitations and laughs. Hence we need to find a forced alignment tool that is highly error-tolerable, or can be customized to achieve this. Furthermore, our speech verification is processed in real-time, hence performance and robustness of the tool is also an important criteria.

4.1 Forced alignment using EHMM

Since we use many MARY components and also extended MaryXML files as internal file exchange format, we firstly evaluated the forced alignment components in MARY. MARY has a long history using two labeling tools: EHMM (a component in Festvox[1]) and Sphinx [2]. The Sphinx labeler uses semi-continuous HMMs while allowing skip rates, and trains context-dependent models for performing forced alignment using 13 MFCCs plus their delta and delta-delta. But MARY favors EHMM more because it trains context-independent models although no skip rate is allowed. It also has the following features (Schröder et al.

[1] http://festvox.org
[2] http://cmusphinx.sourceforge.net

2008):

- Trained models can be adapted since it can be initialized both with flat start or model.
- Different kinds of pauses can be modeled, including short, long and optional pauses. This allows a finer alignment on speech segments.
- Context-independent acoustic model ensures sharper labeling boundaries, and more resolution can be achieved via frame shift.

However, both forced alignment tools target at training models with corpus from single speaker, since the goal is to create synthetic voices using unit selection or HMM. But in our work, forced alignment needs to label speech data from different speakers. To achieve this, speaker-independent acoustic models need to be trained from speech corpus of different speakers.

Our corpus contains 1506 sentences read by a native Briton. To avoid monotone, we select these sentences from dialogs in different situations and ask the reader to read them with emotion. Among them, we choose 96 sentences that cover almost all phonemes and their diphone and triphone combinations, and have them read by German learners of English at different levels.

The EHMM model trained using the corpus above had good labeling on the gold standard speech data but relative poor alignment on the learners' data. We evaluate the result of the later by checking the alignment accuracy when the following four kinds of pronunciation errors happen:

- Insertion: an extra phoneme is inserted.
- Deletion: a certain phoneme is deleted.
- Substitution: a phoneme is replaced with another.
- Distortion: a phoneme is pronounced in the wrong way, and into a phoneme that does not exist in the target language.

Although we tried to apply different parameter such as windowing size and filter frequencies, the result of EHMM was still unsatisfying, especially in the case that learners pronounce certain vowels (e.g. /ʌ/ and /ə/) with error, in

which case the wrong pronounced phoneme and the ones next to it were all assigned the minimal phoneme duration in the alignment result. Although the alignment is less affected by distortion, insertion and deletion seriously damages the result. We evaluated 2 sets of learner speech data, and the best alignment achieved is shown in Table 4.1.

	Insertion	Deletion	Substitution	Distortion
Number	4	10	243	117
Error rate	100%	100%	68%	26%

Table 4.1: Evaluation of the EHMM alignment on 2 sets of learners' speech data

4.2 Forced Alignment using HTK

MARY has introduced an HTKLabeler since version 5.0 (Charfuelan 2012), which performs acoustic model training and transcription aligning using tools from HTK, a portable toolkit for building HMM-based speech processing tools. HTK is widely used in speech recognition and speech synthesis, and even DNA sequencing (Grundy 1997). The toolkit provides a whole set of tools for all kinds of HMM related purposes, including: (Young et al. 2002)

- Data preparation tools: *HCopy* copies input files to an output file while parameterizing them at the same time. *HList* is used to check the conversions of speech data. *HLed* performs required transformations to the label files, or unify them to a master label file. And *HLStates* is used for statistical purposes.
- Training tools: *HInit* and *HRest* initializes model with fully labeled bootstrap data; while *HCompV* initializes model with flat start, i.e. without bootstrap data. The training is performed with *HERest*, the core HTK training tool.
- Recognition tools: *HVite* is the core tool for recognition. It requires a word network as input, which can be either generated directly with *HBuild* or parsed from grammars using *HParse*. *HSGen* and *HDMan* are utility tools for generating language examples and managing dictionaries.
- Analysis tools: *HResults* is employed for evaluating the performance of recognition.

The main challenge of using HTK in MARY workflow is that MARY text analytics tools generate transcriptions without pause information. But in order to train a more robust model with better noise tolerance, 4 kinds of silence are to be modeled:

- S_u: silence at the beginning and end of an utterance.
- S_p: silence between phrases, typically caused by punctuation. They are normally shorter than the silence at utterance boundaries.
- S_w: pause between words.
- S_s: intra-segmental pause, caused by onset time of a stop consonant.

Among these silences, S_u,S_p and S_s are relative stable, as they can be inferred from transcriptions; while S_w tends to be flexible since it depends on the speaker's habit and judgment on syntaxes of utterances (Byrd and Saltzman 1998). However, in the context of L2 learning, the difference between S_w and S_s are much less, since learners are not familiar with the utterances and their rhythms, and could pause anywhere in the utterances, also between syllables. Furthermore, in runtime, the forced alignment labeler needs to deal with incoming speech data in open environments with background noises, which will contaminate the silences and pauses. Hence we adapt standard MARY HTK-Labeler, make it more suitable for aligning L2 speech data by modeling the 4 kinds of silences, and training iteratively with evolved models for stepwise optimization.

We start with the output of MARY phonemizer, the *MARY ALLOPHONES* files. These XMLs provide information for different silences using element *<boundary>s* with different *breakindex* attribute value. But they are mainly for speech synthesis and play far less important role for L2 speech data aligning according to the discussion above. From each *MARY ALLOPHONES* file, we generate a phoneme sequence from the following step:

1. Add *sil* to the start and end of a sentence, to represent S_u.
2. Add *ssil* to phrase boundary, to represent S_p.
3. Add *sp* to word and syllable boundaries, since learners may deal with S_w and S_s in the same way.

A phoneme dictionary that contains a unique phone set can be retrieved from all phoneme sequences. To follow the original HTKLabeler training steps, we also create phoneme sequences and a phone set without S_w and S_s, and utilize it as the prototype HMM for flat start.

Next we create models for the silences. We use default *sil* model from HTK and insert the silence model to the start and end of each utterance with

```
IS sil sil
```

To create the *ssil* model, we copy the 3 states of *sil* and tie them to their original states, using command

```
TI silst2 {sil.state[2],ssil.state[2]}
TI silst3 {sil.state[3],ssil.state[3]}
TI silst4 {sil.state[4],ssil.state[4]}
```

We also add transitions from the second to the fourth state and backwards, for both models, using

```
AT 2 4 0.2 {sil.transP}
AT 4 2 0.2 {sil.transP}
AT 2 4 0.2 {ssil.transP}
AT 4 2 0.2 {ssil.transP}
```

The modeling is illustrated in Figure 4.1.

The generated *ssil* model can then be used to create *sp* model by again copying the internal states and tying. Specifically, *sp* in our modeling is highly optional since we add it to each boundary of syllable and word, hence it requires a direct transition from entry to exit. The topology of this modeling is presented in Figure 4.2, using commands in Figure 4.3.

To avoid sequential silences/pauses, we merge them to single silence or pause, using the *ME* command. As shown in Figure 4.4, *sp* is merged when it's next to another silence. After the models are prepared, we start the *HERest* training with parameters in Figure 4.5, using a set of context-independent monophone

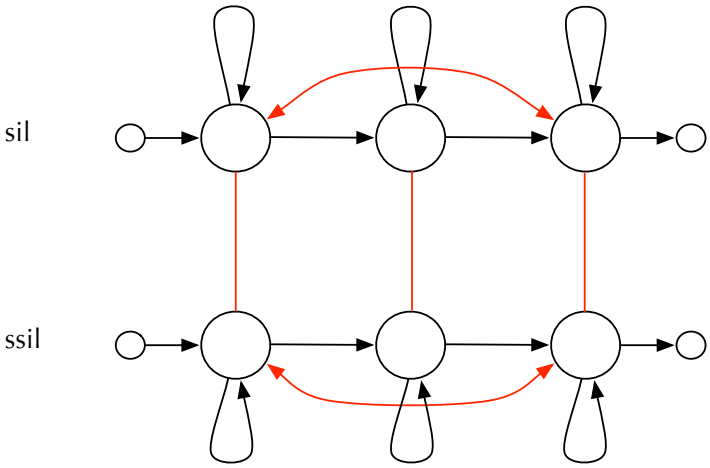


Figure 4.1: *ssil* Modeling

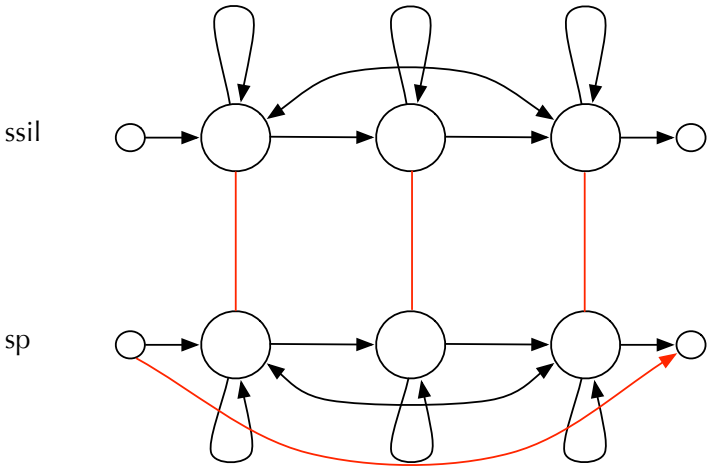


Figure 4.2: *sp* Modeling

```
AT 1 5 0.3 {sp.transP}
TI ssilst2 {ssil.state[2],sp.state[2]}
TI ssilst3 {ssil.state[3],sp.state[3]}
TI ssilst4 {ssil.state[4],sp.state[4]}
```

Figure 4.3: Commands for virtual pause modeling

HHMs with one Gaussian per state. The resulting HHMs are iterated using parameters in Figure 4.6, as provided by the HTKLabeler.

```
ME sp sp sp
ME sil sil sp
ME sil sp sil
ME ssil ssil sp
ME ssil sp ssil
```

Figure 4.4: Merging sequential silences and pauses

Module/Tool	Parameter	Value
#	ENORMALISE	FALSE
#	CEPFILTER	22
#	NUMCHANS	26
#	PREEMCOEF	0.970000
#	USEHAMMING	TRUE
#	NUMCEPS	12
#	TARGETRATE	50000.000000
#	WINDOWSIZE	100000.000000
#	PARAMETERKIND	MFCC_0_D_A
#	TARGETKIND	MFCC_0_D_A

Figure 4.5: Aligning parameters for adapted HTK labeling

Module/Tool	Parameter	Value
#	MAX_ITERATIONS	150
#	MAX_SP_ITERATION	10
#	MAX_VP_ITERATION	20
#	MAX_MIX_ITERATION	30

Figure 4.6: Parameters for iterative HTK training

We apply the trained acoustic model to realign the utterances. In order to keep consistency with MARY labels, all silences with duration equal to 0 are removed in the final labeling, and all remaining silences are marked with "_". And since we use MARY phonemes in the dictionary, no phone conversion is required. The output of the sentence *"I have to check my planner first."* is shown in Figure 4.7 as an example alignment.

The result is structured as following:

- # at the beginning shows the start of the transcription;
- the first column shows the time stamp of the end of each phoneme;
- the second column is the sampling frequency of the speech signal;
- in the third column are the aligned phonemes. _ means pause at the beginning and end, and eventually in between.

```
#
0.555000 125 _
0.675000 125 AI
0.735000 125 h
0.775000 125 {
0.835000 125 v
0.905000 125 t
0.930000 125 u
1.070000 125 tS
1.115000 125 E
1.225000 125 k
1.270000 125 m
1.325000 125 AI
1.495000 125 p
1.525000 125 l
1.605000 125 {
1.660000 125 n
1.810000 125 @
1.945000 125 f
2.090000 125 r=
2.215000 125 s
2.275000 125 t
2.405000 125 _
```

Figure 4.7: Example of forced alignment results.

The alignments were briefly evaluated by annotators during annotation, and the result was satisfying considering that there are pronunciation errors in the speech data. We ran the alignment on the 2 sets of speech data used for EHMM labeling evaluation, and the result is shown in Table 4.2. Insertion and deletion still cause misalignment, however, in most of the deletion cases, only the phoneme before or after the deleted one is affected and expanded to the boundary of the next or previous phone, whereas the deleted phoneme is set to the minimal phone duration. The remaining labeling errors in substitution and distortion cases are also acceptable.

	Insertion	Deletion	Substitution	Distortion
Number	4	10	243	117
Error rate	50%	20%	9%	5%

Table 4.2: Evaluation of the HTK alignment on 2 sets of learners' speech data

4.3 Conclusion

This chapter presented the preparatory work, which is required prior to L2 speech verification research and development. In order to pinpoint the pronunciation and prosodic error in the learners' speech, we needed to perform forced alignment with acoustic model that takes L2 speech error into consideration. Besides training with both native and L2 speech data, which were carefully selected to include almost all phoneme combinations, we also applied refined models for different kinds of silences, including silence at utterance boundary, segmental silence, inter-word silence and intra-segmental silence, to improve the forced alignment with more accuracy and better noise tolerance. Our adapted forced alignment recognizer based on MARY's HTKLabeler and HTKAligner is able to perform speaker independent alignment on the L2 learner speech data and maintains accuracy also when there are pronunciation errors in the learners speech data.

Chapter 5

MAT: MARY Annotation Tool

In this chapter we introduce MAT (MARY Annotation Tool) a tool for annotation of L2 learners' pronunciation errors at phoneme, syllable, word and sentence level. As presented earlier, many tools exist for speech data annotation. However, these tools are mainly intended to perform phonetic alignment and prosody analysis in general, with limited or no support for annotating L2 pronunciation errors at various levels. In fact, there has been no clear definition of a pronunciation error. In this thesis, the term "pronunciation error" refers to the difference between L2 and L1 pronunciation, which can be perceived by phoneticians. A pronunciation error can be either phonetic, e.g. to remove or replace a phoneme in speech, or prosodic in terms of rhythm and intonation (Witt 2012). In this work, we want to give phoneticians the opportunity to annotate all pronunciation errors they can find. We have summarized errors in phoneme, syllable, word and sentence levels, and listed them in Table 5.1.

Based on MARY (Schröder and Trouvain 2003), we work out a new annotation tool MAT (Ai and Charfuelan 2014) which particularly targets L2 pronunciation error annotation. MAT is built with the following features:

- Java based standalone program, hence runs on all platforms. There is also a Java applet version that runs in browsers.
- Open source software under GPL license.
- No previous knowledge to other software is required. MAT provides an intuitive interface, which is everything the annotators need.

Level	Errors	Description
Phoneme	Deletion	The phoneme is deleted in the learner's utterance.
	Insertion	A phoneme is inserted after the phoneme.
	Distortion	The phoneme is distorted in the learner's utterance.
	Substitution	The learner substituted the phoneme with another phoneme.
Syllable	Stress	The stress is misplaced by the learner.
Word	Long or short pause before or after word	L2 learners sometimes make long pauses in their pronunciation because of hesitation. Coughs, laughs and other interjections may also cause pauses in speech.
Sentence	Rhythm	The rhythm of the whole sentence is not smooth.
	Intonation	The sentence has problems with intonation.

Table 5.1: Pronunciation errors at various levels.

- Pronunciation errors in phoneme, syllable, word and sentence level can be annotated separately and easily.
- The selection of error types is configurable.
- Free switch between HTK and ehmm to perform forced alignment. Other forced alignment components are also embeddable with minimal adaptation and implementation of the interface.
- Annotated data is stored in extended MaryXML format (Schröder and Breuer 2004), can also be converted to TextGrid format for viewing in Praat.
- Speech analysis like F0, Energy and Spectrum graphs of speech data are provided as well.

5.1 System Architecture

The architecture of MAT is illustrated in Figure 5.1. The workflow can be briefly divided into two phases:
Firstly, input data, i.e. audio files and their transcriptions, are processed by

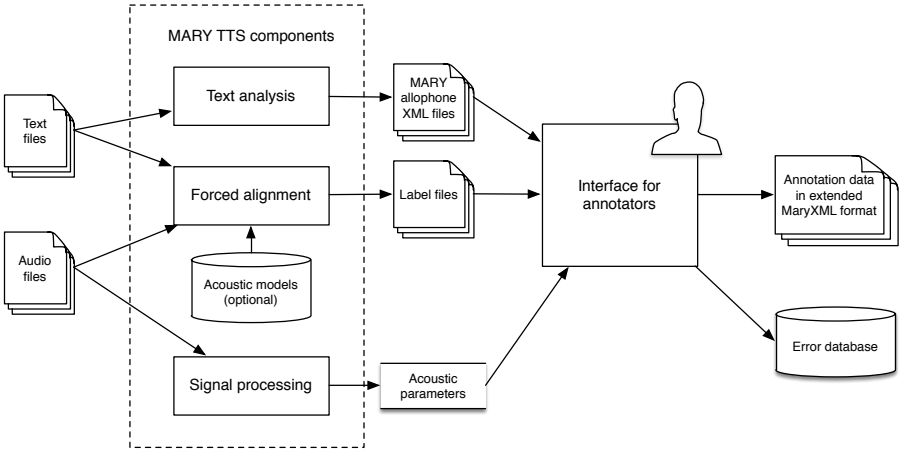


Figure 5.1: The Architecture of MAT

MARY components to generate intermediate data that can be rendered in the user interface:

- Transcription are parsed by MARY's text analysis tools to generate their MaryXML representations, which contain both phonetic and semantic information of the given texts, like the tokens they contain and the phonemes in each token.
- Audio files are processed by MARY's signal processing tools so that parameters of fundamental frequency, energy and spectrum can be calculated.
- MARY comes with two sets of configurations for performing forced alignment with HTK and ehmm. Dictionaries containing mapping of phonemes used in MARY are also provided. The output of forced alignment are so called label files, where each phoneme and its duration in a given text is stored.

In the second step, phonetic information in these intermediate files is rendered in an interface that interacts with annotators.

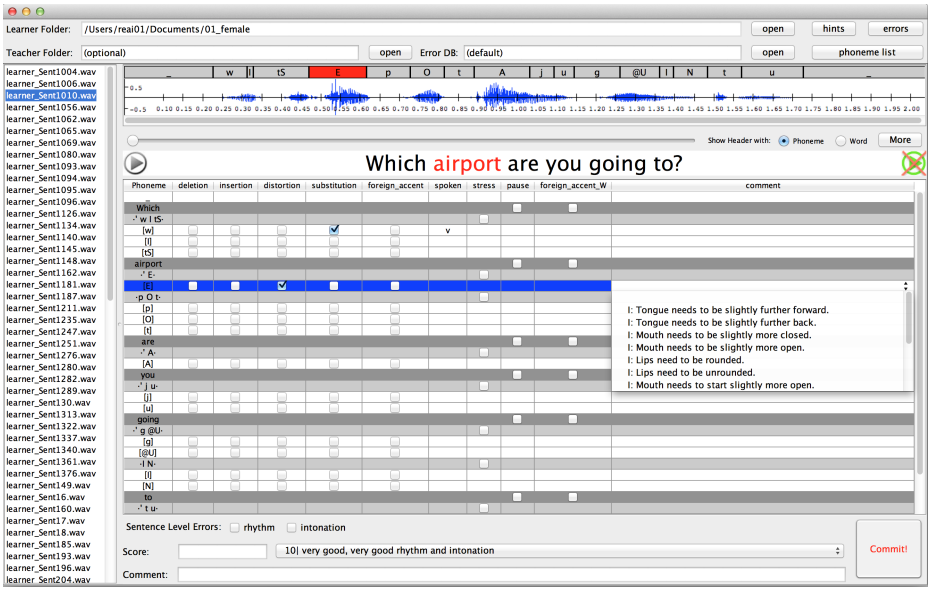


Figure 5.2: A Screenshot of MAT

5.2 User Interface

MAT is built with a user friendly interface, as depicted in Figure 5.2. Tasks for annotators are quite clear: to load a set of audio files with their transcriptions, to listen to each of them carefully, and to mark the pronunciation errors they find.

After loading the data with the *open* button above, the audios are listed on the left, and their phonetic information are shown on the right when chosen. Transcription of selected audio file is shown in the middle as text. The upper pane displays aligned audio with its transcription, segmented into phonemes or words. Annotators can choose to play the audio clip of these speech units by clicking on the header bar, or play any part of the whole audio by choosing a clip using the mouse in the waveform. In this way, annotators can easily focus on single speech units if they find them suspicious, without having to hear the long sentence repeatedly.

The lower pane is where the annotation takes place. Possible pronunciation errors are arranged in a table. The types of errors are read from columns, and the rows shows the speech unit each error could happen to, which can be phoneme,

syllable and word, ordered and grouped according to their appearances in the transcription. Generally, errors can be annotated by simply selecting the checkbox in the table cell corresponding to the speech unit in the row and error type in the column, beyond that, annotators are also asked to provide more information on these errors according to their perception. Hence each type of error is handled slightly differently:

- For *deletion*, no other annotation is required, simply to select the checkbox when a certain phoneme is removed in speech.
- For *insertion*, annotators should write the inserted phoneme in the *spoken column*.
- In case of *substitution*, the substituted phoneme should also be annotated in the *spoken* column.
- Distortion is handled with more diligence. Besides checking the *distortion* box, annotators also need to perceive in which way a phoneme is distorted and provide hints on how to correct it. Hints are chosen from a previously configured list. If the hint for correcting the distortion is not available in the list, it can be added from the configuration panel.
- If annotators find that a phoneme or word is spoken with an obvious foreign accent, they can check *foreign_ accent* or *foreign_ accent_ W*.
- If a stress is misplaced for syllables in words, *stress* should be checked.
- *Pauses* after certain words should also be annotated.

Sentence level errors are annotated below the table. Furthermore, annotators are asked to give a score by considering all phonetic and prosodic errors in the speech. These scores can be used to evaluate machine generated ones. Scores are by default integers between 1 and 10, and could be given either manually or chosen from the list. Fractional numbers like 8.5 are also allowed if for example annotator thinks the score should be between 8 and 9. Annotators can also leave comments if there is anything not covered by the user interface.

Annotators can define which types of error they want to annotate, by opening the error configuration panel. In the tree view annotators can adapt error types for each level. Furthermore, they can also modify the hints list before and during

Tongue needs to be slightly further forward.
Tongue needs to be slightly further back.
Mouth needs to be slightly more closed.
Mouth needs to be slightly more open.
Lips need to be rounded.
Lips need to be unrounded.
Mouth needs to start slightly more open.
Mouth needs to start slightly more closed.
Tongue needs to start slightly further back.
Tongue needs to start slightly further forward.
Lips need to be rounded at the end.
Vowel needs to be longer.
Vowel needs to be longer and tongue needs to be slightly further back.

Table 5.2: Collections of distortions specified by annotators.

annotating, so that the responsible hint can be conveniently chosen. Hints play a key role in classifying distortion errors as they not only show how annotators suggest to correct the distortions but also represent how the errors are made, e.g. hint "Lips need to be rounded" suggest that the distortion is caused by pronouncing with unrounded lips. Hints gathered by annotators for German learning English are listed in Table 5.2.

MAT also inherits speech signal processing ability from MARY. If annotators are willing to explore more information in the corpus, they can turn on the speech analysis view, as shown in Figure 5.4. From there annotators can inspect the waveform, spectrum, pitch contours and energy graph of the speech signal and view values at a certain time chosen via a mouse click.

When all annotations are done, annotators press the *Commit* button. Several checking are performed before the annotation is stored to file, including:

- Errors need to be annotated in the correct way, i.e. the previously described annotation behavior should be complied, for example if *substitution* is checked, the substituted phoneme should also be given in *spoken*.
- Only one type of phonetic error per phoneme. There is no clear boundary between substitution and distortion, a phoneme could be heavily distorted and in the end sound like another one, which is equal to substitution. In this case we ask annotators to choose only one of the two types of errors

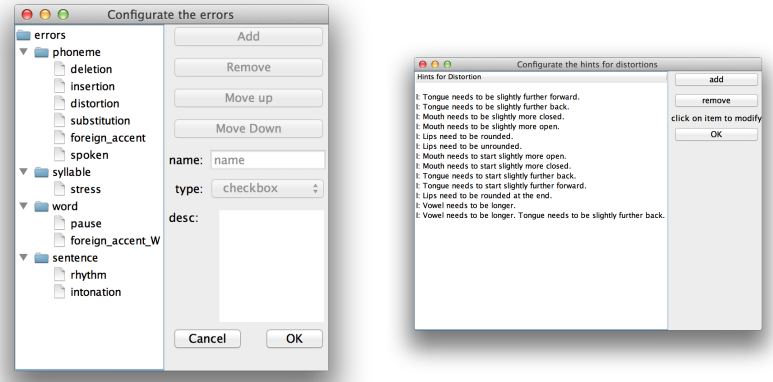


Figure 5.3: MAT Configuration Panels: (a) Configuring errors of different levels; (b) Managing hints.

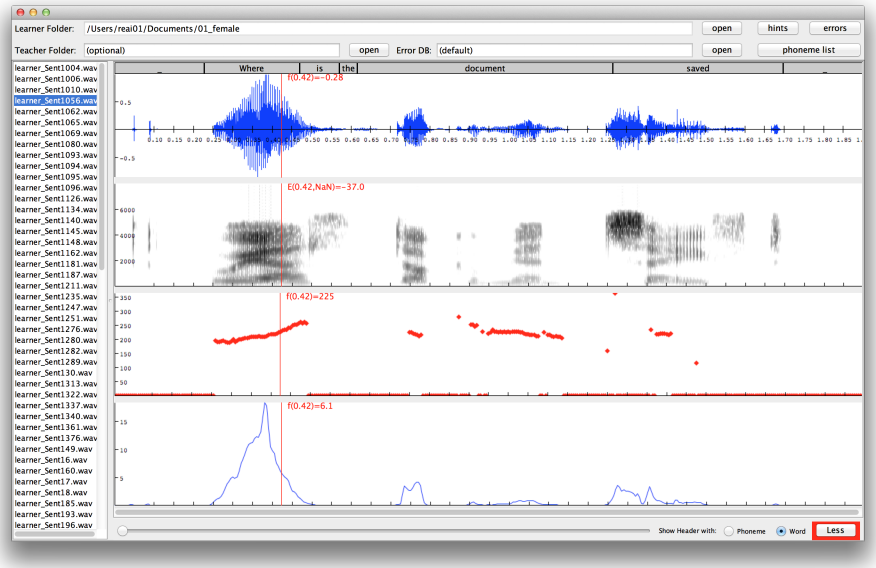


Figure 5.4: MAT Signal Processing View

according to their own perception. Another example is the mispronunciation of /ɐ/ to /ə/, some linguists consider it as substitution, while others regard it as deletion, since /ɐ/ is equal to /ər/. For this case, annotators need to be coordinated beforehand to ensure unified annotation.

- Phoneme written in *spoken* must be a MARY phoneme symbol. A built in mapping from IPA phonemes to MARY ones can be opened via a single mouse click for annotators to refer to.
- Score is optional but should not exceed 10 if manually given.

If all checking pass, the annotation is stored in extended MaryXML format, described in the next section.

5.3 Annotation Format

Pronunciation errors are stored in MaryXML files generated via MARY text analysis tools. Depending on the levels of error, several extension points are defined:

- Sentence level annotations are added as attributes in <s> element, including *comment*, *intonation*, *rhythm* and *score*.
- Word level annotations are added as attributes in <w> element, including *comment*, *pause* and *foreign accent*.
- Currently, only error with *stress* is syllable related, and it is added to <syllable> as attribute.
- Phoneme level annotations are added to <ph>, including *insertion*, *deletion*, *substitution*, *distortion*, *foreign accent* and *spoken*.

Figure 5.5 shows a snippet of annotated data for the sentence “Can I make an appointment for an interview?”. In this sample, the annotator has found that the /æ/ in “Can” is replaced with /ə/, and there is a pause after the word “I”. Judging from these errors and an intonation problem with the overall sentence, the annotator gave a score of 7.

```
<?xml version="1.0" encoding="UTF-8" >
<maryxml xmlns="http://mary.dfki.de/2002/MaryXML"
version="0.5" xml:lang="en-US">
<p>
<s comment="" intonation="true" opened="1" score="7">
<phrase>
<t g2p_method="lexicon" ph="’ k { n" pos="MD">
Can
<syllable  ph="k { n">
<ph p="k"/>
<ph p="{" spoken="@" substitution="true"/>
<ph p="n"/>
</syllable>
</t>
<t comment="pause after word" g2p_method="lexicon"
pause="true" ph="’ AI" pos="PRP" stress="true">
I
<syllable ph="AI">
<ph p="AI"/>
</syllable>
</t>
<t accent="H*" g2p_method="lexicon" ph="’ m EI k" pos="VB">
make
<syllable accent="H*" ph="m EI k" stress="1">
<ph p="m"/>
<ph p="EI"/>
<ph p="k"/>
</syllable>
</t>
...
<t pos=".">
?
</t>
<boundary breakindex="5" tone="H-H%"/>
</phrase>
</s>
</voice>
</p>
</maryxml>
```

Figure 5.5: Example of annotation output in an extended MaryXML file.

5.4 Conclusion

In this chapter we presented MAT, a tool for annotating different types of L2 speech errors. It uses open source components to perform speech signal analysis and forced alignment. Annotators can easily define which types of error they want to annotate via the configuration panels. MAT provides a user-friendly interface, in which annotators can play any audio clip, either well aligned speech elements like phoneme, syllable and words, or audio clip from their selection. Annotations are also easily performed via mouse clicks. MAT checks if the annotation is valid at the end of the annotating process, and saves valid annotation in extended MaryXML format, or prompts annotators in the case of invalid annotation.

Chapter 6

Phoneme Error Detection and Feedback

In this chapter, we present our methods of L2 pronunciation error detection at phoneme level, and the automatic generation of corrective feedback. The core of our method is to train an acoustic model using HTK [1] for phoneme recognition. As a preparation of the training, errors found by annotators are classified. Then a model can be trained from correct and error phonemes. Before recognition, a grammar, which takes consideration of all possible errors that can appear in the given sentence, is generated. By passing the grammar and the acoustic model, and also the learner's audio to the recognizer, we can identify possible errors in the learner's audio and also retrieve information for feedback from the recognizer's output.

Furthermore, we have built a self-learning architecture, enabling the speech verification process with the annotation process, as depicted in Figure 6.1. The learners' audio data are collected and annotated in the online system, the error database is updated on the fly, and the acoustic model is improved dynamically, hence speech verification will improve itself iteratively until enough phoneme errors are gathered and no more annotation will be needed. In our work, the speech verification tool is initialized with an acoustic model trained with audio data from 10 learners.

[1] http://htk.eng.cam.ac.uk/

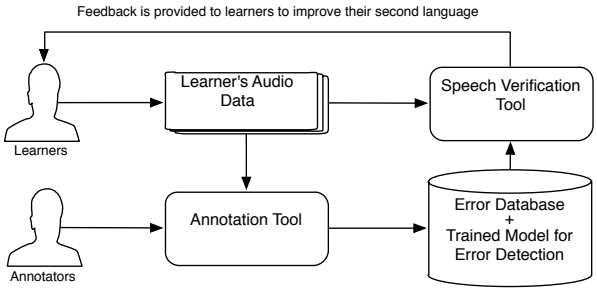


Figure 6.1: Bootstrapping and self-learning architecture.

6.1 Pronunciation Error Detection

6.1.1 Error Classification

Error types such as insertion, deletion and substitution requires no further classification. In the training phase, new phoneme sequences are created from original texts and their annotations to reveal such errors, e.g. if the learner deletes a phoneme in an utterance, it will also be removed from the phoneme sequence of this utterance. Since learners make different errors in pronunciation, for each sentence, there will be different pairs of *{phoneme sequence, speech data}* used as training corpus.

Distorted phonemes are classified by their ways of distortion and represented by new phonemes. For example, phoneme /ɑː/ in word 'are' can be distorted in two ways: either "The tongue needs to be slightly further forward." or "The tongue needs to start slightly further back.", so two new phonemes, A1 and A2, are created to represent wrongly pronounced /ɑː/. We use a database to keep track of all errors and integrate it into MAT, so every newly annotated error is automatically classified and stored.

The same strategy is also applied to phonemes with a foreign accent, i.e. if a phoneme is annotated with foreign accent, a new phoneme will be created as its distorted version. However, marking foreign accents is only useful in detecting error phonemes, but less helpful in feedback generation since it does not

provide any corrective information. In our work, annotators try to categorize foreign accents to distortion, substitution or deletion, and provide instructions for correction. For example, German tends to pronounce /ɐ/ as /ə/, this will be annotated as substitution. Considering /ɐ/ is also transcribed as /əʀ/, annotators may also mark the error as a deletion.

6.1.2 Language Model Training

The standard training for a phoneme recognition model using HTK is adapted to training a phoneme error detection model, as shown in Figure 6.2. The audio data contains both the gold standard data and the learners' data. The gold standard data are handled in the same way as a normal training for phoneme recognition. As for the learner's data, in order to keep the diphone and triphone information of error phonemes, we adjust the labels to make them represent the actually pronounced phoneme sequences. The output of MARY phonemizer is modified according to what type of error the corresponding audio file contains, which can be retrieved from the annotation.

- for deletion, the removed phoneme in the learner's speech is also removed from the output of the phonemizer;
- for insertion, the inserted phoneme in speech is also inserted before or after the target phoneme, based on the annotation.
- for substitution, the annotated phoneme, which is actually spoken by the learner, replaces the original one.
- for distortion, the newly created distorted phoneme replaces the original one.

For example, the sentence "I'll be in London for the whole year." should have the right labels as (in MARY phoneme representations)

I'll	be	in	London	for	the	whole	year.
A l	b i	I n	l V n d @ n	f O r	D @	h @U l	j I r

If a learner swallows /d/ in 'London', pronounces /ɔː/ in 'for' with backward tongue and replaces /ð/ with /z/ in 'the', the following labels are generated and used for training:

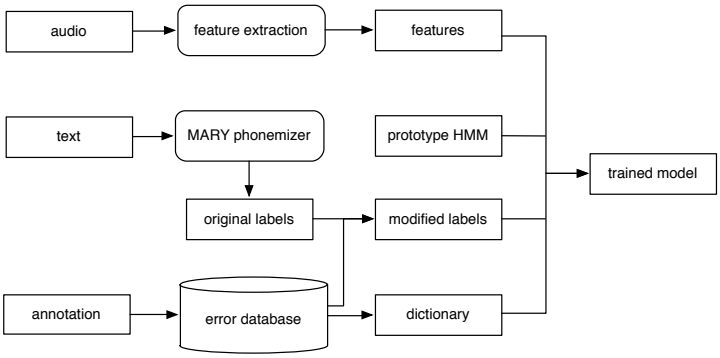


Figure 6.2: Process to train an acoustic model that detects phoneme errors.

I'll	be	in	London	for	the	whole	year.
A l	b i	I n	l V n @ n	f **O2** r	**z** @	h @U l	j I r

During training, distorted phonemes are treated the same as normal ones and are also added to the phone dictionary. Both the gold standard and the learners' data are send to iterations together so the trained model has information of inserted and removed phonemes, and is also able to deal with the differences between right phonemes and distorted ones.

6.1.3 Grammar Generation

To run phoneme recognition, HTK needs a grammar which defines the possible phoneme sequence of an input audio file. We generate grammars from the distribution of errors stored in database and texts that learners read. Taking the sentence "I'll be in London for the whole year" as an example, firstly, the correct phoneme sequence is retrieved from MARY phonemizer and surrounded with 'sil', which represents the silence at the beginning and the end of the sentence. The grammar looks like

(sil A l b i I n l V n d @ n f O r D @ h @U l j I r sil)

Next, all possible errors made by learners in the same sentence are applied to the grammar, in this case, there could be errors in the words, 'London', 'for', 'the' and 'year', after this step the grammar is:

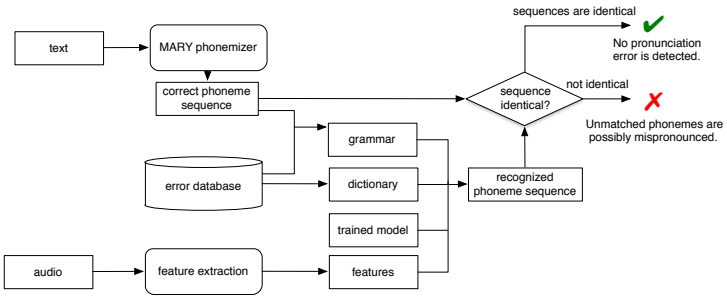


Figure 6.3: Workflow of automatic error detection.

(sil A l b i I n l **(V |A |O)** n **[d]** @ n f **(O |O2)** r **(D |z)** @ h @U l j **(I |I1) [(r |A)]** sil)

At last, we observe errors in diphones and triphones and add them to the grammar too. These include errors in the same word in other sentences, and also errors with phonemes from other sentences that have the same pre and post phonemes as appeared in the target phoneme sequence. In this case the only other error found is in the word 'be', so the final grammar is adapted to:

(sil A l b **(i |i1)** I n l (V |A |O) n d @ n f (O |O2) r (D |z) @ h @U l j (I |I1) [(r |A)] sil)

Unlike training an acoustic model, grammar is generated based on the incoming text in the runtime of error detection, and compiled to a word network before HTK can use it in recognition.

6.1.4 Error Detection

The process of automatic pronunciation error detection is illustrated in Figure 6.3. Phoneme recognition is performed using HTK with the trained model, adapted dictionary, generated grammar and extracted features. The recognition result is a phoneme sequence, which is then compared to the correct phoneme sequence generated from MARY phonemizer. If they are identical, no error is made in the learner's pronunciation; if not, possible phoneme errors can be

traced from the difference between the two sequences in a simple way:

- if a distorted phoneme, e.g. I1, appears in the result, the original phoneme is distorted by the learner.
- if a phoneme from the correct sequence is missing, inserted or replaced in the result sequence, a deletion, insertion or substitution error can be inferred.

6.1.5 Evaluation

We run automatic error detection using the trained model on 4 sets of sentences, which have the same texts as the sentences used for training but read by 4 new learners. The results are then converted to extended MARY ALLOPHONES XML data with the same format as the annotations, so that they could be opened with the annotations tool for double-checking. The following are the results of comparing the generated data and the annotations, i.e. comparing errors detected by the system and errors found by annotators.

	true positive	**false positive**	**false negative**	**total**	**recall**	**precision**
deletion	46	0	4	50	92%	100%
insertion	17	0	1	18	94.4%	100%
substitution	1264	14	2	1266	99.8%	98.9%
distortion	745	102	26	771	96.6%	88.0%
total	2072	116	33	2105	98.4%	94.6%

Table 6.1: A statistic of the error detection result. True positive: actually detected errors; false positive: correct pronounced phonemes detected as errors; false negative: errors not detected.

The result shows very high precision and recall for error types as deletion, insertion and substitution. In fact, the four deletion errors, which the system failed to detect, never appeared in the training data, e.g. for the word 'central', the phoneme /r/ is removed by one of the testers. As for substitution, German tends to make the same substitution errors when speaking English, like replacing /ð/ in 'the' with /z/, and /z/ in 'was' with /s/. There are no new substitution errors in test data. Detecting distortions is not an easy task. In the 745 errors found, 114 of them are falsely categorized although they were successfully detected as

distortion, e.g. the system returned “Tongue needs to be slightly further back." but the annotator thinks “Tongue needs to start slightly further back."

Despite a relative low accuracy at detecting distortion, we think the method is feasible for industrial CAPT applications, and we believe that the accuracy will improve if more training data is provided.

6.2 Feedback for Phoneme Errors

6.2.1 Methods

Finding the errors is not the final destination. Intuitive feedback is needed so that learners know not only where the phoneme errors are but also how to correct them. The advantage of our method is that these corrective information are retrieved at the same time as errors are detected. For example, if ‘O2’ is found in the word ‘for’ in the learner’s pronunciation, we can show the annotation, from which this distorted phoneme is categorized, directly to the learner, and in this case it’s “The tongue needs to be slightly further forward.". Or, if ‘London’ is recognized as ‘l **O** n d @ n’ instead of the correct ‘l **V** n d @ n’, we can tell the learner that he pronounces the first ‘o’ like /ɔː/ in ‘often’, but it should be like the /ʌ/ in ‘cut’.

Simply displaying texts as instruction to learners is insufficient. An example of how exactly the error phoneme is pronounced, is needed. However, playing the gold standard version of the error word or sentence to the learners is not enough, because they may not be able to perceive the difference between the error phoneme and the correct one due to their L1 background (Flege 1995). In our evaluation system, we use a new type of feedback: the learner’s own voice.

For each phoneme, we find two words that are pronounced correctly from the voice data of a given learner. E.g. for /ʌ/ we have ‘coming’ and ‘utter’. The words are chosen in the way that they have the target phoneme in different location and with different combination with other phonemes, and better represented by different letters. For /ʌ/, ‘but’ + ‘cut’ is not a good choice, neither is ‘but’ + ‘utter’. Next, audio clips for each phoneme and its two example words are extracted. We also record some clips from native speaker. They are used for generating the final feedback. For example, if ‘l **O** n @ n’ is in the recognition

result instead of 'l **V** n d @ n', the learner is presented with the a window as in figure 6.4. If she clicks on 'London' on the first row, the gold standard version of 'London' is played. If she clicks on the /ʌ/ on the second row, the following concatenated audio is played, where **/ʌ/** and **London** are extracted from the gold standard voice, other underlined text are clips from the learner and the rest are pre-recorded audio prompts. We extract audio clips of phonemes and words by using the forced alignment information from the trained model (for the gold standard voice) and the phoneme recognition result (for the learners' voice). And the text is also displayed on screen.

"You pronounced **/ʌ/** in **London** like */ɔː/* in *'all'* and *'door'*. It should sound like */ʌ/* in *'coming'* and *'utter'*. Please try again."

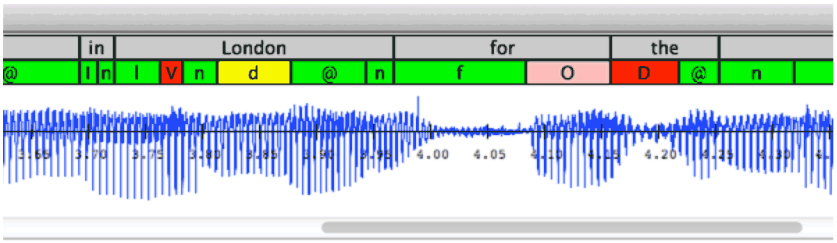


Figure 6.4: A window showing the the learner's phoneme error in our evaluation system. The background colors of the phoneme show what error type the learner has made: green: correct, yellow: deletion, red: substitution, pink: distortion and purple: insertion (not presented in this example).

Similarly, if /d/ and /ɔː/ are clicked, the following texts are displayed and the corresponding audios are played:

"You missed **/d/** in **'London'**, it should sound like */d/* in *'deny'* and *'good'*. Please try again."

"There is a little problem with the **/ɔː/** in **'for'**, it should sound like */ɔː/* in *'all'* and *'door'*. The tongue needs to be slightly further forward. Please try again."

In this way, learners are taught how to pronounce a phoneme correctly, in a way that they are surely able to: by recalling how they used the sound right in other words. Learners can perceive the difference between correct and incorrect phonemes better, if they compare their own voices rather than comparing their voice with the gold standard (Flege 1995).

6.2.2 Evaluation

To use a learners' own voice data as feedback, poses a dilemma: before a learner can pronounce a phoneme correctly, his/her correct voice data for this phoneme is not available. This problem becomes especially crucial when dealing with distortion because for some phoneme, beginners couldn't even pronounce them correctly only once, e.g. /ə/ at the end of 'number' or 'year'. In this case, we only display the annotator's hint as text, e.g. "The mouth needs to be slightly more open", to check if the learner manages to correct the pronunciation.

In our experiment, testers follow the scenario described in these steps:

1. The learner chooses a file with error and is presented with the window as in Figure 6.4. But at this time, clicking on the error phoneme only displays feedback as text.
2. The learner could click on the gray words on the first row to play the gold standard as often as she wants. When she thinks she has the information in the feedback, she presses Record and speaks the whole sentence into the microphone again. The automatic error detection process runs again and presents the learner with a new window. In this window, clicking on the error phonemes not only displays text but also plays audio.
3. If there are still errors shown in the new window, the learner can play the audio and check the text until she thinks she's able to correct all the errors, and then records again.
4. Another window should then show if the learner has corrected all her errors.

	total	corrected after viewing text	corrected after listening to audio
deletion	20	19	20
insertion	6	6	6
substitution	641	430	608
distortion	338	104	125

Table 6.2: Statistics showing how feedback helps learners correct their pronunciation errors.

The results of two of the four tests that learners took part in in the experiment is shown in table 6.2. By deletion and insertion, it's helpful enough to display the

text information to make the learners aware of what they missed or inserted. The only case that require a second time was a mistake: the learner did pronounce /s/ in 'months', but in the first time correction she focused on the /s/ and didn't pronounce the /θ/ before it clearly enough.

The case with substitution is interesting. We think there are three types of substitution. The first is like replacing /z/ with /s/ in 'Please' or /v/ with /f/ in 'of', the cause of which might be that learners forget the spelling rules. If prompt texts such as "like /z/ in 'zero' " or "like /v/ in 'very' " are given to learners, they understand instantly what the correct pronunciations are. In the learners' first attempt, most of this kind of substitution and those that were made by mistake were corrected. Here, example words play an important role. Both learners have error with replacing /əʊ/ with /ɔ/ in 'most'. The learner with the example words 'blow' and 'over' succeeded in correcting the error by only reading the textual feedback, while the other learner with the words 'hotel' and 'go' had to hear her own pronunciation of these two words to make successful correction. The second type is similar to the first, only that the original phoneme does not exist in the learners' mother tongue, and is replaced with an existing one, e.g. /θ/ with /d/ in 'This'. The difficulty here is that a learner may not know how to pronounce it and makes no correct pronunciation on this phoneme, and hence no correct audio template can be generated. If this happens, our feedback won't work. The learner has to be taught systematically how to pronounce it. The third type is more in the way of a distortion, the error phonemes are distorted so much that they become another phoneme, e.g. replacing /æ/ with /e/ in 'exactly' or /ʌ/ with /a/ in 'number'. These errors are hard for learners to correct but after hearing their correct version of the same phoneme in other words, a large amount of them can be fixed.

The result shows that our feedback is not as good at helping to correct distortion errors as with other error types. Learners were able to correct around a third of the errors by changing their mouth, tongue or lips according to the textual instruction. Playing audio wasn't helping much. We also noticed that learners could distort a phoneme on her second attempt, although the same phoneme was correct on her first try. Our conclusion with distortion is that it is caused by the learners' habit or accent, and might be hard to correct at once. In fact, distortion is still acceptable as long as the error phoneme is not distorted into a new phoneme, because learners may not even be able to perceive the difference between the correct phoneme and their distorted version, and will feel confused

or discouraged if they are told that they pronounce wrongly every time they try to correct it.

6.3 Conclusion

This chapter described in detail our methods for pronunciation verification. The innovation in our method is classifying L2 phoneme errors based on linguistic judgments. By using a delicate designed annotation tool, linguists are able to classify an error phoneme to a known type or define a new type of pronunciation error for each phoneme, which means error and correct phonemes are not classified by their acoustic behaviors such as spectrum, but rather features that can be perceived by humans. Our evaluation shows that the classification is effective and could detect L2 phoneme errors with high accuracy. Our method also has the advantage that corrective feedback could be generated at the same time as when errors are detected, since the information required is already provided in the error recognition results.

In addition to displaying instructions on how to pronounce correctly in text, we also try to provide perceivable feedback by means of utilizing the learners' own pronunciations. Our progressive evaluation shows that text instructions are already very helpful for learners to correct deletion and insertion. Perceivable feedback is more effective when dealing with substitution. But neither kind of feedback can help learners to overcome distortion efficiently. Future work may seek to integrate video instruction, in which the movement of the mouth, tongue and lips can be better explained, when distortion is detected.

Our method applies to all languages as long as they are supported by the core components: MARYTTS and HTK. However, since they are both open source, new languages could also be supported by providing acoustic models, including tonal languages like Chinese and Vietnamese (Huang et al. 2004) (Quang et al. 2010). For certain languages, the amount of corpus required for training needs to be taken into consideration.

The workflow also needs to be adapted if the mother tongue of the target learners change, since different L1 may affect L2 in different ways and causes new errors. The flexible configuration in MARY Annotation Tool is designed with this in mind. Annotators can easily add new types of errors that exists in a new target

learner group. The rest in the workflow including training acoustic model and error recognition should work with minimal adaptation.

Chapter 7

Prosodic Error Detection and Feedback

Prosody plays an important role in language. In tonal languages like Chinese and Japanese, pitch accent is used to differentiate words. Also in non-tonal languages as English, listeners assign meanings to prosodic patterns. For example, to answer a question "*Who sent Melina home?*", new information would be accented, like "*Her* ***boyfriend*** *did.*". Further more, pitch accent is used to correct information in the context, e.g. "*– Melina was brought home by her father yesterday.*" "*– Really? I thought her* ***boyfriend*** *sent her home!*". Hence a pitch accent can have a sort of meaning similar to contrast (Pierrehumbert and Hirschberg 1990). Although this is argued by Swerts et al. (2002), stating that pitch accent does not imply a particular meaning, but merely provides information with enhanced importance, it is generally accepted that even in non-tonal languages, prosody conveys the speaker's emotion and focus, so that two literally identical sentence with different prosody could have distinct meanings (Cho 2002).

In the context of CAPT, learners would try to read sentences after they have heard the gold standard version, just like they read after teachers in classroom, i.e. learners would try to mimic what they have heard. Since most CAPT materials are dialogs rather than articles, the emotion and focus of the dialog partners must be taken into consideration. Assigning prosodic stress to particular words can clarify the meaning of a sentence differently, e.g. these literally identical sentences are used in different dialogs:

- - Who was home yesterday to sign for the packet delivery?
  - **I** was home yesterday.
- - I don't think you were home yesterday.
  - I **was** home yesterday.
- - Where were you yesterday?
  - I was **home** yesterday.
- - When was the last time you were home?
  - I was home **yesterday**.

Due to the possibility of sentence stress variation, we can't judge the correctness of the prosodic stress from the learners' speech alone, but have to compare them with the gold standard sentences. Prosody is determined by several acoustic characteristics. For example, a stressed syllable is different from an unstressed one in the following features:

- Intensity: stressed syllables tend to be louder; (McGilloway et al. 2000)
- Fundamental frequency: stressed syllables are usually higher in pitch; (Cahn 1990)
- Duration: stressed syllables may be longer; (Bou-Ghazale and Hansen 1997)
- Frequency spectrum: vowels are pronounced with full articulation; (Bou-Ghazale and Hansen 1998)
- Wavelet based subband features: stressed syllables yield different results in multi-rate subband analysis. (Sarikaya and Gowdy 1998)

When any of these features in the learners' speech mismatch with the ones from the gold standard, the utterances would sound like a "foreign accent", as can be noticed by native speakers. Generally, it is caused by the segmental and suprasegmental interferences from L1 (Ingram and Pittman 1987), when L2 learners apply L1 categories to perceive L2 segments and consequently use L2 articulatory patterns to pronounce them (Valdman 1976) (Shen 1990). To be more specific, the learners' failure in applying the correct prosody can be reasoned from the following aspects:

- Unfamiliarity in L2: When learners (usually beginners) are unfamiliar with the structure of given sentences, especially long and complex ones, even though they have heard the gold standard versions, it would still be hard for them to follow. A usual work-around is: they decompose the whole sentence into shorter phrases and read them in sequence. However, the dividing is based on their familiarity of existing parts of the sentence, not grammar, hence causes mismatch of the original syntactic structure and leads to a different prosody.
- Influence from L1: Some languages, like Italian, have their own prosody rules (Burzio 1994) and beginners tend to apply the same rules in L2 learning. And if L1 has a different syllable structure with L2, or an L2 syllable structure does not exist in L1, learners may produce prosodic error due to structure deformation.(Imoto et al. 2002)
- Failure in prosody perception. In some cases, learners fail to perceive the pitch variation and stress of the gold standard sentences, and therefore are not able to produce the correct prosody.(Paul et al. 2005)

Modern CAPT applications need not only detect the prosody differences but also provide feedback with which learners can perceive these differences. In this chapter, we aim at comparing the three main features in prosody: energy, pitch and duration, which mainly contribute to the prosodic stress, melody and rhythm. The comparison result is presented to learners as feedback in a perceivable way.

7.1 Word Stress Misplacement

Word stress misplacement is common for L2 beginners while pronouncing poly-syllabic words. They may be either lack of knowledge about the L2 stress rules, e.g. children tend to pronounce all syllables with equal stress (Allen and Hawkins 1980); or influenced by their L1 phonological systems, e.g. Italian tend to stress the last syllable while speaking English (Flege et al. 1999). Learners are more subjective to this type of error if the target language has variable stress which can't be predicted, such as in English.

There has been much research in stress identification, especially in combination with emotion analysis and classification. A well known approach is to measure

phonetic parameters in order to simulate characteristics of human auditory system, e.g. LPC (Linear Prediction Characteristics) based Cepstral coefficient (Atal 1974) and Mel-Frequency Cepstral Coefficient (Davis and Mermelstein 1980). Nwe et al. (2003) have proposed a new robust method by calculating Log-Frequency Power Coefficients (LFPC). They segment the speech signals into short time ($20ms$) windows and move them at frame rate $13ms$ and then calculate the frequency content in each frame using the Fast Fourier Transform (FFT) method. The results are accumulated into a bank of log-frequency filters that split input speech signal into multiple outputs. Energy in the m^{th} filter bank output is calculated as:

$$S_t(m) = \sum_{k=f_m-\frac{b_m}{2}}^{f_m+\frac{b_m}{2}} (X_t(k)W_m(k))^2 \quad m = 1, 2, ..., 12$$

where $X_t(k)$ is the k^{th} spectral component of the windowed signal, t is the frame number, and f_m, b_m are the center frequency and bandwidth of the m^{th} subband respectively. And the energy distribution parameters among subbands are calculated as:

$$SE_t(m) = \frac{10log_{10}(S_t(m))}{N_m}$$

where N_m is the number of spectral components in the m^{th} filter bank.

The result $SE_t(m)$ is further applied to emotion classification. However, in our research, the task is relative simpler: we are not interested in the exact energy distribution in the whole sentence, but only in the comparison of distributed energy to each syllable inside a word. And since the window sizes are equal, we could simplify equation 7.1 with:

$$E_t(s) = \sum_{s_{min}}^{s_{max}} X_t(k)^2$$

where $X_t(k)^2$ describes the relative energy in each frame and $E_t(s)$ is the total relative energy distributed to each syllable lasts from frame s_{min} to s_{max}. Syllable with greater $E_t(s)$ is regarded stressed by the learner.

Since the correct stress can be retrieved from the MARY text analysis tool, we compare the actual pronounced stress with the correct one, and notify the

learner in case of inconsistency.

7.2 Duration Difference

Duration is an important factor of prosody, and also a perceptual cue to judge if an utterance is spoken natively (Mennen 2007). Hence it's also helpful for learners to pay attention to the duration of each word so that they could speak in a more native way. Since learners keep their own pace while speaking, some talk generally faster and some slower, we don't compare the differences of each phoneme's duration in the gold standard and the learners' speech, instead, we compare the relative duration, i.e. how much of the entire time in the utterance is distributed to each word. Suppose we have two phoneme sequences G (for the gold standard) and L (for the learner), each contains n words and each word contains several phonemes:

$$G = g_1^1, g_1^2, ..., g_2^1, g_2^2, ..., g_n^1, g_n^2, ...$$

$$L = l_1^1, l_1^2, ..., l_2^1, l_2^2, ..., l_n^1, l_n^2, ...$$

At first, we determinate whether the learner speaks faster than the gold standard or slower. This can be roughly estimated by comparing the sum of durations of each phoneme:

$$r = \frac{\sum_{k=1}^{n} l_k^{1\to...}}{\sum_{k=1}^{n} g_k^{1\to...}}$$

Next, we normalize the phonemes to calculate the duration difference. To get more precise results, the normalization is done to phonemes word by word which they belong to. Taken the first word as an example:

$$G'_1 = g_1'^1, g_1'^2, ...$$

$$L'_1 = l_1'^1, l_1'^2, ...$$

We can derive the *duration scale factor* of the kth phoneme in the first word:

$$d_1^k = \frac{g_1'^k}{l_1'^k} r$$

Then we can tell the learners to

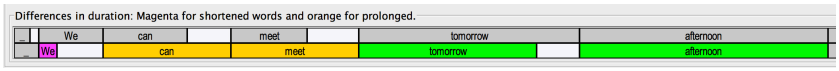


Figure 7.1: Feedback to learners by displaying differences in different colors.

- shorten a phoneme if its *duration scale factor* is less than 1.0;
- prolong a phoneme if its *duration scale factor* is greater than 1.0.

Since phoneme-level prosody feature is partially covered in word stress verification, we focus on detecting word-level duration differences. The *duration scale factor* of the first word can be derived from:

$$G'_{t1} = g'^{1}_{1} + g'^{2}_{1} + ... + g'^{n}_{1}$$

$$L'_{t1} = l'^{1}_{1} + l'^{2}_{1} + ... + l'^{n}_{1}$$

$$d_{t1} = \frac{G'_{t1}}{L'_{t1}} r$$

The duration differences can be easily illustrated using graphs like Figure 7.1

7.3 Pitch Difference

Fundamental frequency (*f0*), also referred to as pitch in the perspective of human perception, is another deterministic factor of prosody. The fundamental frequency of human speech is determined by vibration rate of the speaker's vocal fold, and is different among speakers: *f0* for an adult male is from 85 to 180 Hz, and for an adult female from 165 to 255 Hz (Baken and Orlikoff 2000). Speech with a higher *f0* is perceived as with higher pitch. For example, w-words in questions are usually pronounced with higher pitch than other words in the same sentence.

f0 can be easily measured at a given point in the duration of the fundamental period, but it's not in the interest of our work because it's more important to know how *f0* changes over time in the course of an utterance, i.e. the so called pitch contours. A speech signal can change in all spectral characteristics,

including *f0*, in even very short time intervals, hence it's difficult and time-consuming to measure the duration of each pitch period and then calculate pitch contours. There has been several robust methods to track pitch contours, like in the work of Ross et al. (1974), Miller (1975) and Dubnowski et al. (1976). These methods are either based in frequency or time domain, or joint time-frequency domain, and generally contain 3 steps:

- preprocessing: set filters to remove higher harmonics or vocal tract resonances, also applies center clipping to reduce harmonic structure.
- *f0* candidate prediction: various methods are used in this step to obtain possible *f0* values, e.g. spectral equalization LPC method (Atal and Rabiner 1976), simplified inverse filtering technique (Markel 1972), average magnitude difference function (Ross et al. 1974), etc..
- post-processing: remove erroneous results from pitch doubling and pitch halving, using methods such as median smoothing (Rabiner et al. 1975).

Newer approaches like presented by Neiberg et al. (2011) improve the accuracy of pitch tracking slightly but also add complexity. In order to integrate with our other open source components, we choose the Snack toolkit (Sjölander et al. 1999) as a pitch tracker. Snack supports a large amount of audio file formats and signal encoding schemes, and is implemented as a Tcl/Tk extension. The core code of Snack is implemented in C, while the Tcl/Tk script layer configures the objects and links them together. Snack can be extended at C level as well as at script level. Benefited from the impressive performance of C and powerful graphic toolkit of Tcl, Snack has become one of the most popular speech signal processing toolkits since its release in 1998, and was built into many famous speech analysis tools, among them are Wavesurfer (Sjölander and Beskow 2000) and Transcriber (Barras et al. 2001).

Snack treats each audio file as a sound object and provide a set of functions for manipulations, including recording, playback, editing and other signal processing. In our work, we use the method *pitch* to extract the pitch contours, with parameters[1]:

- -method ESPS. This generates pitch values in a list containing 4 fields for each frame: pitch, probability of voicing, local root mean squared

[1] http://www.speech.kth.se/snack/man/snack2.2/tcl-man.html

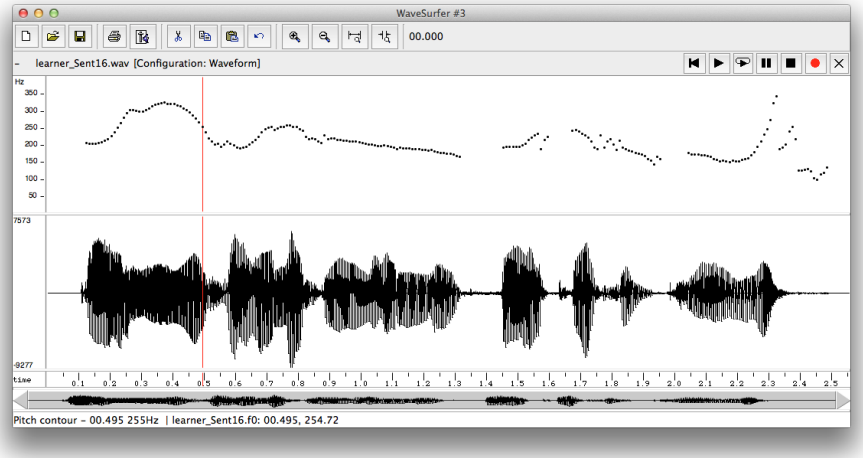


Figure 7.2: Pitch Contours calculated using Snack in Wavesurfer, for the utterance "Are you available on the thirteenth of June?", read by a native female Briton.

measurements, and the peak normalized cross-correlation value that was found to determine the output pitch value.

- -framelength 0.005, specifies the intervals between the values.
- -windowlength 0.04, specifies the size of window in seconds.
- -minpitch and -maxpitch, these two specify the minimum and maximal pitch value. To raise the accuracy of pitch extraction, these are set based on the speaker's gender: *-minpitch 60 -maxpitch 300* for male and *-minpitch 100 -maxpitch 500* for female.

The script for extracting pitch contours using Snack is quite straightforward:

After running the script on both the learner and the gold standard speech data, we compare the pitch differences based on the generated output files. In the first step, the pitch values sequences are read from the output files to arrays.

$$G = g_1, g_2, ...$$

$$L = l_1, l_2, ...$$

```
package require snack

snack::sound s

s read [lindex $argv 0]

set fd [open [lindex $argv 1] w]

puts $fd [join [s pitch -method esps -framelength 0.005
        -windowlength 0.04 -maxpitch [lindex $argv 2]
        -minpitch [lindex $argv 3]] \\n]

close $fd

exit
```

Figure 7.3: Script for extracting pitch contours using Snack

In the next step, we check the forced alignment results and align the pitch contour values to the phonemes. This is done by comparing the time stamp in the forced alignment results and the index of the pitch arrays, i.e. the index of the windows. Since unvoiced phonemes have no pitch value, we remove them in our next processing step by checking if the pitch value is greater than 10.0. Suppose the utterance contains k voiced phonemes, we shall have a new sequence of pitch marks:

$$G^v = \{g^1\}, \{g^2\}, ...\{g^k\}$$

$$L^v = \{l^1\}, \{l^2\}, ...\{l^k\}$$

where $\{g^n\}$ and $\{l^n\}$ $(1 < n < k)$ are sets of pitch marks that exist in the duration of the n^{th} phoneme.

For each of the voiced phonemes, we check which one of the following pitch variation they have, taking learner's pitch marks as an example:

- pitch for this phoneme is raised: $l^n_{start} = l^n_{min} \quad and \quad l^n_{end} = l^n_{max}$;
- pitch for this phoneme is lowered: $l^n_{start} = l^n_{max} \quad and \quad l^n_{end} = l^n_{min}$;
- pitch is raised and then lowered: $l^n_{start} < l^n_{max} \quad and \quad l^n_{max} > l^n_{end}$;
- pitch is lowered and then raised: $l^n_{start} > l^n_{min} \quad and \quad l^n_{min} < l^n_{end}$;

If any of the pitch mark set is not consistent in pitch variation, we can conclude that the learner should change the pitch while pronouncing these phonemes.

We observe that pitch variation is almost always coherent with energy distribution, which means stressed syllables are usually associated with higher pitch. So we decide not to show word stress misplacement and pitch difference separately, instead, we tell the learners to fix the pitch variation in their utterance, and word stress misplacement should be corrected at the same time.

7.4 Prosodic Feedback Generation

To most learners especially beginners, simply telling them how to modify the prosody is not enough, because they have problems perceiving the prosody difference between native utterance and their own (Flege 1988). To overcome this, many researchers have suggested to use L2 learners to listen to their own voices with native prosody (Sundström et al. 1998) (Tang et al. 2001), in this way, the difference of voice quality is removed and left only the difference of prosody which is easier for learners to perceive. On the other hand, learners can imitate the correct version of their own utterances much easier than the gold standard or the teachers' voice. In our work, we provide feedback using the learners' own voice with transplanted prosody. The method of generating such feedback is based on FD-PSOLA (Moulines and Charpentier 1990b).

7.4.1 Frequency Domain Pitch Synchronous Overlap and Add (FD-PSOLA)

Among all the pitch modification and conversion methods, FD-PSOLA was proved to be the most robust since it causes the least spectral distortion. But at the same time it also requires complexer computation, involving the 3 following steps (Moulines and Charpentier 1990b):

Pitch-synchronous Analysis The goal in this step is to generate an intermediate non-parametric representation of the original speech waveform $x(n)$, which consists of a sequence of short-term signals $x_m(n)$ that are derived by multiplying the signal by a sequence of pitch-synchronous analysis windows

$h_m(n)$:

$$x_m(n) = h_m(t_m - n)x(n)$$

where t_m is the pitch mark around which the windows are centered, and the windows are always longer than a single pitch period to make sure that neighboring short term signals always involve a certain overlap.

Pitch-synchronous Modification In this step, the signal obtained from the first step is converted into a modified stream of synthesis short-term signals $\tilde{x}_q(n)$ synchronized on a new set of synthesis pitch-marks $\tilde{t}_q$. The algorithm first calculates a mapping $\tilde{t}_q \rightarrow \tilde{t}_m$ between the synthesis and analysis pitch-marks to decide which short-term signals $x_m(n)$ should be used to produce $\tilde{x}_q(n)$. Let δ_q be the sequence of delays of $\tilde{t}_q$ and t_m:

$$\delta_q = \tilde{t}_q - t_m$$

and the synthesis short-term signal is derived by applying frequency domain transformation to the translated signal:

$$\tilde{x}_q(n) = \mathcal{F}(x_m(n - \delta_q))$$

Pitch-synchronous Overlap and Add Synthesis Next, "overlap and add" is performed on the short-term signal segments for different requirements:

- To raise the pitch, the signals are moved closer together;
- To lower the pitch, the signals are moved further apart;
- To raise the duration, the corresponding signals are repeated;
- To shorter the duration, the affected signals are removed.

and transformed back to time domain:

$$\tilde{x}(n) = \frac{\sum_q \mathcal{F}^{-1}\tilde{x}_q(n)}{\sum_q \tilde{h}_q(\tilde{t}_q - n)}$$

where $\mathcal{F}^{-1}$ is the inverse frequency domain transformation and $\tilde{h}_q(n)$ donates the synthesis windows.

Besides the robustness of modification, FD-PSOLA is also known for its flexibility, which enables the combining of different transformation methods into a mixed one, for example to combine compression-expansion in the low frequencies and elimination-repetition scheme in the high frequency to preserve the phrase coherence and avoid the need of high frequency generation at the same time (Moulines and Charpentier 1990b).

7.4.2 Prosody Modification based on the Gold Standard Reference

In this section we present our method of applying the gold standard prosody of a native speaker to the learner's voice, in order to generate feedback for the learner to perceive prosody differences. The workflow of our method is shown in Figure 7.4, and can be divided in 3 steps:

7.4.2.1 Feature Extraction

In this step, we use corresponding tools to extract the necessary audio features from both the learner and the gold standard utterance, including:

- Pitch contours. They can be extracted using Snack directly from the learner and the gold standard voice.
- Duration of each phoneme in both utterances. Firstly, the text that the learner reads is parsed with MaryTTS NLP tools to generate a MaryXML file. This file is then adopted to perform forced alignment. The speech model for the alignment is trained with both native and L2 learner speech data, so that the accuracy is maintained while allowing variants in speech from the learners.

7.4.2.2 Scale factor calculation

Before any feedback is generated, we need to know if feedback is worthy, because learners, especially beginners, may skip words they don't know how to read, or add extra pause or hesitation like "*er*" and "*en*". Forced alignment won't work

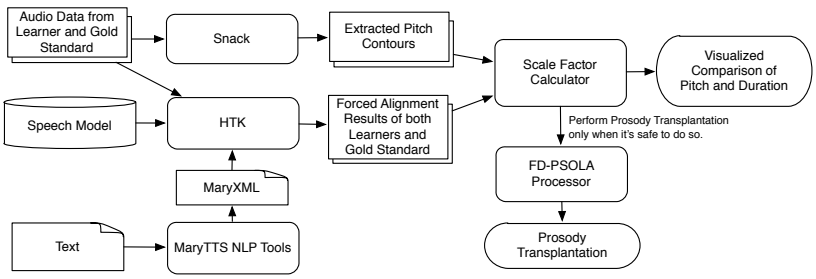


Figure 7.4: Workflow of generating feedback for prosodic errors.

precisely on such speech input, and the later time domain transformation in prosody transplantation will be jeopardized. If feedback is worth producing, we still need to decide what type of feedback is generated. In our experiment we found that if duration and pitch are to be scaled largely, it will be hard to maintain the quality of the synthesized utterance. In this case, prosody transplantation is not a proper type of feedback and the differences will only be shown visually.

To calculate the scale factors, we firstly apply a band-pass filter with lower cutoff frequency at $30Hz$ and upper cutoff at $700Hz$ to remove higher harmonics and vocal tract resonances. Since we use fixed frequencies for the filters, we preprocessed the sampling rate (f) of the audio files to 16000Hz. Then we calculate the number of *f0* candidates n_f with:

$$n_f = \frac{n_p - w_s}{s_s}$$

where n_p is the number of samples in the speech signal, w_s and s_s are the window and skip size in samples, given the window size $40ms$ and skip interval $5ms$ in time, they are calculated by:

$$w_s = 0.04 \times f$$

$$s_s = 0.005 \times f$$

Then we process the pitch contours in the windows by performing center clipping to the values, and then correct the *f0* estimations from the previous two frame voicing decision, which is the comparison result between a threshold 0.35 and

the probability of voicing p, yielded by:

$$p = \frac{max(\sum_{i=1}^{n} f_i \times f_{i+1})}{\sum_{i=1}^{n} {f_i}^2}$$

$f0$ candidates with corresponding $p < 0.35$ will be set to 0 and treated as unvoiced. Afterwards we apply both median and linear filter to smooth the validated pitch contours.

Next, we reverse the pitch contours to pitch values:

$$fp_i = \frac{fs}{f0_{i+1} - f0_i}$$

where $f0_i$ and $f0_{i+1}$ are the pitch contours of the i^{th} and its next sample, and fp_i is the derived fundamental frequency for the waveform in this window. Then, linear interpolation for the unvoiced parts is performed on the derived *f0*s by using the neighboring voiced parts. And we can determine how many new pitch values need to be inserted until sufficient voiced segments are reached, i.e. the number of pitch marks that stores the sample points where a new pulse starts.

Finally, the total number of pitch synchronous frames is calculated with:

$$N = \lfloor \frac{\frac{P_{last}}{fs} - 0.5 \times w_s}{s_s} + 0.5 \rfloor + 2$$

where P_{last} is the last value in the pitch mark sequence when it is the end of the signal, if not, an additional pitch mark is added to the last period and padded with zeros.

Duration scale factors Now we calculate the duration scale parameter for each frame. Firstly, the time stamp of the i^{th} frame in the source signal, is obtained with:

$$s_i^d = \frac{0.5 \times (P_{i+N_p} + P_i)}{fs} \quad i = 0...N-1$$

where N_p is the number pitch synchronous periods, and P_i is the value of the pitch mark. Then we compare this time stamp with the labels $(L_0, L_1, ...)$ forced alignment result to find out the index (k), which indicates to which phoneme

this frame belongs to. So the duration of this phoneme is:

$$S_i^d = L_k^s - L_{k-1}^s$$

Likewise, we have in the target signal the duration of the phoneme for the same frame:

$$T_i^d = L_j^t - L_{i-j}^t$$

To make the transformation as smooth as possible, we use triphone to calculate the duration scale factor for this frame:

$$\zeta_i^d = \frac{T_{i-1}^d + T_i^d + T_{i+1}^d}{S_{i-1}^d + S_i^d + S_{i+1}^d}$$

Pitch scale factors For this, we also need to firstly obtain the index of this frame in the source *f0* contour array and fetch the *f0* value:

$$S_i^p = f_0^s\left[\max(0, \lfloor\frac{s_i^d - 0.5 \times w_s}{s_s} + 0.5\rfloor)\right]$$

To get the pitch value in the target speech signal, we first estimate the corresponding duration of this frame with:

$$t_i^d = L_{k'-1}^t + \delta_i \times T_i^d$$

where k' is the matched index in the target labels of the k^{th} source label, and δ_i is the percentage of the phoneme duration which the i^{th} frame takes, calculated by:

$$\delta_i = \frac{s_i^d - L_{k-1}}{S_i^d}$$

and the target *f0* value can be retrieved with the same method as earlier:

$$T_i^p = f_0^t\left[\max(0, \lfloor\frac{t_i^d - 0.5 \times w_s}{s_s} + 0.5\rfloor)\right]$$

Then we have the pitch scale factor for the i^{th} frame as $\frac{T_i^p}{S_i^p}$, since we only generate the same contour as the gold standard but not the exact same voice, we reduce the scale value a bit to accept a wilder range of the learners' pitch, by multiplying the scale factor with a modification parameter ξ, which is:

- 0.7 for male and 0.8 for female, if the scale factor is greater than 1.0

- 1.3 for male and 1.2 for female, if the scale factor is less than 1.0

Also, pitch scaling is applied only to voiced frames.

$$\zeta_i^p = \begin{cases} \frac{T_i^p}{S_i^p} \times \xi, & \text{if } S_i^p \geq 10.0 \text{ and } T_i^p \geq 10.0 \\ 1.0, & \text{otherwise} \end{cases}$$

To avoid drastic scale, we limit the max and min scale factor for duration 2.5 and 0.5, and for pitch 2.0 and 0.5, hence the last step of prettifying scale factors for single frames:

$$\zeta_i'^d = \min(\max(\zeta_i^d, 0.5), 2.5)$$

$$\zeta_i'^p = \min(\max(\zeta_i^p, 0.5), 2.0)$$

Overall scale factor Now that we have the scale factors for single frames, we can estimate the overall scale factor of the whole signal, from which we could decide whether a prosody transplantation can be done without much distortion.

Since duration is linear with the frames, we can simply calculate the average duration scale factor with:

$$\zeta^d = \frac{\sum_{i=1}^{n} \zeta_i'^d}{n}$$

Here we only choose the non-silent frames (1...n) to calculate the overall duration scale factor, because learners could make very long pauses between words but shortening these won't affect the quality of the prosody transplantation. Similarly, the average pitch scale factor can be calculated with:

$$\zeta^p = \frac{\sum_{i=1}^{m} \zeta_i'^p}{m}$$

We set the following values as the thresholds for performing duration or pitch scale:

- $0.8 \leq \zeta$ or $\zeta \leq 1.2$: the difference is not so obvious, there is no need to perform scale.
- $0.6 \leq \zeta \leq 0.8$ or $1.2 \leq \zeta \leq 1.5$: the difference should be perceived by

learners after scaling, and a good quality of the synthesized voice after scaling can be guaranteed.

- $0.6 \geq \zeta$ or $\zeta \geq 1.5$: there are huge duration or pitch differences between the learners' utterance and the gold standard, and it's not suitable to perform scale.

Depending on the duration and pitch scale factor ζ^d and ζ^p, we either scale duration or pitch only, or scale them both.

7.4.2.3 Prosody Transplantation

As long as the scale factors are within the threshold values, we will perform prosody transformation. To reach out the maximized effect and provide more corrective information for the learners to perceive, we also scale the energies of each voiced window. Together with fundamental frequency, energy also plays a role in stress generation. According to Rosenberg and Hirschberg (2006), energy and pitch accent are correlated, hence it's also reasonable to modify energy in prosody transplantation to match pitch modification. In our case, energy is scaled based upon the gold standard speech, the same as duration and pitch:

$$\zeta_i^e = \frac{T_i^e}{S_i^e}$$

where S_i^e is the energy of the i^{th} voiced frame in the learner speech signal, and T_i^e is the energy of the corresponding voiced frame of the same phoneme fragment in the gold standard signal.

Our prosody transplantation method implements the FD-PSOLA algorithm described previously, and enhances it with the following aspects:

- Forced alignment is already performed prior to transplantation and a fine time-alignment between the learner and the gold standard voice signal is already made. This alignment procedure yields is more robust and efficient than traditional methods such as dynamic time warping since it fully considers the L2 pronunciation issues while training. As a result, problems like time continuity and pitch mark interpolation in transplanted speech signal can be minimized substantially. Moreover, the duration scale

factors are used to calculate the duration difference to decide if the frames are expanded or compressed, while the pitch scale factors are used to calculate the new frame size and maximized frequency of the fast Fourier transform (FFT), they both simplify the calculation, which compensates the high requirement of computation in FD-PSOLA.

- We introduce energy transformation to the whole prosody transformation, which aims at providing corresponding energy to the modified pitch to make the output speech more like native. In this way, the learners' perception to certain pitch differences are enhanced.

7.5 Visual Feedback Generation

Visual feedback is always generated for learners to observe differences in pitch and durations, no matter if prosody transplantation is performed as perceptual feedback or not. The differences shown are not simply the scale factors, but the differences in pitch or duration variations, because although we compare the learners with the gold standards only with the same gender, their fundamental frequencies may still differ a lot, which generate relative large scale factors and could mislead the learners if these factors are shown.

7.5.1 Visual Feedback for Pitch Error

Melody curve has been widely used to describe pitch contours in many softwares and applications, such as in Wavesurfer (Figure 7.2). Since users are familiar with this concept, we also use melody curves as the carrier of visual feedback for pitch error, as shown in Figure 7.5. The blue curve is for the gold standard, and the green one is for the learner's. As can be seen, the blue curve has significant raise in pitch for the word "document", while in the green one the pitch is maintained the same from the start till the end of this word. This difference is also shown in the bars above: a green bar means there is no pitch error for the phoneme that the bar represents, a red bar means there is a serious error, whereas a yellow bar means a trivial error. In Figure 7.5, the vowels in "document" are marked red, which is consistent with the melody curve difference for this word. The last phoneme, /d/ in "saved", is marked yellow, because the learner pronounced it so softly that the system didn't recognize it.

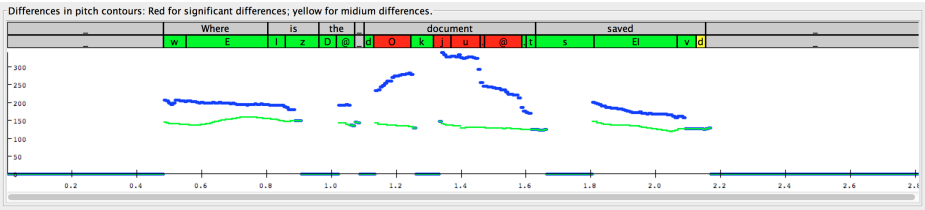


Figure 7.5: Visual feedback for pitch errors

To generate this graph, the gold standard and the learner's speech signal are aligned first, and this is already done in the preprocessor, where the duration of each phoneme in the gold standard signal is stretched or shrunk to match the one from the learner's speech signal. Then the value ϕ^p which shows whether a certain phoneme has pitch error is determined by:

$$\phi^p = |\frac{\sum_i^n \frac{P_i^t - P_{min}^t}{P_i^s - P_{min}^s}}{n} - 1|$$

where P_{min}^t and P_{min}^s are the minimal voiced pitch contours for the gold standard and the learner's signal, P_i^t and P_i^s are the i^{th} pitch contour values, and n is the total amount of window the phoneme contains.

For $0 < \phi^p < 0.3$, we conclude that there is no significant pitch variation difference, and mark the phoneme in a green bar. If $\phi^p > 0.8$, the phoneme will be marked in red, which indicates a huge pitch variation difference. In all other cases, the phoneme is marked in yellow, including a medium pitch error where $0.3 \leq \phi^p \leq 0.8$, and also the case where ϕ^p is unable to be calculated because the learner didn't pronounce the phoneme and $P_i^s < 10$ (unvoiced).

7.5.2 Visual Feedback for Duration Error

An example of visual feedback for duration difference is already shown in Figure 7.1. The duration of each phoneme in the gold standard and the learner speech signal can be retrieved directly from the forced alignment results. Since the phoneme boundaries are already known, no dynamic time warping is required. But learners speak at different speeds, especially beginners tend to speak more slowly, in this case, we don't want to tell them to simply speak faster, but

threshold value:	true positive	false positive	false negative	total	recall	precision
0.73	230	48	45	275	83.6%	82.7%
0.74	232	48	43	275	84.4%	82.9%
0.75	233	50	42	275	84.7%	82.3%

Table 7.1: Threshold values and results for prolonged duration error detection.

only to shorten those vowels like /i/, so that they don't sound like their longer versions, like /iː/. This is done by comparing the percentage of the duration of each phoneme or words in the elapse of the whole speech signal. Since L2 beginners might have problem perceiving the duration differences between longer and shorter vowels, especially those that are absent in their mother tongue (Bohn and Flege 1990), we provide two kinds of visual feedback for duration, which can be toggled by learners: either showing the duration difference in word, or in phoneme. In this way, learners are given the feedback to pronounce a word longer or shorter, in case it's monosyllable. For polysyllable words, the toggling also enables a stepwise feedback: learners first know which word should be pronounced with different duration, then down to the specific syllable or phoneme with error.

The value to decide whether a difference should be marked with a different color is derived by dividing the duration of corresponding phoneme or word in the gold standard (t_T) and the learner speech signal (t_S).

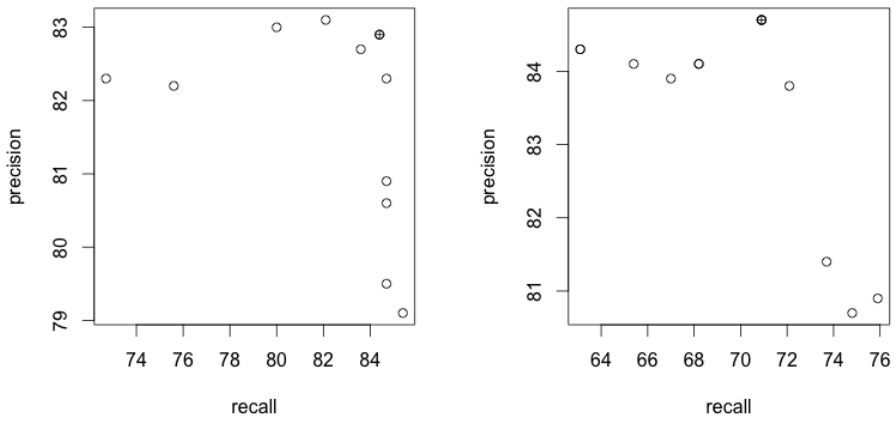
$$\phi^d = \frac{t_T}{t_S}$$

The threshold values which decide the acceptation of detected duration error are derived in the way that both precision and recall are optimal for the learner speech corpus. MaryTTS uses only short vowels in its phoneme alphabet, hence mispronouncing a vowel to its longer or shorter version won't be recognized in pronunciation error detection, although such error should be categorized to phoneme substitution, since the correct and substituted phonemes are two phonemes, like /i/ and /iː/. Because no specific training is involved in duration error detection, we run all 14 sets of learner corpus in our system and compare the result with the annotation, as shown in table 7.1 and 7.2.

The best win for the values are 0.74 for prolonged error and 1.31 for shortened

threshold value:	true positive	false positive	false negative	total	recall	precision
1.30	129	25	50	179	72.1%	83.8%
1.31	127	23	52	179	70.9%	84.7%
1.32	127	23	52	179	70.9%	84.7%
1.33	122	23	57	179	68.2%	84.1%

Table 7.2: Threshold values and results for shortened duration error detection.



(a) shortened duration error (b) prolonged duration error

Figure 7.6: Precision and recall variation for different threshold values.

error, hence the threshold are set as follows:

- $0.74 \leq \phi^d \leq 1.31$: The duration differences are acceptable, and the corresponding phonemes or words are marked green.
- $\phi^d < 0.74$: Learners' words or phonemes should be shortened, marked with orange.
- $1.31 < \phi^d$: Learners should pronounce the phoneme or word longer, marked with magenta.

The precision and recall are analyzed in the evaluation.

vowels are pronounced:	true positive	false positive	false negative	total	recall	precision
longer	232	48	43	275	84.4%	82.9%
shorter	127	23	52	179	70.9%	84.7%

Table 7.3: Statistic of duration error detection

7.6 Evaluation

Similar to phoneme errors, we also evaluate the prosodic feedback objectively and subjectively. However, pitch errors in the learner speech data weren't annotated, so the accuracy of pitch error detection was evaluated subjectively.

7.6.1 Duration Error Accuracy

Table 7.3 shows the precision and recall for prolonged and shortened duration error, with selected threshold values for optimal results from section 7.5.2.

The precisions for prolonged and shortened duration errors are similar, but the recalls differ a lot. The prolonged vowels have better recall because it's easier for annotators to decide whether a short vowel in pronounced as a long one, whereas a shortened vowel might be hard to define because the duration of vowels decrease much more than consonants in fast speech (Gay 1978), and German vowels are generally shorter than English ones (Iverson and Evans 2007), especially experienced German learners produce shorter vowels than both English native speaker and inexperienced German learners (Bohn and Flege 1992).

We assume one of the reasons why there are much more prolonged duration error than shortened is: in order for learners to easily follow, the gold standard speech has to be loud and clear, so the native speakers have to read relatively slow. And the same speed is implied to the learners and applied by them. When they speak slow, the duration of the vowels are mainly affected (Gay 1978), and a short vowel can be potentially stretched to a long one and captured by the annotators. This reminds us that we should use real dialogs as the gold standard instead of the ones being literally read by natives.

	true positive	false positive	false negative	total	recall	precision
pitch should be raised	1545	121	27	1572	98.3%	92.7%
pitch should be lowered	274	45	16	290	94.5%	85.9%

Table 7.4: Statistics for pitch error detection. Included are both significant pitch errors (marked in red in the user interface, with $\phi^p > 0.8$) and suspicious pitch errors (marked in yellow, with $0.3 \leq \phi^p \leq 0.8$)

There are still some duration errors that our method fails to detect. By reviewing them we find that the main problem is in the forced alignment step, which our method heavily depends on. When learners mispronounce certain phonemes, for example substitute /ə/ with /æ/, the forced alignment results for this phoneme and the next ones are slightly wrong. Similarly, replacing /ð/ with /d/ also causes the next vowel to have the wrong duration. To perform more accurate forced alignment, we need more learner data to train the speech model.

7.6.2 Pitch Error Accuracy

The pitch error detection results were examined by native speakers. Since no training is required, and the main work is to use Snack to extract the pitch contours and calculate the scale factors, all of 14 sets of learner data are involved in the statistics, as shown in Table 7.4.

By analyzing the results we find that there are basically two types of learners considering pitch handling in speech:

- Some learners speak in monotone, and most of them are beginners who focus on the pronunciation and neglect the pitch variation. They need to raise pitch in many parts of their utterances.
- Other learners notice pitch variation, but fail to follow the gold standard. In the utterances of these learners, problems of pitch are almost equally distributed: for some phonemes, pitch needs to be lowered, while for others, pitch needs to be raised.

Forced alignment results greatly affect the accuracy. When the durations of certain phonemes are wrong, their pitch contours will be mapped to the phonemes before and after them, and affect the average pitch contour variation and the pitch scale factor.

As mentioned earlier, phoneme errors from learners lower the accuracy too. When a phoneme is distorted or substituted with another, forced alignment results are affected; when a phoneme is inserted or deleted, it changes not only the durations of its adjacent phonemes, but also pitch scale factors if the inserted or deleted phoneme is a vowel.

7.6.3 Feedback Evaluation

In order to estimate how good our feedback can help learners to improve their prosody in language learning, we designed a progressive experiment. 4 learners are chosen to read 30 sentences from the list. They can listen to the gold standard as many times as they need, and record the speech when they are ready. If there are prosodic errors, they can view the hints or listen to the gold standard, and then try again. If the prosody is still incorrect, they can then hear the transplanted speech, as many times as they need to, and then try to record again.

	differences	correct after step 1	correct after step 2
pitch	474	343	438
duration	173	113	146

Table 7.5: Amount of detected and corrected prosody differences from test speech data. Numbers are calculated per phoneme for pitch and per word for duration. Step 1: showing visual feedback. Step 2: playing transplanted prosody as audio feedback.

The results in Table 7.5 show that most prosody differences can be perceived and adjusted by the learners, if they are given enough information. Visual feedback is helpful in pinpointing the location of the errors that learners need to pay attention to, and they are already capable of correcting most of them. Prosody transplantation as audio feedback is also proved to be working in helping learners realize their prosodic errors and fix them.

Almost all the remaining errors are from long sentences. Learners couldn't take care of all errors in one attempt. Also to be noted is that learners in the second

Nr.	Task
1	I didn't realize there are problems with my prosody in English before.
2	The prosody errors found in the experiment are indeed something I need to improve.
3	Visual feedback is very helpful.
4	I was confident in fixing all the prosody errors by viewing the visual feedback.
5	Audio feedback is necessary.
6	The quality of audio feedback is good.
7	Audio feedback helps me find out some errors that I ignored.
8	Audio feedback helps me fix errors which I still don't know how to correct even after viewing visual feedback and listening to the gold standard.
9	I was confident in fixing all the prosody errors after listening to prosody transplantation.
10	If I was given more attempts, I'm sure I can fix all the remaining errors.
11	This is the kind of feedback that I'm looking for to improve my English.

Table 7.6: Texts in the questionnaire distributed to the learners.

try could make the mistake in the original utterance again, even if they already corrected them in the first attempt. Both these facts remind us that multiple rounds of correction is necessary because some prosody errors are affected by L1 and will be neglected if learners are focusing on other parts of utterance.

To investigate how learners feel about the feedback, we designed a questionnaire and collected answers from the 4 candidates. Learners should choose from 1 (I totally disagree) to 5 (I totally agree) from a 5-Step Likert Scale to the statements in Table 7.6.

The answers from the learners led us to the following conclusions:

- Some advanced learners didn't notice that their prosody could still be improved; while the beginners knew they have issues with prosody and want to fix them. All learners agree that visual and audio feedback are very helpful.
- Visual feedback is good at pinpointing specific prosody errors but lack direction on how to correct them. Learners think they can fix them by

raising or lowering pitch, or change duration of words or phonemes. But they still can't correct all errors when they try, hence tutorials for correction are needed.

- Audio feedback is necessary. Not only because it provides a perceivable example to follow, but also reminds advanced learners of some prosody errors they ignored before.
- We believe if given more attempts so that the learners can repeat the listen-perceive-imitate iteration multiple times, they should be able to fix all prosody errors.

7.7 Conclusion

In this chapter, we described our novel method of detecting prosodic error in the learners' utterance and providing feedback at the same time, which is an extension of the earlier work presented by Ai and Xu (2015). The highlights of our approach are:

- It uses previously trained language model to process incoming speech on the fly. Analysis result and feedback can be shown to learners instantly.
- It uses only open-source components and is under GPL license itself.
- It generates both visual feedback, which pinpoints error phonemes or syllables intuitively; and audio feedback, which allows learners to perceive the prosody difference.

We use melody curve and duration bar to show prosody errors visually. Audio feedback is only shown when pitch and duration scale factors meet certain conditions, in order to make sure the prosody transplantation in audio feedback does not distort.

Thanks to a trained language model that provides forced alignment result with high accuracy, we are able to skip the dynamic time warping step and use the durations of each phoneme directly. We also adapt the traditional FD-PSOLA method to use the duration scale factors in previous steps to save on calculation

complexity. Further more, we include energy transformation in our method too, which ensures a complete transplanted prosody besides duration and pitch.

Chapter 8

Conclusion and Discussion

This thesis has presented a novel solution to the key problem in Computer Assisted Pronunciation Training (CAPT): pinpointing pronunciation or prosodic error accurately in the learners' utterances and provide corrective feedback, which can be effectively perceived. In recent decades, much research has been conducted in this special field, but solutions are still rarely seen in modern Computer Assisted Language Learning (CALL) platforms. Our method is not applied for general pronunciation error detection or prosody verification, but targets directly in CAPT applications, because two kinds of information are necessary to the system to perform precise error detection as well as generate personalized feedback, and they can only be retrieved from the learners on CALL platforms:

- Features of the learners: gender, mother tongue and L2;
- The learner's own speech data.

After deployed, the gathered L2 speech data can be used in training finer acoustic models which perform more accurate error detection, so the system will improve iteratively.

8.1 Summary

Our pronunciation error detection system implements a hybrid method which combines linguistic expertise and modern speech technologies. Pronunciation errors are classified by linguists via carefully listening to L2 speech data corpus of a specific learner-target language pair. Acoustic models are trained from the gold standard plus L2 annotated speech data. The results of error detection are directly converted to feedback, in which the learners' own speech corpus are used. Our prosody verification method compares prosodic features like accent, duration and pitch between the learner's speech and the gold standard. If an error is found, it transplants the gold standard prosody onto the learner's utterance to provide perceivable feedback for the learners.

8.1.1 Forced Alignment Adaption for L2 Pronunciation Error Recognition

Forced alignment serves as the fundamental technology in speech synthesis and recognition. It is also important in our work since we need it to compare the gold standard and the learner speech data at the phoneme level. In our work, we adapted the HTK forced alignment tool by adding new silence models. This grants the acceptation of unpredictable pauses made by L2 learners. Using the adapted tool, we also trained an acoustic model that dealt with various L2 pronunciation errors for language pairs German-English.

We created silence models that cover all kinds of pauses caused by L2 learners, including silences at the beginning and end of an utterance, silences between phrases, pauses between words and intra-segmental pauses. These silences and pauses were carefully handled in the model and merged if necessary. After adaptation, the HTK forced alignment model was able to align L2 speech data with much improved accuracy. Moreover, error phonemes no longer affected their neighbors.

In order to recognize possible pronunciation errors, we trained the acoustic model with both native and L2 speech data. Native speech data contains 1506 sentences. Among them, 96 sentences, which cover almost all phonemes and their diphone and triphone combinations, were selected and read by German learners of English at different levels. Our evaluation showed that the accuracy

of forced alignment on L2 speech data improved significantly after using the adapted model.

8.1.2 L2 Pronunciation Error Annotation

To ease the acquisition of error-annotated L2 speech data, we worked out a novel annotation tool MAT based on MaryTTS. MAT(Ai and Charfuelan 2014) is an open source software written in Java, and is designed for linguists to annotate L2 speech data with different kinds of errors, including phoneme insertion, deletion, substitution and distortion, error in syllables and whole sentence such as accent and intonation can also be annotated. MAT is highly configurable, annotators can set up which errors they want to annotate for a given speech data. Furthermore, components in MAT can also be replaced upon requirement.

Unlike other annotation tools, MAT is a standalone software with no dependency to other applications. It provides an intuitive interface for which no previous knowledge of annotation is needed. The interface allows most annotations easily done, e.g. via clicking a checkbox, so annotators can focus more on determining if certain phonemes are pronounced incorrectly. With the help of adapted forced alignment, MAT provides annotators the possibility to listen to single phoneme, syllable and word, or any selected part in an utterance. MAT is also capable of generating signal processing graphs for speech data, including waveform, spectrogram, pitch contours and energy graphs, which could help analyzing speech data.

Annotations are stored in extended MaryXML format. Errors in different speech units are added to corresponding elements in MaryXML. Certain annotating rules need to be followed to keep annotation valid and MAT checks the results before generating XML, and prompts annotators in case annotations are invalid or incomplete.

8.1.3 Bootstrapping and Supervised Self-learning Architecture for L2 Verification

Another novelty in our work is the supervised self-learning architecture for L2 speech verification. Our system is bootstrapped with both the gold standard

and L2 learners' speech data, along with their transcriptions and error annotations, from which an initial language model is trained and an error database is established.

Once the system is deployed and interacts with L2 learners, their speech errors are validated and stored in MaryXML format, the same format used for annotation. Linguists regularly open the validation results in the annotation tool, evaluate them, and update the annotations. These new annotations are used in the next iteration of training to enhance the language model for better recognition. In this way, the main components in the system, namely the annotation and validation tools, are fully employed, and the knowledge of linguists are utilized efficiently in both development and the system upgrading phase.

8.1.4 Phoneme Error Detection and Feedback Generation

One of the major contributions of our work is a novel method for phoneme error detection and feedback generation. We classified four kinds of phoneme error (insertion, deletion, substitution and distortion) from annotations of linguists. Special attention was paid to distortion, where distorted phonemes were assigned with newly created ones to represent the distortions. Acoustic models were trained using corpus handled in this way. In our experiment, verification using acoustic model trained with 10 sets of learners' speech data could already validate 4 new sets of learner data with high accuracy.

In this sense, our method combines modern speech technologies with linguist expertise. The advantages are not only to identify each phoneme error correctly and accurately, but also to provide comprehensive and perceivable feedback, which can be built directly from the error detection results. By applying an appropriate grammar to phoneme recognition and comparing its result with the correct phoneme sequence, we are able to retrieve what kinds of error has occurred in an utterance, and in which ways exactly they occurred. The hints provided by linguists on how to correct these errors will reveal themselves as textual feedback when the recognition results are mapped to the error database collected from annotations.

Beside textual instructions, e.g. "The tongue should be further back" for the word "open" or "/b/ should not be pronounced" for the word "dumb", we also

generate perceivable feedback from the learners' own utterance. This requires recordings of their speech and also progress of learning, but is feasible under the goal of creating personalized CALL application. Perceivable feedback was proved to be effective for learners unaware of or confused by the pronunciation rules. For example, the learner may pronounce the first "e" in "decorate" as /ɪ/ in "detective", for this case a textual feedback is given: "e" should sound like in "get", and at the same time, this text is synthesized by using an audio clip of "get", for which the learner has previously correctly pronounced the "e". In this way, learners not only perceive the difference between correct and incorrect pronunciations, but also strengthen the impression of the correct ones by listening to their own voices.

8.1.5 Prosodic Error Detection and Feedback Generation

A further contribution of our system is integrated prosodic error detection and feedback generation. We focused on verifying 3 main prosodic features: stress, duration and pitch. Prosodic errors were detected by comparing acoustic parameters between the learner and the gold standard utterance. Since learners speak with different speeds and tones, we applied dynamic programming to compare not the value of the parameters but their variations. Different from phoneme errors, each prosodic error has a severity, i.e. some really matter but others are trivial and play a small role in communication. To denote this, we calculated a threshold for each kind of prosodic error and categorized them accordingly, e.g. a duration error, for which a learner pronounces a vowel shorter than the gold standard, can be either a trivial or serious error, depending on how long he pronounces this vowel and other vowels in his utterance, in comparison with the durations of the same vowels in the gold standard.

Different kinds of graphs are employed to represent these errors as visual feedback to the learners. Stress errors are integrated to a phoneme error graph since they both locate on phonemes. Although duration errors are mainly related to phonemes as well, we find that learners will correct durations more easily if they are told to pronounce a word or syllable longer or shorter, rather than a vowel. Hence a bar graph is used to display duration differences. As for pitch errors, we use a melody curve since it is the clearest and most widely used way. We also show phonemes on top of the curve with different colors to represent the severity of error.

We also implemented a novel perceivable feedback via prosody transplantation, i.e. to apply the gold standard prosodies onto the learners' utterances of their own voices. The transplantation is realized using FD-PSOLA method combined with dynamic time warping. The parameters required are already calculated in a previous detection phase. To prevent distorted transplantation in cases when the learner's utterance deviates too much from the gold standard, we set thresholds and check the parameters beforehand to make sure this audio feedback is always smooth and perceivable.

8.2 Future Work

Although the system reported here is already complete and online with web interface [1], the components and methods can still be improved. Furthermore, new ideas for follow-up research have also been inspired.

8.2.1 Boosting Precision

We have shown in the evaluation that both phoneme and prosodic error detection are performed with high accuracy, however there is still potential for improvement:

- For phoneme error detection, on one hand, the acoustic model could be trained with much more speech data if available, and both the gold standard and L2 speech data will contribute to the error tolerance of the model. The model and also the system will evolve after deployment when more L2 speech data is gathered, and perform more accurate error detection. On the other hand, the detection is based on phoneme recognition using HTK. In this work, n-best is not enabled during the recognition phase, but in the next steps, we will try to execute n-best recognition and compare the log likelihood, and see if it would help eliminating false negatives.
- Prosodic error detection uses calculated acoustic parameters, which are already precise, but the comparison of determining serious, trivial or no error can still be improved. With larger sample speech data, we can

[1] LinguaTV (www.linguatv.com) provides currently online pronunciation courses using our speech verification methods.

perform more accurate statistics and hence produce better categorization of the errors.

8.2.2 Feedback Improvements

Firstly, by observing the evaluation result from learners of different levels, we conclude that it makes more sense to distinguish beginners and advanced learners, and display distortion errors only for advanced learners, since it is hard for beginners to perceive the difference between the correct phonemes and their distorted ones. Instead, they should focus on those errors they could easily recognize and fix, like deletion or substitution. Repetitive trying to make correct pronunciation without knowing exact articulation ends up discouraging learners themselves.

Another improvement on feedback: for learners that just start to use the system, there is no information about which phonemes they could pronounce error-free. In this case, words from the learner's mother tongue could also be used as example words, if they contain the target phonemes. This could be an option for advanced learners too because they know how to pronounce their native words better.

From the annotators side, hints of articulation for some substitution errors could also be given. Currently only distortions are handled this way. However, some phonemes have similar articulations and could be "distorted" into each other, e.g. /æ/ and /e/. For this case, hints such as "Mouth needs to be slightly more open" should still be provided, although they will not be used for categorizing distortion since no new phoneme needs to be created. In this way, the feedback not only indicates a different phoneme should be pronounced, but also show the exact difference between wrong and correct phonemes.

At last, there is still huge room for improvement of prosody transplantation. In order to keep the transplantation as smooth as possible, we set thresholds for parameters so that very distorted L2 speech won't be transplanted. But the ideal goal should be that learners get prosody transplantation feedback as long as there is no serious pronunciation error, such as phoneme insertion or deletion, in the utterance. The other point is that currently there are sometimes "audio gaps" in the synthesized transplantation, which do not affect feedback perception

but still can be noticed. Future work will seek to fix these gaps with modified algorithm.

8.2.3 Extensions of Application

Extra video tutorial can be prepared for particular difficult phonemes such as how to pronounce /æ/ and /e/, /əʊ/ and /ɔ/, etc. When errors with these phonemes are detected, learners can choose to watch a corresponding video to learn the pronunciation systematically. These videos target at difference between similar phonemes and hence are much more effective than normal video tutorial.

Another extension of this direction is to integrate an articulation simulation system, e.g. (Engwall et al. 2004), into the application. Since the error database already contains phoneme errors annotated with articulation, by comparing to the correct phonemes, we are able to articulation simulation of both correct and error phonemes. When learners are presented with the simulation, they will gain an even better view of the movement of the tongue, lips and mouth, than watching the videos.

Appendices

Appendix A

Phonemes in SAMPA and IPA Transcription

SAMPA	IPA	Example	Transcription
A	ɒ	pot	p A t
O	ɔ	cause	k O z
u	u	lose	l u z
i	i	ease	i z
{	æ	pat	p { t
V	ʌ	cut	k V t
E	e	pet	p E t
I	ɪ	pit	p I t
U	ʊ	put	p U t
@	ə	allow	@ ’l aU
r=	ɜ	furs	f r= z
aU	aʊ	rouse	r aU s
OI	ɔɪ	noise	n OI z
@U	əʊ	nose	n @U z
EI	eɪ	raise	r EI z
AI	aɪ	rise	r AI z
p	p	pin	p I n
b	b	bin	b I n
t	t	tin	t I n
d	d	din	d I n
k	k	kin	k I n
g	g	give	g I v
tS	ʧ	chin	tS I n
dZ	ʤ	gin	dZ I n
f	f	fin	f I n
v	v	vim	v I m
T	θ	thin	T I n
D	ð	this	D I s
s	s	sin	s I n
z	z	zing	z I N
S	ʃ	shin	S I n
Z	ʒ	measure	’m E Z r=
h	h	hit	h I t
m	m	mock	m A k
n	n	knock	n A k
N	ŋ	thing	T I N
r	r	wrong	r O N
l	l	long	l O N
w	w	wasp	w A s p
j	j	yacht	j A t

Table A.1: Phonemes in SAMPA and IPA transcriptions, with example words and exemplar transcriptions.

Bibliography

Ahmad, K. (1985). *Computers, language learning, and language teaching.* Cambridge University Press.

Ai, R. (2013). Perceptual feedback in computer assisted pronunciation training: A survey. In *Proceedings of the Student Research Workshop associated with International Conference on Recent Advances in Natural Language Processing 2013 (RANLPStud 2013)*. RANLP.

Ai, R. (2015). Automatic pronunciation error detection and feedback generation for call applications. In *International Conference on Learning and Collaboration Technologies*, pp. 175–186. Springer.

Ai, R. and M. Charfuelan (2014). MAT: a tool for l2 pronunciation errors annotation. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association.

Ai, R., M. Charfuelan, W. Kasper, T. Klüwer, H. Uszkoreit, F. Xu, S. Gasber, and P. Gienandt (2014). Sprinter: Language technologies for interactive and multimedia language learning. In *LREC*, pp. 2733–2738.

Ai, R., M. Charfuelan, W. Kasper, T. Klüwer, H. Uszkoreit, F. Xu, S. Gasber, and P. Gienandt (2014). Sprinter: Language technologies for interactive and multimedia language learning. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association.

Ai, R., S. Krause, W. Kasper, F. Xu, and H. Uszkoreit (2015). Semi-automatic generation of multiple-choice tests from mentions of semantic relations. *ACL-IJCNLP 2015*, 26.

Ai, R. and F. Xu (2015). A system demonstration of a framework for computer assisted pronunciation training. *ACL-IJCNLP 2015*, 1.

Alderman, D. L. et al. (1978). Plato and ticcit: An evaluation of cai in the community college. *Educational Technology 18*(4), 40–5.

Allen, G. D. and S. Hawkins (1980). Phonological rhythm: Definition and development. *Child phonology 1*, 227–256.

Ananthakrishnan, G., P. Wik, O. Engwall, and S. Abdou (2011). Using an ensemble of classifiers for mispronunciation feedback. In *SLaTE*, pp. 49–52.

Anumanchipalli, G. K., K. Prahallad, and A. W. Black (2011). Festvox: Tools for creation and analyses of large speech corpora. In *Workshop on Very Large Scale Phonetics Research, UPenn, Philadelphia*.

Atal, B. S. (1974). Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *the Journal of the Acoustical Society of America 55*(6), 1304–1312.

Atal, B. S. and L. R. Rabiner (1976). A pattern recognition approach to voiced-unvoiced-silence classification with applications to speech recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on 24*(3), 201–212.

Atwell, E., D. Herron, P. Howarth, R. Morton, and H. Wick (1999). Pronunciation training: Requirements and solutions. *ISLE Deliverable 1*.

Baken, R. J. and R. F. Orlikoff (2000). *Clinical measurement of speech and voice*. Cengage Learning.

Bangs, P. (2008). Introduction to call authoring programs. *Information and communications technology for language teachers (ICT4LT)*.

Barras, C., E. Geoffrois, Z. Wu, and M. Liberman (2001). Transcriber: development and use of a tool for assisting speech corpora production. *Speech Communication 33*(1), 5–22.

Bax, S. (2003). CallâĂŤpast, present and future. *System 31*(1), 13–28.

Beinborn, L., T. Zesch, and I. Gurevych (2012). Towards fine-grained readability measures for self-directed language learning. In *Electronic Conference Proceedings*, Volume 80, pp. 11–19.

Bernstein, J., A. Najmi, and F. Ehsani (1999). Subarashii: Encounters in japanese spoken language education. *CALICO journal*, 361–384.

Bigi, B. and D. Hirst (2012). SPeech Phonetization Alignment and Syllabification (SPPAS): a tool for the automatic analysis of speech prosody. In *6th International conference on Speech Prosody*, Shangai China.

Bissiri, M. P., H. R. Pfitzinger, and H. G. Tillmann (2006). Lexical stress training of german compounds for italian speakers by means of resynthesis and emphasis. In *Proc. of the 11th Australasian Int. Conf. on Speech Science and Technology (SST 2006). Auckland*, pp. 24–29.

Bitzer, D., P. Braunfeld, and W. Lichtenberger (1961). Plato: An automatic teaching device. *Education, IRE Transactions on 4*(4), 157–161.

Bohn, O.-S. and J. E. Flege (1990). Interlingual identification and the role of foreign language experience in l2 vowel perception. *Applied Psycholinguistics 11*(03), 303–328.

Bohn, O.-S. and J. E. Flege (1992). The production of new and similar vowels by adult german learners of english. *Studies in Second Language Acquisition 14*(02), 131–158.

Bonneau, A. and V. Colotte (2011). Automatic feedback for l2 prosody learning. *Speech and Language Technologies*, 55–70.

Bou-Ghazale, S. and J. Hansen (1998). Stress perturbation of neutral speech for synthesis based on hidden markov models. *IEEE Transactions on Speech and Audio Processing 6*(3), 201–216.

Bou-Ghazale, S. E. and J. H. Hansen (1997). A novel training approach for improving speech recognition under adverse stressful conditions. In *EUROSPEECH*.

Brognaux, S., S. Roekhaut, T. Drugman, and R. Beaufort (2012a). Train&Align: A new online tool for automatic phonetic alignment. In *IEEE Spoken Language Technology Workshop (SLT)*, Miami, FL, USA, pp. 416–421.

Brognaux, S., S. Roekhaut, T. Drugman, and R. Beaufort (2012b). Train&align: A new online tool for automatic phonetic alignment. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 416–421. IEEE.

Bronack, S., R. Sanders, A. Cheney, R. Riedl, J. Tashner, and N. Matzen (2008). Presence pedagogy: Teaching and learning in a 3d virtual immersive world. *International journal of teaching and learning in higher education 20*(1), 59–69.

Broselow, E., R. R. Hurtig, and C. Ringen (1987). The perception of second language prosody. *Interlanguage phonology: The acquisition of a second language sound system*, 350–361.

Brown, A. (1991). *Teaching English pronunciation: A book of readings*. Routledge.

Brown, H. D. (1987). Principles of language learning and teaching.

Bunderson, C. V. (1973). *The TICCIT project: design strategy for educational innovation*. Institute for Computer Uses in Education, Division of Instructional Services, Brigham Young University.

Burzio, L. (1994). *Principles of English stress*. Number 72. Cambridge University Press.

Byrd, D. and E. Saltzman (1998). Intragestural dynamics of multiple prosodic boundaries. *Journal of Phonetics 26*(2), 173–199.

Cabral, J. P. and L. C. Oliveira (2005). Pitch-synchronous time-scaling for prosodic and voice quality transformations. In *Proc. Interspeech*, pp. 1137–1140.

Cahn, J. E. (1990). The generation of a ect in synthesized speech. *Journal of the American Voice I/O Society 8*, 1–19.

Cairns, D. A. and J. H. Hansen (1994). Nonlinear analysis and classification of speech under stressed conditions. *The Journal of the Acoustical Society of America 96*(6), 3392–3400.

Cameron, K. (1989). *Computer assisted language learning: program structure and principles*. Intellect Books.

Carlisle, R. S. (2001). Syllable structure universals and second language acquisition. *IJES, International Journal of English Studies 1*(1), 1–19.

Charfuelan, M. (2012). Mary tts hmmbased voices for the blizzard challenge 2012. In *Blizzard Challenge Workshop*, Volume 2012.

Cho, T. (2002). *The effects of prosody on articulation in English*. Psychology Press.

Chun, D. (2013). Teaching tone and intonation with microcomputers. *CALICO Journal 7*(1), 21–46.

Cooke-Plagwitz, J. (2013). New directions in call: An objective introduction to second life. *CALICo Journal 25*(3), 547–557.

Crompton, P. and S. Rodrigues (2001). The role and nature of feedback on students learning grammar: A small scale study on the use of feedback in call in language learning. In *Proceedings of the workshop on Computer As-*

sisted Language Learning, Artificial Intelligence in Education Conference, pp. 70–82.

Cucchiarini, C., H. Strik, and L. Boves (2000). Quantitative assessment of second language learners? fluency by means of automatic speech recognition technology. *The Journal of the Acoustical Society of America 107*(2), 989–999.

Curtin, C., D. Clayton, C. Finch, D. Moor, and L. Woodruff (1972). Teaching the translation of russian by computer. *The Modern Language Journal 56*(6), 354–360.

Dansuwan, S., K. Nishina, K. Akahori, and Y. Shimizu (2001). Development and evaluation of a thai learning system on the web using natural language processing. *Calico Journal*, 67–88.

Davies, G. (2005). Computer assisted language learning: Where are we now and where are we going. In *Keynote speech at the University of Ulster Centre for Research in Applied Languages UCALL conference:âĂŸâĂŸDeveloping a pedagogy for CALL*, pp. 13–15.

Davis, R. I. and B. C. Lovell (2004). Comparing and evaluating hmm ensemble training algorithms using train and test and condition number criteria. *Formal Pattern Analysis & Applications 6*(4), 327–335.

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

De-La-Rosa, K., G. Parent, and M. Eskenazi (2010). Multimodal learning of words: A study on the use of speech synthesis to reinforce written text in l2 language learning. In *Proceedings of the Second Language Studies: Acquisition, Learning, Education and Technology*, Tokyo, Japan.

Derwing, T. M. and M. J. Munro (1997). Accent, intelligibility, and comprehensibility. *Studies in second language acquisition 19*(01), 1–16.

Derwing, T. M., M. J. Munro, and M. Carbonaro (2000). Does popular speech recognition software work with esl speech? *TESOL quarterly 34*(3), 592–603.

DeSmedt, W. (1995). Herr kommissar: An icall conversation simulator for intermediate german. *Intelligent language tutors: Theory shaping technology*, 153–174.

Dubnowski, J. J., R. W. Schafer, and L. R. Rabiner (1976). Real-time digital hardware pitch detector. *Acoustics, Speech and Signal Processing, IEEE Transactions on 24*(1), 2–8.

Eckert, J. J. P. and J. W. Mauchly (1964, February 4). Electronic numerical integrator and computer. US Patent 3,120,606.

Egbert, J., E. Hanson-Smith, and C. Chao (2007). Introduction: Foundations for teaching and learning. *CALL environments: Research, practice, and critical* (2nd), 19–28.

Egbert, J. L. and L. M. Jessup (1996). Analytic and systemic analyses of computer-supported language learning environments. *TESL-EJ 2*(2), 1–24.

Ellis, R. (2003). *Task-based language learning and teaching*. Oxford University Press.

Engwall, O. (2012). Analysis of and feedback on phonetic features in pronunciation training with a virtual teacher. *Computer Assisted Language Learning 25*(1), 37–64.

Engwall, O., O. Bälter, A.-M. Öster, and H. Kjellström (2006). Feedback management in the pronunciation training system artur. In *CHI'06 extended abstracts on Human factors in computing systems*, pp. 231–234. ACM.

Engwall, O., P. Wik, J. Beskow, and G. Granström (2004). Design strategies for a virtual language tutor. In *Proc. of ICSLP-04*, Volume 3, pp. 1693–1696.

Eskenazi, M. (1999). Using automatic speech processing for foreign language pronunciation tutoring: Some issues and a prototype. *Language learning & technology 2*(2), 62–76.

Eskenazi, M. (2009). An overview of spoken language technology for education. *Speech Communication 51*(10), 832 – 844.

Felps, D., H. Bortfeld, and R. Gutierrez-Osuna (2009). Foreign accent conversion in computer assisted pronunciation training. *Speech communication 51*(10), 920–932.

Flege, J. E. (1988). The production and perception of foreign language speech sounds. *Human communication and its disorders: A review 1*, 224–401.

Flege, J. E. (1995). Second language speech learning: Theory, findings, and

problems. *Speech Perception and Linguistic Experience: Theoretical and Methodological Issues*, 233–273.

Flege, J. E., I. R. MacKay, and D. Meador (1999). Native italian speakers' perception and production of english vowels. *The Journal of the Acoustical Society of America 106*(5), 2973–2987.

Franco, H., L. Neumeyer, V. Digalakis, and O. Ronen (2000). Combination of machine scores for automatic grading of pronunciation quality. *Speech Communication 30*(2), 121–130.

Gay, T. (1978). Effect of speaking rate on vowel formant movements. *The journal of the Acoustical society of America 63*(1), 223–230.

Gimenez de los Galanes, F., M. Savoji, and J. Pardo (1994). New algorithm for spectral smoothing and envelope modification for lp-psola synthesis. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, Volume 1, pp. I–573. IEEE.

Gimeno-Sanz, A., G. Davies, and P. Bangs (2002). Call software design and implementation.

Goddijn, S. M. and G. de Krom (1997). Evaluation of second language learners' pronunciation using hidden markov models. In *EUROSPEECH*. Citeseer.

Goldman, J. (2011). EasyAlign: an automatic phonetic alignment tool undert Praat. In *Interspeech*, Florence, Italy.

Gorman, K., J. Howell, and M. Wagner (2011). Prosodylab-aligner: A tool for forced alignment of laboratory speech. *Canadian Acoustics 39*(3), 192–193.

Grundy, W. N. (1997). Modelling biological sequences using htk. Technical report, Technical Report, prepared for Entropic Research Laboratory, Inc.

Hamada, H. and R. NAKATSU (1993). Automatic evaluation of english pronunciation based on speech recognition techniques. *IEICE TRANSACTIONS on Information and Systems 76*(3), 352–359.

Handley, Z. (2009). Is text-to-speech synthesis ready for use in computer-assisted language learning? *Speech Communication 51*(10), 906 – 919.

Hansen, J. H. (1996). Analysis and compensation of speech under stress and noise for environmental robustness in speech recognition. *Speech communication 20*(1), 151–173.

Hansen, T. K. (2006). Computer assisted pronunciation training: the four'k's of feedback. In *4th Internat. Conf. on Multimedia and Information and Communication Technologies in Education, Seville, Spain*, pp. 342–346. Citeseer.

Hart, R. S. (1995). The illinois plato foreign languages project. *CALICO journal 12*(4), 15–37.

Hauser, M. D., N. Chomsky, and W. T. Fitch (2002). The faculty of language: what is it, who has it, and how did it evolve? *science 298*(5598), 1569–1579.

Hill, J. H. (1970). Foreign accents, language acquisition, and cerebral dominance revisited. *Language Learning 20*(2), 237–248.

Hiller, S., E. Rooney, R. Vaughan, M. Eckert, J. Laver, and M. Jack (1994). An automated system for computer-aided pronunciation learning. *Computer Assisted Language Learning 7*(1), 51–63.

Holland, V. M., J. D. Kaplan, and M. A. Sabol (1999). Preliminary tests of language learning in a speech-interactive graphics microworld. *Calico Journal*, 339–359.

Huang, C., Y. Shi, J. Zhou, M. Chu, T. Wang, and E. Chang (2004). Segmental tonal modeling for phone set design in mandarin lvcsr. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*, Volume 1, pp. I–901. IEEE.

Huang, X., A. Acero, H.-W. Hon, and R. Foreword By-Reddy (2001). *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR.

Imoto, K., Y. Tsubota, A. Raux, T. Kawahara, and M. Dantsuji (2002). Modeling and automatic detection of english sentence stress for computer-assisted english prosody learning system. In *INTERSPEECH*.

Ingram, J. and J. Pittman (1987). Auditory and acoustic correlates of perceived accent change: Vietnamese schoolchildren acquiring australian english. *Journal of Phonetics*.

Ito, A., Y.-L. Lim, M. Suzuki, and S. Makino (2007). Pronunciation error detection for computer-assisted language learning system based on error rule clustering using a decision tree. *Acoustical science and technology 28*(2), 131–133.

Iverson, P. and B. G. Evans (2007). Learning english vowels with different first-language vowel systems: Perception of formant targets, formant movement, and duration. *The Journal of the Acoustical Society of America 122*(5), 2842–2854.

Jenkins, J. (2006). Global lntelligibility and local diversity: Possibility or poroolox? *English in the world: Global rules, global roles*, 32.

Jo, C.-H., T. Kawahara, S. Doshita, and M. Dantsuji (1998). Automatic pronunciation error detection and guidance for foreign language learning. In *ICSLP*.

Johnson, D. (1991). Second language and content learning with computers: Research in the role of social factors. *Computer-assisted language learning and testing: Research issues and practice 83*.

Johnson, K. E., P. Dunkel, and D. Rekart (1991). Computer-assisted english pronunciation training. In *Presentation at the Third National Conference on the Training and Employment of Graduate Teaching Assistants*.

Jones, C. and S. Fortescue (1987). *Using computers in the language classroom*. Addison-Wesley Longman Limited.

Kanters, S., C. Cucchiarini, and H. Strik (2009). The goodness of pronunciation algorithm: a detailed performance study. *SLaTE 2009*, 2–5.

Kawahara, H. (2006). STRAIGHT, exploitation of the other aspect of VOCODER: Perceptually isomorphic decomposition of speech sounds. *Acoustical Science and Technology 27*(6), 349–353.

Kelman, P. (1990). Alternatives to integrated instructional systems. *CUE Newsletter 13*(2), 7–9.

Kendrick, H. (1997). Keep them talking! a project for improving students' l2 pronunciation. *System 25*(4), 545–560.

Kim, Y., H. Franco, and L. Neumeyer (1997). Automatic pronunciation scoring of specific phone segments for language instruction. In *Eurospeech*.

Krashen, S. D. and T. D. Terrell (1983). The natural approach: Language acquisition in the classroom.

Kresan, S. (1981). Second language acquisition and second language learning.

Krüger, A. and S. Hamilton (1997). Recall: individual language tutoring through intelligent error diagnosis. *ReCALL 9*(02), 51–58.

Kuhl, P. K., K. A. Williams, F. Lacerda, K. N. Stevens, and B. Lindblom (1992). Linguistic experience alters phonetic perception in infants by 6 months of age. *Science 255*(5044), 606–608.

Lado, R. (1957). Linguistics across cultures: Applied linguistics for language teachers.

Latsch, V. L. and S. L. Netto (2011). Pitch-synchronous time alignment of speech signals for prosody transplantation. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, Rio de Janeiro, Brazil.

Lee, K.-F. (1989). *Automatic Speech Recognition: The Development of the Sphinx Recognition System*, Volume 62. Springer Science & Business Media.

Lenneberg, E. H., N. Chomsky, and O. Marx (1967). *Biological foundations of language*, Volume 68. Wiley New York.

Levy, M. (1997). *Computer-assisted language learning: Context and conceptualization*. Oxford University Press.

Lively, S. E., J. S. Logan, and D. B. Pisoni (1993). Training japanese listeners to identify english/r/and/l/. ii: The role of phonetic environment and talker variability in learning new perceptual categories. *The Journal of the Acoustical Society of America 94*(3), 1242–1255.

Lu, J., R. Wang, and L. Silva (2012). Automatic stress exaggeration by prosody modification to assist language learners perceive sentence stress. *International Journal of Speech Technology 15*(2), 87–98.

Luo, D., N. Shimomura, N. Minematsu, Y. Yamauchi, and K. Hirose (2008). Automatic pronunciation evaluation of language learners' utterances generated through shadowing. In *INTERSPEECH*, pp. 2807–2810.

Mackey, A. and J.-Y. Choi (1998). Review of tripleplayplus! english. *Language Learning & Technology 2*(1), 19–20.

Mak, B., M. Siu, M. Ng, Y.-C. Tam, Y.-C. Chan, K.-W. Chan, K.-Y. Leung, S. Ho, F.-H. Chong, J. Wong, et al. (2003). Plaser: pronunciation learning via automatic speech recognition. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pp. 23–29. Association for Computational Linguistics.

Makhoul, J., F. Kubala, T. Leek, D. Liu, L. Nguyen, R. Schwartz, and A. Srivastava (2000). Speech and language technologies for audio indexing and

retrieval. *Proceedings of the IEEE 88*(8), 1338–1353.

Markel, J. D. (1972). The sift algorithm for fundamental frequency estimation. *Audio and Electroacoustics, IEEE Transactions on 20*(5), 367–377.

Marty, F. (1981). Reflections on the use of computers in second-language acqusitionâĂŤi. *System 9*(2), 85–98.

McGilloway, S., R. Cowie, E. Douglas-Cowie, S. Gielen, M. Westerdijk, and S. Stroeve (2000). Approaching automatic recognition of emotion from voice: a rough benchmark. In *ISCA Tutorial and Research Workshop (ITRW) on Speech and Emotion*.

Meng, F., H. Meng, Z. Wu, and L. Cai (2010). Synthesizing expressive speech to convey focus using a perturbation model for computer-aided pronunciation training. In *Proceedings of the Second Language Studies: Acquisition, Learning, Education and Technology*, Tokyo, Japan.

Meng, F., Z. Wu, H. Meng, J. Jia, and L. Cai (2012). Generating emphasis from neutral speech using hierarchical perturbation model by decision tree and support vector machine. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP 2012)*, Warwik, UK.

Mennen, I. (2007). Phonological and phonetic influences in non-native intonation. *TRENDS IN LINGUISTICS STUDIES AND MONOGRAPHS 186*, 53.

Menzel, W., E. Atwell, P. Bonaventura, D. Herron, P. Howarth, R. Morton, and C. Souter (2000). The ISLE corpus of non-native spoken English. In *LREC*, Athens, Greece.

Menzel, W., D. Herron, R. Morton, D. Pezzotta, P. Bonaventura, and P. Howarth (2001). Interactive pronunciation training. *ReCALL 13*(01), 67–78.

Miller, N. (1975). Pitch detection by data reduction. *Acoustics, Speech and Signal Processing, IEEE Transactions on 23*(1), 72–79.

Miura, K.-i. (2004). Tell me more japanese beginner & advanced.

Moreno, P. J., C. F. Joerg, J.-M. Van Thong, and O. Glickman (1998). A recursive algorithm for the forced alignment of very long audio segments. In *ICSLP*, Volume 98, pp. 2711–2714.

Morley, J. (1991). The pronunciation component in teaching english to speakers of other languages. *Tesol Quarterly 25*(3), 481–520.

Morton, R. (1999). Recognition of learner speech. *Deliverable D3 3*.

Moulines, E. and F. Charpentier (1990a). Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication 9*(5-6), 453–467.

Moulines, E. and F. Charpentier (1990b). Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication 9*(5), 453–467.

Murray, J. H. (1995). Lessons learned from the athena language learning project: using natural language processing, graphics, speech processing, and interactive video for communication-based language learning. *Intelligent language tutors: Theory shaping technology*, 243–256.

Nagano, K. and K. Ozawa (1990). English speech training using voice conversion. In *First International Conference on Spoken Language Processing*.

Nagata, N. (2009). Robo-senseiâĂŹs nlp-based error detection and feedback generation. *Calico Journal 26*(3), 562–579.

Neiberg, D., G. Ananthakrishnan, and J. Gustafson (2011). Tracking pitch contours using minimum jerk trajectories. In *INTERSPEECH*, pp. 2045–2048.

Neri, A., C. Cucchiarini, and H. Strik (2006). Asr-based corrective feedback on pronunciation: does it really work? In *INTERSPEECH*.

Nwe, T. L., S. W. Foo, and L. C. De Silva (2003). Speech emotion recognition using hidden markov models. *Speech communication 41*(4), 603–623.

Paul, R., A. Augustyn, A. Klin, and F. R. Volkmar (2005). Perception and production of prosody by speakers with autism spectrum disorders. *Journal of autism and developmental disorders 35*(2), 205–220.

Peabody, M. and S. Seneff (2006). Towards automatic tone correction in non-native mandarin. In *Chinese Spoken Language Processing*, pp. 602–613. Springer.

Picard, S., G. Ananthakrishnan, P. Wik, O. Engwall, and S. Abdou (2010). Detection of specific mispronunciations using audiovisual features. In *AVSP*, pp. 7–2.

Pierrehumbert, J. and J. Hirschberg (1990). The meaning of intonational contours in the interpretation of discourse. *Intentions in communication 271*, 311.

Pollock, K. E., D. Brammer, and C. F. Hageman (1989). An acoustic analysis of young children?s productions of word stress. *Papers and Reports on Child Language Development 28*, 140–7.

Prahallad, K., A. W. Black, and R. Mosur (2006). Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, Volume 1, pp. I–I. IEEE.

Price, C., A. Bunt, and G. McCalla (1999). L2tutor: A mixed-initiative dialogue system for improving fluency. *Computer Assisted Language Learning 12*(2), 83–112.

Quang, N. H., T. Van Loan, et al. (2010). Automatic speech recognition for vietnamese using htk system. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on*, pp. 1–4. IEEE.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*(2), 257–286.

Rabiner, L. R., M. R. Sambur, and C. E. Schmidt (1975). Applications of a nonlinear smoothing algorithm to speech processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on 23*(6), 552–557.

Ramo, S. (1957). A new technique of education. *Engineering and science 21*(1), 17–22.

Roen, D. H. (1989). Developing effective assignments for second language writers. *Richness in writing*, 193–206.

Rosenberg, A. and J. Hirschberg (2006). On the correlation between energy and pitch accent in read english speech.

Rosenfelder, I., J. Fruehwald, K. Evanini, and J. Yuan (2011). Fave (forced alignment and vowel extraction) program suite. *U RL http://fave. ling. upenn. edu*.

Ross, M. J., H. L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley (1974). Average magnitude difference function pitch extractor. *Acoustics, Speech and Signal Processing, IEEE Transactions on 22*(5), 353–362.

Salomon, G. (1990). Cognitive effects with and of computer technology. *Communication research 17*(1), 26–44.

Sanders, R. H. (1995). Thirty years of computer assisted language instruction: Introduction. *Calico Journal 12*(4), 6–14.

Sarikaya, R. and J. N. Gowdy (1998). Subband based classification of speech under stress. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, Volume 1, pp. 569–572. IEEE.

Schneider, E. W. and J. L. Bennion (1983). Veni, vidi, vici via videodisc: a simulator for instructional conversations. *System 11*(1), 41–46.

Schröder, M. and S. Breuer (2004). Xml representation languages as a way of interconnecting tts modules. In *INTERSPEECH*.

Schröder, M., M. Charfuelan, S. Pammi, and I. Steiner (2011). Open source voice creation toolkit for the MARY TTS Platform. In *Interspeech*, Florence, Italy.

Schröder, M., M. Charfuelan, S. Pammi, and O. Türk (2008). The mary tts entry in the blizzard challenge 2008. In *Proc. blizzard challenge*.

Schröder, M. and A. Hunecke (2007). Creating german unit selection voices for the mary tts platform from the bits corpora. *Proc. SSW6, Bonn, Germany*.

Schröder, M. and J. Trouvain (2003). The german text-to-speech synthesis system mary: A tool for research, development and teaching. *International Journal of Speech Technology 6*(4), 365–377.

Shen, X. S. (1990). Ability of learning the prosody of an intonational language by speakers of a tonal language: Chinese speakers learning french prosody. *IRAL-International Review of Applied Linguistics in Language Teaching 28*(2), 119–134.

Sherwood, B. A. (1974). The tutor language.

Sjölander, K. and J. Beskow (2000). Wavesurfer-an open source speech tool. In *INTERSPEECH*, pp. 464–467.

Sjölander, K., J. Beskow, J. Gustafson, R. Carlson, and B. Granström (1999). Web-based educational tools for speech technology. In *MATISSE-ESCA/SOCRATES Workshop on Method and Tool Innovations for Speech Science Education*.

Skinner, B. F. (1954). The science of learning and the art of teaching. *Cambridge, Mass, USA*, 99–113.

Soria, J. (1997). Expert call: data-based versus knowledge-based interaction and feedback. *ReCALL 9*(02), 43–50.

Stepp-Greany, J. (2002). Student perceptions on language learning in a technological environment: Implications for the new millennium. *Language Learning & Technology 6*(1), 165–180.

Stouten, F. and J.-P. Martens (2006). On the use of phonological features for pronunciation scoring. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, Volume 1, pp. I–I. IEEE.

Strik, H., K. Truong, F. de Wet, and C. Cucchiarini (2009a). Comparing different approaches for automatic pronunciation error detection. *Speech Communication 51*(10), 845–852.

Strik, H., K. Truong, F. de Wet, and C. Cucchiarini (2009b). Comparing different approaches for automatic pronunciation error detection. *Speech Communication 51*(10), 845 – 852.

Strik, H., K. P. Truong, F. de Wet, and C. Cucchiarini (2007). Comparing classifiers for pronunciation error detection. In *Interspeech*, pp. 1837–1840.

Strong, J., J. Wegstein, A. Tritter, J. Olsztyn, O. Mock, and T. Steel (1958). The problem of programming communication with changing machines: a proposed solution. *Communications of the ACM 1*(8), 12–18.

Sundström, A. et al. (1998). Automatic prosody modification as a means for foreign language pronunciation training. In *Proc. ISCA Workshop on Speech Technology in Language Learning (STILL 98), Marholmen, Sweden*, pp. 49–52.

Swain, M. (1985). Communicative competence: Some roles of comprehensible input and comprehensible output in its development. *Input in second language acquisition 15*, 165–179.

Swerts, M., E. Krahmer, and C. Avesani (2002). Prosodic marking of information status in dutch and italian: A comparative analysis. *Journal of Phonetics 30*(4), 629–654.

Tang, M., C. Wang, and S. Seneff (2001). Voice transformations: from speech synthesis to mammalian vocalizations. In *INTERSPEECH*, pp. 353–356.

Titze, I. R. (2000). *Principles of voice production*. National Center for Voice and Speech.

Tokuda, N. and L. Chen (2001). An online tutoring system for language translation. *MultiMedia, IEEE 8*(3), 46–55.

Truong, K., A. Neri, C. Cucchiarini, and H. Strik (2004). Automatic pronunciation error detection: an acoustic-phonetic approach. In *InSTIL/ICALL Symposium 2004*.

Truong, K., A. Neri, F. D. Wet, C. Cucchiarini, and H. Strik (2005). Automatic detection of frequent pronunciation errors made by l2-learners. In *Interspeech*, Lisbon, Portugal.

Tsubota, Y., T. Kawahara, and M. Dantsuji (2002). Recognition and verification of english by japanese students for computer-assisted language learning system. In *INTERSPEECH*.

Uszkoreit, H. (2002). Language technology a first overview.

Valdman, A. (1976). *Introduction to French phonology and morphology*. Newbury House Pub.

Vandeventer, A. (2001). Creating a grammar checker for call by constraint relaxation: a feasibility study. *ReCALL 13*(01), 110–120.

Virvou, M. and V. Tsiriga (2001). Web passive voice tutor: An intelligent computer assisted language learning system over the www. In *Advanced Learning Technologies, 2001. Proceedings. IEEE International Conference on*, pp. 131–134. IEEE.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory 13*(2), 260–269.

Volokh, A. and G. Neumann (2012). Dependency parsing with efficient feature extraction. In *KI 2012: Advances in Artificial Intelligence*, pp. 253–256. Springer.

Vygotsky, L. S. (1980). *Mind in society: The development of higher psychological processes*. Harvard university press.

Warschauer, M. (1996). Computer-assisted language learning: An introduction. *Multimedia language teaching*, 3–20.

Warschauer, M. and D. Healey (1998). Computers and language learning: An overview. *Language teaching 31*(02), 57–71.

Wei, S., G. Hu, Y. Hu, and R.-H. Wang (2009). A new method for mispronunciation detection using support vector machine based on pronunciation space models. *Speech Communication 51*(10), 896–905.

Weigelt, L. F., S. J. Sadoff, and J. D. Miller (1990). Plosive/fricative distinction: The voiceless case. *The Journal of the Acoustical Society of America 87*(6), 2729–2737.

Welch, L. R. (2003). Hidden markov models and the baum-welch algorithm. *IEEE Information Theory Society Newsletter 53*(4), 10–13.

Wildner, S. (2013). Learn german now! version 8. *CALICO Journal 20*(1), 161–174.

Witt, S. and S. Young (1997). Computer-assisted pronunciation teaching based on automatic speech recognition. *Language Teaching and Language Technology Groningen, The Netherlands*.

Witt, S. M. (2012). Automatic error detection in pronunciation training: where we are and where we need to go. In *International Symposium on Automatic Detection of Errors in Pronunciation Training*, Stockholm, Sweden.

Witt, S. M. and S. J. Young (2000). Phone-level pronunciation scoring and assessment for interactive language learning. *Speech communication 30*(2), 95–108.

Xu, F., S. Schmeier, R. Ai, and H. Uszkoreit (2014). Yochina: Mobile multimedia and multimodal crosslingual dialogue system. In *Natural Interaction with Robots, Knowbots and Smartphones*, pp. 51–57. Springer.

Yoram, M. and K. Hirose (1996). Language training system utilizing speech modification. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, Volume 3, pp. 1449–1452 vol.3.

Young, S., G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, et al. (2002). The htk book. *Cambridge university engineering department 3*, 175.

Young, S. J. and S. Young (1993). *The HTK hidden Markov model toolkit: Design and philosophy*. Citeseer.

Zechner, K., D. Higgins, and X. Xi (2007). Speechrater: A construct-driven approach to scoring spontaneous non-native speech. *Proc. SLaTE*.

ZHANG, R. W. Y. (2012). Sentence realization with unlexicalized tree linearization grammars. In *24th International Conference on Computational Linguistics*, pp. 1301.

Zimerle, B. K. (2010). Visible language: Inventions of writing in the ancient east and beyond.