Automatic Detection Of Language Levels in L2 English Learners

Stella M. Podgornik

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2012

Committee:

Gina-Anne Levow

Michael Tjalve

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Automatic Detection Of Language Levels in L2 English Learners

Stella M. Podgornik

Chair of the Supervisory Committee:
Dr. Gina-Anne Levow, Associate Professor
Department of Linguistics

This study analyzes different features which would enable classifiers to detect language levels in adult second language (L2) English Learners. Approximately 46 different speech samples from users speaking 15 different native or L1 languages were selected from the Learning Prosody in a Foreign Language (LeaP) corpus (Gut 2004 [13]) collected in Germany. Using a variety of selected features from the spoken L2 (second language English) languages, the Support Vector Machine (SVM), was trained and the speakers were classified into three different categories: c1, c2, and s1. These categories correspond to beginner, intermediate, and advanced levels of the target secondary or L2 language, English.

The chosen features are grouped into four different categories: sentence, syllable, duration, and pitch. Count features such as sentence word count, sentence article count, etc. had the most influence on the system, while the sentence features had the second most influence. The duration features pushed the accuracy numbers into the 60s. Surprisingly, most of the pitch features used had no effect on the accuracy. A small common stop word list was also used, which proved to be very helpful. The edit distance measures of the sentences with common words removed showed a measurable effect, and the spoken duration of those same words in the sentence helped push the accuracy numbers for the test configuration above 60%. The test configuration was

selected because it had an accuracy rating close to the mean of a set of 50 randomly generated configurations. Due to the small size of the training and testing sets, it was found the L1 language of the speaker had a significant effect on the accuracy of the classification predictions. The classification predictions have a variance as much as 40%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to the University of Washington, where she has had the opportunity to work with the wonderful professors in the Linguistics Department. A special thanks to her advisor, Dr. Gina-Anne Levow for being supportive and so knowledgable. Dr. Levow was wonderful to work with and easy to talk to. Thank you to Dr. Michael Tjalve for agreeing to be a reader, even though he has a full time job outside of academia and this is a volunteer position. And last but not least, thank you to Leanne Rolston. She is a fellow Computational Linguistics student and former technical paper editor. She took time out of her busy schedule of being a student and a mother to help another student in this onerous task of thesis writing.

.

DEDICATION

To my daughter Alexa

.

Chapter 1

BACKGROUND AND INTRODUCTION

1.1 Background

Listening is a fundamental skill children are born with. Children learn their native language by listening to and absorbing all the information around them. From their first day of birth, an infant‘s speech perception is influenced by the language he or she is listening to, with the ability to discriminate sounds not used around him or her gradually disappearing during the first year of life (Gleason, 1993 [12]; Earnshaw, et. al, 1991 [4]; Werker and Tees, 1984 [39]). During that first year of life, a child‘s consistent nonword vocalizations are an attempt to convey meaning (Gleason, 1993 [12]; Halliday, 1975 [15]), imitating and practicing the sounds heard in his or her environment. Cooing, laughing, and babbling are all intentional attempts at social communication. This system of communication develops into protowords, words, then phrases, and finally full sentences. While an overt correction of language by adults has a minor influence the acquisition of language, children will gradually adjust their speech production as they mature, approximating a combination of the speech patterns produced by their parents and peers. This is how most people learn their first or native language, their L1 language (Gleason, 1993 [12]).

1.1.1 Second Language Learning Requirements

Learning a second language is usually less unconscious, and much more conscious. In order to learn a second language (an L2 language) there are three requirements (Wong–Filmore, 1991 [10]):

- a motivation and a need to learn a second language;

- access to speakers of the target language who are willing to give feedback, comment, and repetition; and,

- a social setting where the L2 learner may practice on a frequent and persistent basis.

Furthermore according to Wong-Fillmore (1991), without these three requirements, a person will experience a difficulty or even an inability in learning a second language. Both children and adults will experience these issues in trying to learn a L2 language. As in the learning a L1 language, listening is still a major aspect of these three requirements.

1.1.2 Second Language Learning Obstacles

Many L2 English learners have very little access to the second and third requirements stated by Wong–Filmore (1991). According to many different studies on second language acquisition, the length of time exposed to and using the target language has a very direct effect on proficiency in the target language. The longer a person uses and listens to the language, and the amount spent on using and listening to the language in the time frame will cause the L2 English learner to become better in the language. In this scenario, practice does help tremendously.

Unfortunately, many L2 English learners learn English via Computer Aided Language Learning (CALL) software or in class room settings that are not in areas heavily populated by native English speakers. Their social settings are usually dominated by other L2 English learners, giving them little opportunity to practice their blossoming English skills on native speakers. They hear the English words spoken incorrectly, and then repeat those same mistakes. The phonemes pronounced in English might not be available in the learners L1 language, and the ability to distinguish those unavailable phonemes disappeared during the first year of life (Gleason, 1993 [12], Earnshaw,

et. al 1991 [4], Werker and Tees, 1984 [39]). Since not all English phonemes are available in other languages (i.e. Cantonese), substitutions are made from similar phonemes which do exist in the learners L1 language and cannot be differentiated by the L2 English learners (Feng, et. al. 2008 [38]). Although these differences can not be detected by the L2 English learners, native speakers of English can detect these phonetic differences.

Cross-language phonological comparisons are difficult due to the inherent differences between language families. Some languages have articles, some do not, some languages are tonal, and some are not, and many other features. The differences between the different native L1 languages can make it hard to find similarities between the different productions of English. This is difficult, but not impossible. Similarities can be found, quantified, and used to compare the similarities and differences in different levels of L2 English learners.

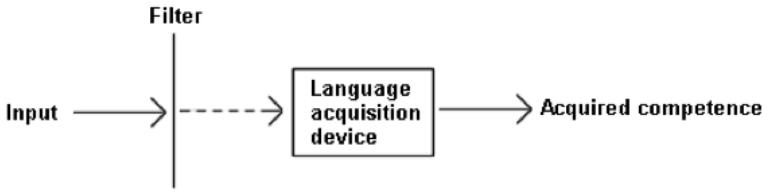

Figure 1.1: **Language Acquisition Filter**. The ”affective filter”, acts to prevent input from being used for language acquisition. (Krashen, 1987 [24])

In the Affective Filter Hypothesis made famous by Krashen, he proposed certain emotions, such as anxiety, self-doubt, and mere boredom interfere with the process of acquiring a second language. In this paper, Krashen‘s theory will not be proved or disproved, instead its proposed the L2 English learners L1 language, and their trained ears for their L1 language are their ”affective filter”. It is proposed each L2 English learner is learning English through different filters and these are impediments

to their learning another language, but nonetheless, because the non-native speakers are learning English through similar filters, it can be proposed there are similarities in the way they produce English. These similarities can be quantified and used to automatically detect their proficiency in English.

1.2 Introduction

Speech recognition techniques are used by most computerized learning systems to access the pronunciation quality for language learners word pairs or utterances. They compare the recognized spoken words with known standards of pronunciation. This project will try to detect the possible pronunciation errors in the phoneme level between the L1 and L2 speakers, and will attempt to classify the L2 speakers on their levels of English proficiency. Disfluencies, pitch, spoken sentence accuracy, edit distance comparisons with known native English speakers, word and syllable durations were used in cross-language phonological comparisons.

1.2.1 Chapter Outlines

Chapter 1 is the Background and Introduction section. It talks about how second languages are acquired, its requirements and the obstacles to learning a non-native language.

Chapter 2 is the Previous Work section.

Chapter 3 details the corpus and how the data was collected for that corpus.

Chapter 4 is the Procedures section, detailing the SVM classifier, how the dataset was selected, and the measurements. This will briefly explain the automatic and manual configurations. This chapter will also describe the testing configuration.

Chapter 5 is the Results and Analysis section. This chapter will detail the differences found between the automatic and manual configurations.

Chapter 6 is the concludes the thesis and gives a brief summary.

The appendices A–F are at the end of the thesis.

Chapter 2

PREVIOUS WORK

Previous work most closely related to detecting L2 proficiency levels in English has been in a few different areas areas, and not all of it could be utilized for this project. In the pure linguistic field of second language acquisition, Flege, Bohn and Jang (1997 [11]) looked at the effects of different L1 languages had on the vowel perception and production of English. Similar to Gut (2004 [13]), the non-native subjects were separated into different groups of experience and inexperienced groups. The aim of Flege et. al (1997 [11]) was to better understand how experience can have an affect on adults‘ production of L2 English vowels. Flege et. al (1992 [11]) looked at the perception and production of English vowels /i ɪ ɛ æ/ from L1 speakers of Spanish, Mandarin, and Korean. This study is useful because it not only looks at the relationship between the perception and the production of the English vowels /i ɪ ɛ æ/, but they also related how the speaker‘s L1 language had an effect on their production of these vowels in the English L2 language. They found the L2 English speakers ability to accurately produce and perceive the English vowels improved as they gained in experience speaking English, but their accuracy not only depended on experience but also on their L1 background. Flege et. al (1997 [11]) conducted a manual study, while this thesis project produced an automatic analysis system. This thesis project is looking at not only a larger sample size of study participants, but also a larger spread of languages.

Using a computational linguistics approach, Shriberg, et al., (2008 [35]) used different features: MLLR, phone N-gram, prosodic, and word N-gram in order to detect whether a person is an L1 or L2 speaker of English in different SVM classifiers. Maximum likelihood linear regression (MLLR) is an adaptation technique that uses small amounts of data to train a linear transform which warps the Gaussian means so as to

maximize the likelihood of the data. This study had an error rate between 12% and 20%, which was explained by the differences in native language distributions in the corpora. This particular study used data and suitable models available for American English from prior work in speech and speaker recognition. This thesis paper and subsequent study had no known English models, but the LeaP corpus (Gut, 2004 [13]) did have native English data with which to compare the non–native English data to. Shriberg, et al., (2008 [35]) proved it was possible to automatically analyse a person's speech patterns and determine their fluency as a L2 speaker of English with a decent accuracy rate. While some of the features looked at were similar, the study by Shriberg, et. al (2008 [35]) used English models for comparison, while this thesis study only had the features from both the native and non–native for comparisons. Language models are time consuming and labor intensive to create. This thesis paper does not use language models due to the manual speech transcriptions.

The work of Chen, Evanini and Sun (2010 [7]) is the closest and the most relevant to this study. They used two different studies, a pilot and then a larger scale study, in which the outputs from a classifier using vowel space characteristics: f1-f2 distance, vowel area, and vowel dispersion, were then compared with human annotation scores of the speakers English proficiency. Similar to how Chen, et. al. (2010 [7]) used the human annotation scores to rank the non–native speakers, this thesis study also uses human annotation scores to rank the non–native speakers. Chen, et. al. (2010 [7]) were able to show how the acoustic analysis of vowel space for L2 English speakers could be used in an automated analysis, and how even the highest profiency L2 English speakers had significant differences from the L1 speakers of English. In this thesis study, the final ranking of the non–native speech will also have a basis in the human annotation scores. Also, while Chen, et. al (2010 [7]) used vowel features in order to score their users, vowel features was not used in this thesis study in order to find the native and non–native speaker differences. This thesis study looks at other acoustical features which also prove highly relevant in evaluating L2 English speakers in their proficiency of English.

Higgins, Xi, Zechner, and Williamson (2011 [9]) explain their work on *SpeechRater*^{SM}, an automatic system which scores L2 English speakers in their proficiency of English. Their study built on previous work in the automated scoring of test responses for the Educational Testing Service (ETS), but Higgins et. al (2011 [9]) used highly unpredictable responses in their particular study. *SpeechRater*^{SM} uses a three-stage process, with the first stage using a filtering model to detect and only use responses which it deemed usable and able to assign a score to. Higgins et. al (2011 [9]) worked on the filtering model and proved it was feasible to develop a filtering model which was able to distinguish responses which would get good scores (1–4) from zero scores or "technical difficulty" ratings. Their study proved useful that they were able to prove using a pre-processing filter to separate "good" data from "bad" data gave them decent accuracy scores which agreed with the human scores. This thesis study will not look at free speech, but instead will look at read speech data in order to determine a L2 speaker‘s proficiency in English. Similar to Higgins et. al (2011 [9]), this thesis study will use a filtering method in the form of a common word list to differentiate a set of common words from other words in the English language.

This study will incorporate some of the findings above in a SVM classifier in order to classify the non–native English speakers proficiency level, building on the knowledge gained not only from the linguistics side of the equation but also from computational methods. The data to be used is broken down into three different categories: word lists, a read story and a free form interview. An attempt at two different baselines will be done using the information gathered from the read data. The first baseline used the read data gathered from the native English speakers, and the second baseline used the story text. Later studies will try to incorporate this baseline and try to classify the data from the free form interviews.

Chapter 3

CORPUS AND DATA COLLECTION

3.1 LeaP Corpus

The corpus being used is the LeaP corpus, collected in the LeaP project at the University of Bielefeld, Germany, from May 2001 to July 2003 (Gut 2004 [13]). It was funded from the Northrine Westphalian Ministry of Science, Education and Research. The LeaP project acquired data from non-native speakers of English and German with a concern towards the acquisition of prosody. The age of the nonnative speakers at the time of the recording ranges from 21 to 60 with a wide range in age, sex, native languages and levels of competence.

About half of the recordings were made in a sound-treated room in the University of Bielefeld, the other half was made in a quiet room, using a dat recorder and a Sennheiser microphone. All recordings were digitised on a computer with 48kHz and 16 bit.

The English learner groups include:

1. native speakers of English
2. advanced superlearners who are almost indistinguishable from the native speakers
3. learners before and after a training course
4. learners before and after going abroad in unguided learning
5. undesignated others

Table 3.1: **English learner groups.**

Four different types of speech styles were recorded.

1. nonsense word lists
2. readings of a short story (about 2 minutes)
3. retellings of the story (between 2 and 10 minutes)
4. free speech in an interview situation (between 10 and 30 minutes)

Table 3.2: **Recorded Speech Styles.**

Manual and automatic methods were used to annotate the data on 8 different tiers: pitch, tones, segments, syllables, words, phrasing, parts-of speech and lemmata. There are 359 annotated file with 131 different speakers speaking 32 different native languages. There are 18 recordings with native speakers.

	English corpus	German corpus
total number of annotated recordings	176	183
number of word lists	45	57
recordings with read speech	66	69
retellings	63	72
free speech	47	42
number of different native languages*	16	21
number of different speakers*	50	62
recordings with native speakers	8	10
recordings with superlearners	19	33
recordings with course participants	87	100
recordings with learners going abraod	38	37
other learners	24	3

Table 3.3: **LeaP Corpus Recording.** Number of recordings in different speech styles and for different speaker groups. More than 12 hours were recorded.

Not all the tiers described in Table 3.4 are in every TextGrid file. Only 4 tiers are described in the table, but there are 6 tiers used in the corpus. From bottom to the top, the tiers are labeled: phrase, words, syllable, segments, tone and pitch. The phrase tier contains 9 categories. The words tier has speech transcribed orthographically using the lower case alphabet in both German and English. The syllables are transcribed in SAMPA. SAMPA stands for Speech Assessment Methods Phonetic Alphabet, and it is a computer-readable phonetic script using 7-bit ASCII characters. For the segments tier, the syllables tier is further elaborated. All vowels and postvocalic semi-vowels are considered vowels; all plosives, fricatives, nasals, approximants, affricates, prevocalic semivowels, laterals, trills and retroflexes are considered to be consonants. The tone tier is using a modified TOBI (Tones and break indices) annotation. TOBI is a set of symbols used for transcribing and annotating the prosody of speech. The POS (part of speech) tier was annotated automatically.

The example in Figure 3.1 is an example from the Arabic speaker, and has all the tiers described in Table 3.4. All the information in this section was taken from the LeaP Corpus documation (Gut 2004 [13]). The story that was read in the English part of the corpus is in the Appendix section B.

Phrase Tier	
p	intonational phrases (of the speaker)
up	interrupted phrases
pause	unfilled pause
noise	noise
breath	breath
laughter	laughter
hp	hesitation phenomena
e	elongated phoneme
int	speech by the interviewer
Segments Tier	
V	vocalic interval
C	consonantal interval
Pause	pause
Tone Tier	
L∗, H∗, !H∗, ∧H∗, L∗+H, L+H∗, H+L∗, H∗+L, H∗+!H, H+!H∗, ∧H+H∗, ∧H∗+H	pitch accents
L%, H%, H–L%, L–H%, H-∧H%, L–, H–, !H–	boundary tones
Pitch Tier	
H	first peak of an intonational phrase
F	final low of an intonational phrase
M	intervening peak
L	intervening valley

Table 3.4: **Annotations.**

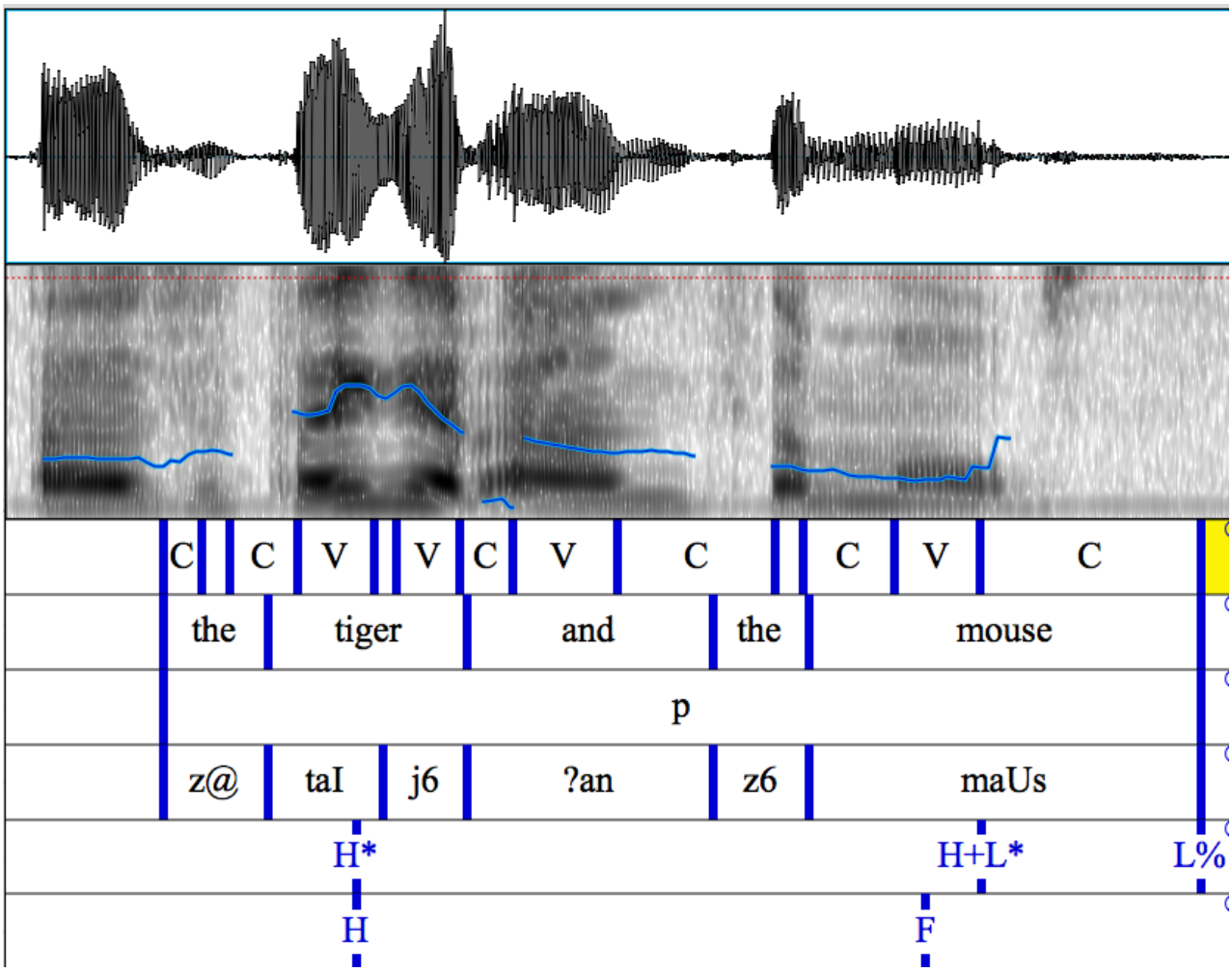

Figure 3.1: **Example Annotated Wave File in Praat**.

3.2 Data Selection

The wave and TextGrid files that were selected all had syllable and tone tiers. All the files had a words tier of transcribed speech, and all the files with a syllable and tone tier had a segments tier as well. In Figure 3.1, the example shows a few blank spots on the segments tier. The blank sections in the segments tier are annotated with a C or a V, but the Praat windows are not zoomed in at a small enough level to see the annotation because the sections are very small. Not all the files had all the tiers. Some

of the TextGrid files only had one annotation tier, some had several tiers, and some files had all the tiers displayed in Figure 3.1.

The reasons for the selections was the need for the formant, intensity and pitch information associated with the vowels, and retrieving this information from the files is very difficult without the vowel and consonant annotation. All these measurements are only taken in the vowel part of the syllable.

Chapter 4

PROCEDURES AND MEASUREMENTS

4.1 System Architecture

There were a few different steps for the overall system. This section will talk about the system architecture, the data collection process, the format of the collected data, the resulting data, and the classification process. The first step required writing a script in Praat, a vocal analysis program (Boersma and Weenink, 2011 [3]). The script can be found in Appendix E. Several modules were written in Java in order to analyze the text files produced from Praat. There were two different variations of analysing the data: one approach only analysed one test configuration and the other method analysed 50 randomly produced configurations. The one testing configuration was used in order to have a consistent, easily verifiable configuration for tests. The randomly produced configurations was used in order to find the accuracy dispersion between minimum and maximum accuracy results resulting from the different configuration. The randomly produced configurations also helped determine the testing configuration, which was selected due to its accuracy result being close to the mean of the 50 different configuration accuracies. Figure 4.1 is a pictorial representation of the overall system architecture.

4.2 Software

There were several different software suites used to compile this system. The Praat program (Boersma and Weenink, 2011 [3]) was used to analyze and print out the output for wav and TextGrid files. Praat is widely used for phonetics and speech technology research due to its ease of use and its active and helpful support group. The Praat software can be obtained at `http://www.fon.hum.uva.nl/praat/` (Boersma and Weenink, 2011 [3]). The LibSVM package (Chang, 2011 [6]) was used to clas-

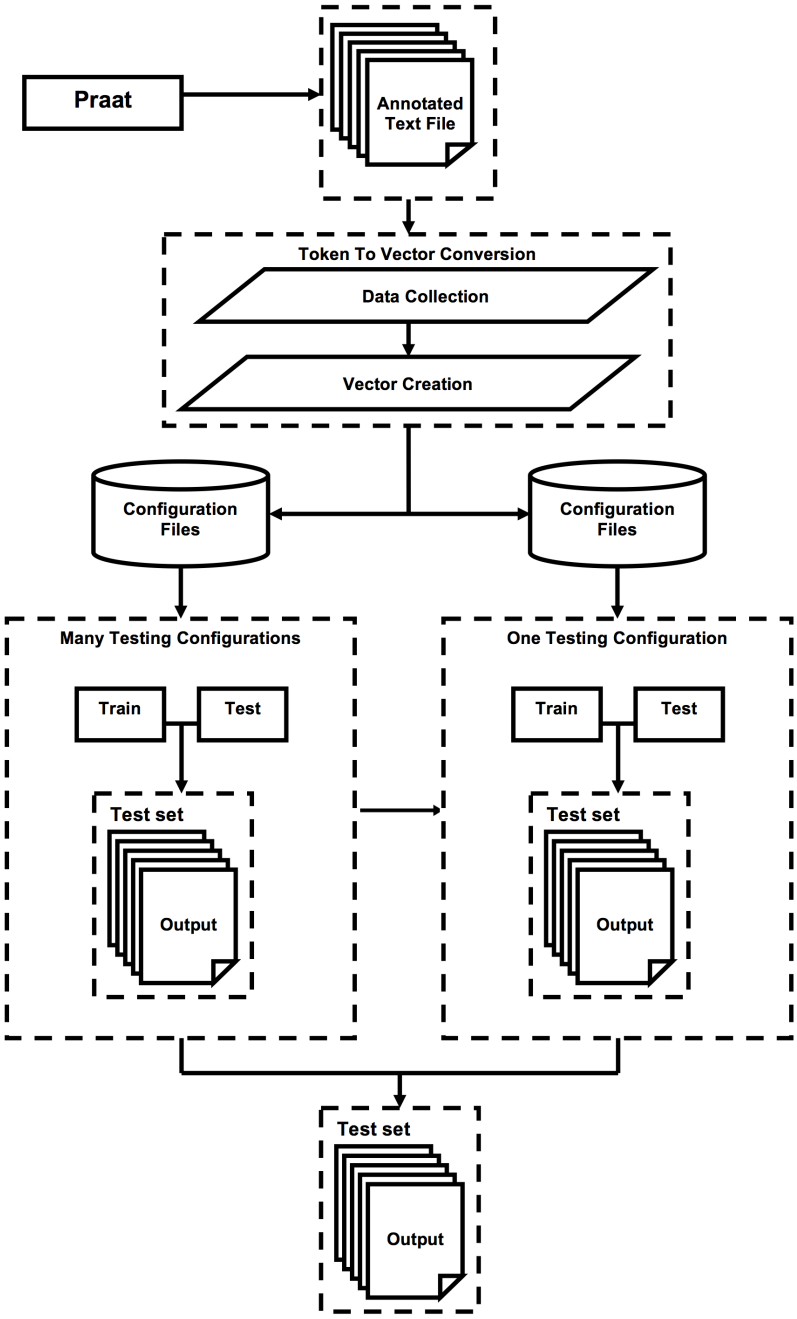

Figure 4.1: **System Design**.

sify the data, while the Java-ML (Abeel, 2009 [1]) was used as the wrapper for the LibSVM classifier. The LibSVM package is a popular implementation of the popular

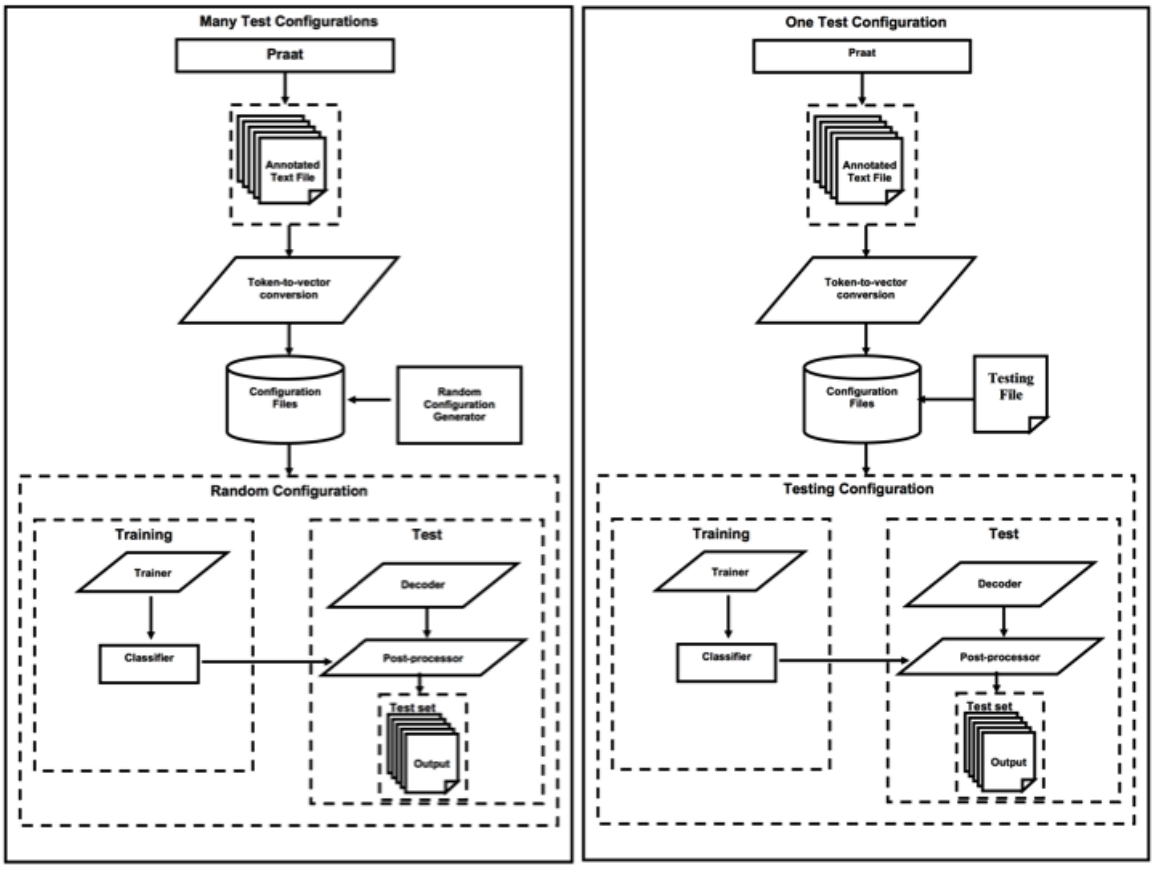

Figure 4.2: **Testing Configurations**. The many testing design produced 50 different random configurations. This was used to select a testing configuration with a mean accuracy of the different configurations, and to see the spread of the produced accuracies depending on the testing/training configuration chosen. The one testing configuration was manually selected from the 50 random variations produced by the many testing design. The one testing configuration produces an accuracy result close to the mean classification accuracy of the 50 random testing variations.

Support Vector Machine (SVM) machine learning method for classification, regression, and other machine learning tasks. It can be found at `http://www.csie.ntu.edu.tw/~cjlin/libsvm/` (Chang, 2011 [6]). The Java-ML package is a collection of machine learning and data mining algorithms. It has a simple and intuitive to use inter-

face, and good supplemental documentation and tutorials on its java api. The website to download Java-ML is `http://java-ml.sourceforge.net/` (Abeel, 2009 [1]).

4.3 SVM classification

SVMs are a set of supervised machine learning methods used for classification and regression shown to be very successful in the classification of continuous speech. SVMs minimize structural risk to find the lowest empirical error, and they will utilize the parameters outlined in Table 4.1 in order to classify or bin the data. SVMs are very capable of learning from small or large amounts of data. The SVMs will also reduce this multi–class categorization task into many binary classification tasks, and can generalize even with a large amount of features (Cristianini and Shawe–Taylor, 2000 [8]). Due to its ease of use and past success, I believe SVMs are the best classifier for this particular classification task.

Speech recognition techniques are used by most computerized learning systems to assess the pronunciation quality for language learners word pairs or utterances. They compare the recognized spoken words with known standards of pronunciation. This project will try to detect the possible pronunciation errors in the phoneme level between the L1 and L2 speakers, and will attempt to classify the L2 speakers on their levels of English proficiency. Disfluencies, pitch, spoken sentence accuracy, edit distance comparisons with known native English speakers, word and syllable durations were used in cross-language phonological comparisons.

4.3.1 A Multi-Class SVM

C–Support Vector Classifier (C–SVC)

SVMs are generally meant for 2-class problems, but this project required a multi-class solution. The users were classified into C (in this case 3) different classes in a 1-versus-rest approach. For the C classes, there will be C positive or C negative decisions. The

largest positive distance is compared against the C 1-versus-rest classifiers, and the classifier with the largest positive distance is usually chosen for the final classification (Cristianini and Shawe–Taylor, 2000 [8]). The simplest type of SVM is the classifier SVM, the C-Support Vector SVM, and this was the SVM type chosen to classify the users for this project.

Given training vectors $x_i \in R^n, i = 1, 2, ..., n$ in the two-class case and the corresponding class labels decision $y_i \in \{-1, +1\}$. the capacity constant is C, the vector of coefficients is represented by 2, b is a constant, and ξ are parameters for handling non separable data inputs. The N training cases is labeled by the index i. $y_i \in \{-1, +1\}$ is the class labels and $\mathbf{x}_i$ is the independent variables. The kernel ϕ is used to transform data from the input (independent) to the feature space. As C gets larger, the more the error is penalized. The lowest C possible should be chosen in order to avoid over fitting. Overfitting will result in higher accuracies for training data, and lower accuracies for testing data (Hsu, Chang, and Lin 2003, [18]). The C-SVC solves the following optimization problem:

$$\min_{w,\beta,\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\xi_i \tag{4.1}$$

(Hsu, Chang, and Lin 2003, [18])

subject to

$$y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq (1 - \xi_i) \tag{4.2a}$$

$$\xi_i \geq 0, i = 1, 2, ..., n \tag{4.2b}$$

(Hsu, Chang, and Lin 2003, [18])

Its dual problem is then

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \tag{4.3a}$$

$$0 \leq \alpha_i \leq C, i = 1, 2, ..., n \tag{4.3b}$$

(Hsu, Chang, and Lin 2003, [18])

subject to

$$\sum_{i:y_i=+1} \alpha_i = 0; \sum_{i=1} \alpha_i = 0 \tag{4.4}$$

(Hsu, Chang, and Lin 2003, [18])

The training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function $\phi(\cdot)$. Then the SVM finds a separating hyperplane with the maximal margin in this higher dimensional space. C is the upper bound of the error, and $K(x_i, x_j)$ is the Kernel Function that describes the behavior of the support vectors and is equal to $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$. It is this function $\Phi(x_j)$ that transforms the training data into a higher dimension where the SVs then enable classification (Hsu, Chang, and Lin 2003, [18]).

There are four basic kernels:

- linear: $K(\mathbf{x}_i \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- polynomial: $K(\mathbf{x}_i \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d$, $\gamma > 0$
- radial basis function (rbf) : $K(\mathbf{x}_i \mathbf{x}_j) = \exp(-\gamma \parallel \mathbf{x}_i - \mathbf{x}_j \parallel^2)$, $\gamma > 0$
- sigmoid : $K(\mathbf{x}_i \mathbf{x}_j) = \tanh(\gamma \parallel \mathbf{x}_i - \mathbf{x}_j + r)$

(Hsu, Chang, and Lin 2003, [18])

The exact practical steps involved in the training phase then are to :

- solve the above optimization problem given the data (x_i, y_i) and the appropriate kernel $K(x_i, y_i)$
- arrive at the appropriate weights α_i and thus evaluate the weight vector **w** and the bias *b*
- evaluate the support vector as given in the expression above.

(Hsu, Chang, and Lin 2003, [18])

The testing stage then involves the following steps:

- evaluating the sign of the decision function $f(x) = \sum_{i=1}^{S} \alpha_i y_i K(s_i, x) + b$ where S is the number of support vectors obtained from training, and s_i are those support vectors
- choosing the hypothesis based on the sign of above expression
- performing the C 1-versus-rest tests to arrive at the final hypothesis

(Hsu, Chang, and Lin 2003, [18])

Depending on which kernel function is chosen, different kinds of SVCs with different performance levels can result, and the choice of the appropriate Kernel for a specific application can be difficult. A necessary and sufficient condition for a kernel to be valid is that it must satisfy Mercer‘s Theorem, which interprets kernel products in a feature space (Cristianini and Shawe–Taylor, 2000 [8]). There is really no mathematically structured approach to prefer one Kernel over the other. There is an obvious choice, if the data is known to be not linearly separable, we would expect that a non-linear Kernel based SVC would perform better than the one based on a linear Kernel. If the data is known to be linearly separable, then it would be make sense to choose a linear Kernel.

Linear Kernel

The Kernel function results in different kinds of SVCs with different performance levels. If the data is not linearly separable, then a non-linear kernel based SVC would perform better than one based on a linear kernel. The simplest kernel is the Linear Kernel and it shows good performance for linearly separable data. It also works well in some cases of non-linear data, and in cases where the feature vector is very large.

A linear classifier is a two-class classifier that decides class membership by comparing a linear combination of features to a threshold. This classifier is a line when it is evaluated in two dimensions (Manning, 2008 [27]). Figure 4.3 is an example of a Linear classifier, very different from the non–linear classifier in Figure 4.4.

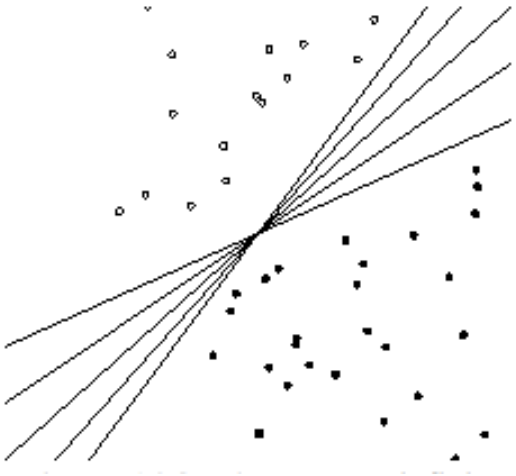

Figure 4.3: **Linear Classifier**. Two linearly separable classes can be separated by an absolute infinite amount of hyperplanes (Manning 2008 [27]).

The SVM looks for criterion in order to make a decision surface, a hyperplane, that has the maximal distance from any data point. This distance measure from the hyperplane to the closest data point determines the margin of the classifier. The decision function for an SVM is specified by a subset of data which defines the position of the

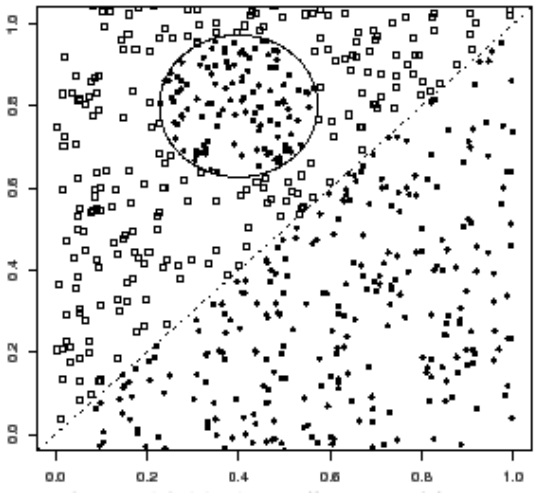

Figure 4.4: **Non-Linear Classifier**. (Manning 2008, [27])

separator. These data points are referenced as support vectors. A point can be thought of as a vector between the origin and the point (Manning 2008 [27]). Figure 4.5 shows the margin and the support vectors.

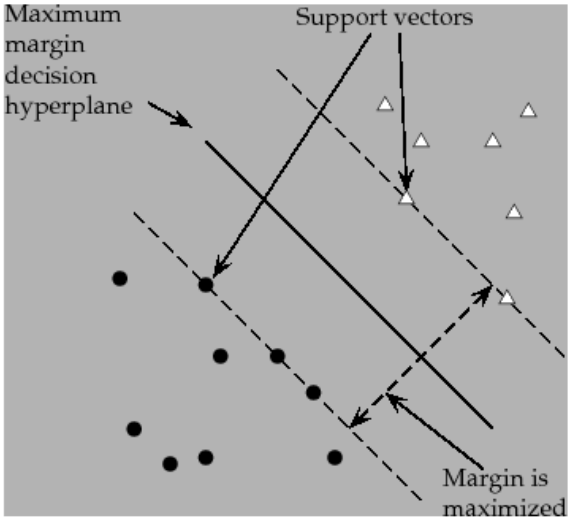

Figure 4.5: **Example Linear Support Vector**. The support vectors are the 5 points next to the classifier margin. (Manning, 2008 [27])

RBF (Radial Basis Function) Kernel

The RBF kernel nonlinearly maps data into a higher dimensional space, and it is able to handle relations between class labels and attributes that are nonlinear. The RBF kernel is a specialized type of linear kernel (Hsu, Chang, and Lin 2003, [18]) where the penalty parameter C for the linear kernel has the same performance of the RBF kernel with the parameters (C, γ). The RBF kernel usually has fewer numerical difficulties than the other kernels due to $0 < K_{ij} \leq 1$. Figure 4.4 is an example of nonlinear data. Figure 4.6 is an illustration of non–linear classification.

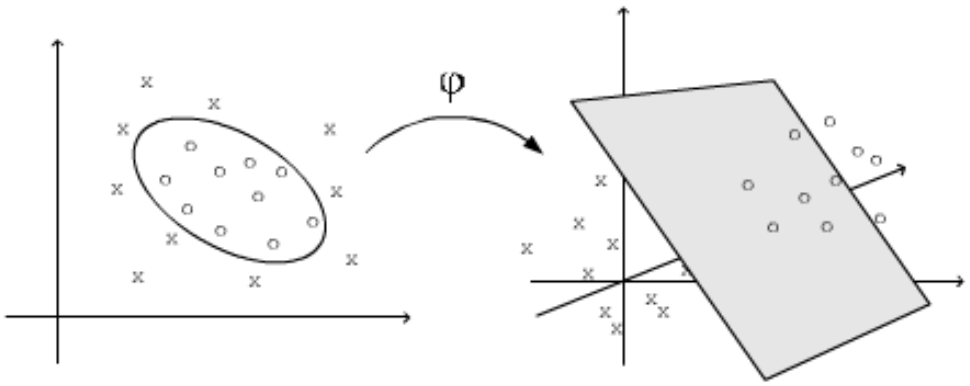

Figure 4.6: **Non-Linear Classifier**. A nonlinear separating surface in the n–dimensional is well approximated by a linear hyperplane in the N–dimensional space (Press, 2007 [31])

.

LibSVM Parameters

Table 4.1 displays the two different sets of LibSVM parameters chosen. The first column in Table 4.1 of parameters were manually chosen, and tuned to a Linear Kernel. The only parameter that needed to be tuned for the Linear kernel is C, and the value corresponding to the highest accuracy was achieved through manual trial and error.

The second column in Table 4.1 of parameters were chosen through the help of a ”grid search”, and tuned to the RBF Kernel. Following the example of Hsu, Chang, and Lin (2003, [18]), various combinations of values were tried for C, γ, and ϵ in order to find the best cross–validation accuracy. The other LibSVM parameter which was manually set is the cache_size. The default cache_size is normally 40 Mbs, but for this scenario, it was set to 256 Mbs due to the possible vector size being associated with each set of speaker data. The rest of the parameters were kept at default settings.

	Linear	RBF
param.svm_type	svm_parameter.C_SVC	svm_parameter.C_SVC
param.C	0.5	128
param.kernel_type	svm_parameter.LINEAR	svm_parameter.RBF
param.degree	3	3
param.gamma	0	3.05176e-05
param.coef0	3	3
param.nu	0.5	0.5
param.cache_size	256	256
param.eps	1e-3	0.03125
param.p	0.1	0.1
param.shrinking	1	1
param.probability	0	0
param.nr_weight	0	0

Table 4.1: LibSVM Parameters

Automatic Configuration – Grid Search

The parameters for the RBF kernel were chosen automatically using the grid search method used Hsu, Chang and Lin (2003, [18]). Table 4.1 shows these parameters

in detail. The grid search method matches the SVM dataset with different parameter values for (C, γ, and ϵ) in order to find the optimal values for that particular dataset. Hsu, Chang, and Lin only looked at the C and γ parameter values, but I modified the loop to also include ϵ. While the parameters caused over 90% accuracy on the individual dataset when doing cross–validation checks in LibSVM, they only resulted in 71% accuracy on the test dataset. Choosing parameters can be somewhat tricky in an SVM due to over fitting. When choosing parameters, several different combinations have to be tested in order to find the right ones.

	RBF
param.C	128
param.kernel_type	svm_parameter.RBF
param.gamma	3.05176e-05
param.eps	0.03125

Table 4.2: LibSVM Parameters

In the automatic configuration, a mixture of Dense and Sparse features gave the best results. The ”c1” features in the training dataset were Dense, while the other two classes were Sparse. In a DenseInstance, the whole Instance has to be specified, with no blank spots in the dataset. In a SparseInstance, only the non–zero data is stored in the dataset, while the empty spots are all set to zero. The Instances are then combined together and transformed by the SVM into the resulting dataset.

Manual Configuration

The manual configuration was less systematic and was more ”trial and error”. Due to the Linear kernel being used in the SVM, only one parameter had to be dealt with, the C parameter. After tuning the features through manual selection, different parameters were manually tested for C. C = 0.5 gave the highest results for the linear kernel. As

with the automatic configuration, a mixture of Dense and Sparse features also gave the best results. A similar method was used as in the automatic configuration, and the L1 languages with only ”c1” features were Dense with other classes and L1 languages with more than c1 class being Sparse.

4.4 Data Features Selected

4.4.1 Edit Distance Formulas

Edit Distance Formulas are also known as approximate string matching or fuzzy string searching and they are used to numerically calculate how to transform one string to another string. If the strings are either equal or both empty, the resulting distance measure will always equal Damerau–Levenshtein will always equal 0, while the Cosine Similarity algorithm will always equal 1.

Damerau–Levenshtein

The Damerau–Levenshtein distance formula is a set of elementary operations of one-character deletions, insertions, substitutions, and transpositions of adjacent characters. It is a non-weighted edit distance, where all operations have the same weight. The lower the result, the closer each string is to each other (Ukkonen, 1983 [37], Navarro, 2001 [29], Thang and Huy, 2010 [36]).

In the Damerau–Levenshtein formula for any pair of symbols α, β, from an alphabet $A \cup \{\epsilon\}$ a positive cost $\delta(\alpha \rightarrow \beta) > 0$ of a substitution of α with β. When $\alpha \rightarrow \epsilon$ is shown, a deletion of α occurred. When $\epsilon \rightarrow \beta$ is shown, a substitution of β occurred (Ukkonen, 1983 [37], Navarro, 2001 [29], Thang and Huy, 2010 [36]). In mathemati-

cal terms the formula can be described:

$$E_\delta(i+1,j+1) = min \begin{cases} E_\delta(i,j+1) + \delta(u_{i+1}) \rightarrow \epsilon \\ E_\delta(i+1,j) + \delta\epsilon \rightarrow (v_{j+1}) \\ E_\delta(i,j) + \delta(u_{i+1}) \rightarrow (v_{j+1}) \\ E_\delta(i-1,j-1) + T_{j,j+1} \end{cases} \tag{4.5}$$

(Ukkonen, 1983 [37], Navarro, 2001 [29], Thang and Huy, 2010 [36])

where $T_{j,j+1}$ equals to the cost of transposition if $u_i = v_{j+1}$ and $v_j = u_{i+1}$, and to ∞ otherwise:

$$T_{j,j+1} = \begin{cases} \delta(v_{i+1} \leftrightarrow v_{j+1}) & \text{if } u_i = v_{j+1} \text{ and } v_j = u_{i+1} \\ \infty & \text{otherwise} \end{cases} \tag{4.6}$$

(Ukkonen 1983 [37], Navarro 2001 [29], Thang and Huy, 2010 [36])

Cosine Similarity

The cosine similarity method is a little more complicated in that it measures the similarity between two vectors by measuring the cosine of the angle between those vectors. The resulting distance is 1 when the angle is 0, and it grows smaller as the angle gets larger. The cosine calculation of the angle between two vectors will therefore determine if two vectors are roughly pointing in the same direction (Han and Kamber, 2006, [16], Hazen, 2010 [17]).

The cosine of two vectors is derived from the Euclidean Dot Product

$$\mathbf{a} \cdot \mathbf{b} = \| \mathbf{a} \| \| \mathbf{b} \| \cos\theta \tag{4.7}$$

(Han and Kamber, 2006, [16], Hazen, 2010 [17])

If A and B, the vectors of attributes with the cosine similarity θ, are represented by

a dot product and magnitude, the mathematical representation is:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum_{i=1}^{N} A_i \times B_i}{\sqrt{\sum_{i=1}^{N} (A_i)^2} \times \sqrt{\sum_{i=1}^{N} (B_i)^2}} \tag{4.8}$$

(Han and Kamber, 2006, [16], Hazen, 2010 [17])

4.4.2 Sentence Features

In Table 4.3 the following features are shown for the sample utterance: the spoken sentence, the sentence text from the story, and both the edit distance measures. For both measures, the closer the numbers are to zero, the more similar the spoken and the text sentences are to each other. Each of the numbers were added to their corresponding list arrays. Each list array, and its statistical values (sum, min, max, mean, median, and standard deviation) were added to the larger array vector for each user's speech data.

spoken	a tiger and a a mouse were walking in a field when they saw a big lump of cheese lying on the ground
text	a tiger and a mouse were walking in a field when they saw a big lump of cheese lying on the ground
damlev	2.0
cossim	0.0643

Table 4.3: Testing Accuracies - Sentence

4.4.3 Syllable Features

In the syllables section, each rough phonemic transcription was compared to the comparable sentences from each of the native English speakers' phonemic transcriptions. The Cosine Similarity edit distance numbers were used to create a summation of the

distance numbers for each passage, and each L2 speaker / native English speaker combination to the other same L2 speaker / other native English combination. The lowest L2 speaker / native English speaker combination was kept for the overall speaker vector.

The following sentence is an example of the phonemic syllabic transcription of the first few words of the story. Instead of the phonetic alphabet, which is sometimes difficult to work with in Praat, the regular keyboard was used to create the transcriptions. Although the pauses were kept in the syllabic transcriptions, during the edit distance measurements the pauses are ignored. The pauses are later used during the duration feature measurements.

```
a tiger and a a mouse were walking in a field
a, taI jer, ant, a, a, maUs, wer, wO kIN, In, ?{, fiKd, pause
```

4.4.4 Duration Features

Both the duration and pitch features were normalized using the the zero normalization method first mentioned by Li et al. (1988, [25]). Variations in duration and pitch can be removed by making the scores relative to the mean and variance. $L(x_i|S)$ is a score for a given speaker S and the given feature x_i where an overall utterance is denoted by X=$\{x_i\}$ $i \in$ [1,N]. $L(x|S)$ is the raw score, while the normalized score is $L_{norm}(x_i|S)$ (Li, 1988, [25]).

$$L(x|S) = \frac{1}{N}\sum_{k=1}^{N} L(x_i|S), \tag{4.9}$$

(Li, 1988, [25])

and the normalized score

$$L_{norm}(x|S) = \frac{L(x|S) - \mu_i}{\sigma_i}, \tag{4.10}$$

(Li, 1988, [25]).

where μ_i is the mean and σ_i is the standard deviation.

4.4.5 Pitch Features

Although pitch measurements were computed at different points in the vowels by Praat, the pitch measurements used for the feature vector were all taken at the 50% point for the vowel. The pitch for each syllable, and the pitch statistics were taken into account for the final vector for each speaker dataset. During the testing phase of the experiment, individual pitch and sentence pitch features were disregarded in favor of the read passage pitch statistics. The same z-norm normalization procedure was used for both pitch and duration features.

4.5 Selecting and obtaining the data set

Praat (Boersma and Weenink, 2011 [3]) was used to extract the numerical data from the wav and TextGrid files. A Praat script was developed to extract features from each wav and associated TextGrid file, and to output the information into a text file. Each language was separated into its own folder, and each language sample had its own file in the associated language folder. The Praat script stepped through the syllable tier in the TextGrid and picked out the following information associated with each syllable unit: word label of the syllable, syllable label, start and end times, tone if it was available, pitch minimum, pitch median, pitch maximum, and the f0, f1, f2 and f3 from the onset, midpoint and glide of the component vowel. To establish a baseline, only the read speech was processed and analyzed. The free form and retold speech samples will be analyzed in a later project. The Praat script can be found in appendix E.

Table 4.5 shows the features that were extracted from the wav and TextGrid files used in making decisions on speaker L2 English proficiency. Table 4.1 shows a compilation of these features. A detailed explanation of the full feature vector can be found in appendix D. It is expected beginners will have the results in column 2 correlate to the features in column 3. The edit distance algorithms in for calculating distances between the different sentences are: Damerau–Levenshtein (Ukkonen, 1983 [37], Navarro, 2001 [29], Thang and Huy, 2010 [36]), Weighted Levenshtein (Paluncic,

et. al. 2009 [30]), and Cosine Similarity (Han and Kamber, 2006, [16], Hazen, 2010 [17]). Edit distance algorithms are applied in many fields such as language processing, speech recognition, biology computation, etc. in order to measure the distance between two strings.

The Praat (Boersma and Weenink, 2011 [3]) script stepped through the wav and TextGrid files to output the information into text files. The system then read through and arranged the information according to the read sentences (using the actual story text as the baseline), created language vectors, and ran these language vectors through LibSVM in order to classify the language according to their computed English class. The computed classes (c1, c2, and s1) were then compared against the classifications given by the human annotators in order to compute system and class accuracies.

The individual features for each language and ID combination were computed and transformed to either arrays of decimal values or individual values. For word duration, each individual word duration is calculated for each sentence and stored in an array for that sentence. If a feature only produces one number instead of an array, that particular number will represent the feature for that sentence. Whether the feature is represented by a single value or an array of values, that data is added to a larger feature vector associated with each sentence, which is then added to the overall passage vector. Each feature vector for each sentence is then added to the overall passage vector. The passage for each speaker is the summation of all their read sentences, their read passage.

The English levels were directly taken from each wav‘s associated TextGrid annotation files. The system computed results over 50 runs of different language configurations. The users being evaluated were divided into 5 groups of L2 English learners. Each group was individually compared against all other L2 English learners, and a different configuration of testing/training data was used for each run. For example, group 1 was compared to groups 2–5, group 2 was compared to groups 1 and 3–5, etc.

Feature Summary and Expected Results		
L2 Level	Expected Results	Features
c1	longer durations	Durations: word, pause, vowel syllable, common words, common syllables, syllalbes per second, pauses per second
c1	lower count	Fluent Read Word Count
c1	higher count due to possible repeats	Non-Common Word Count
c1	less variation	Similarities and Statistics In Pitch relative height
c1	higher distance	Damerau–Levenshtein of the sentences from the actual story text
c1	higher distance	Damerau–Levenshtein of the L2 transliterated syllables from the native English speakers
c1	higher distance	Weighted Levenshtein of the sentences from the actual story text
c1	higher distance	Weighted Levenshtein of the L2 transliterated syllables from the native English speakers
c1	higher distance	Cosine Similarity of the sentences from the actual story text
c1	higher distance	Cosine Similarity of the L2 transliterated syllables from the native English speakers

Table 4.4: Summary of Selected Features.

Automatic Feature Selection – Relief

When looking at a large selection of features, some features may confuse and contradict each other. An individual feature may be useless by itself but really stand out and provide good data separation when combined with another feature. The Relief method is a multi–class example of a multivariate filter, and feature ranker. It ranks subsets of features versus ranking them individually. It can reduce noise and reduce feature redundancy in order to find the optimal set of features for the highest accuracy rates (Guyon and Elisseeff, 2006 [14]). In the automatic configuration, the features were rearranged into the configuration Relief determined was the optimal configuration for data separation.

The Relief algorithm is based on the K–nearest–neighbor algorithm. The index is evaluated by first identifying the original feature space: for each example $\mathbf{x}_i$, the K closest examples of the same class $\{x_{H_k(i)}\}$, k = 1... K (nearest hits) and the K closest examples of a different class $\{x_{M_k(i)}\}$ (nearest misses). In computing the value for feature j, a summation of the distances from the examples to their nearest misses is compared to the sum of distances to their nearest hits (Guyon and Elisseeff, 2006 [14]).

$$C(j) = \frac{\sum_{i=1}^{m} \sum_{k=1}^{K} |x_{ij} - x_{M_k(i),j}|}{\sum_{i=1}^{m} \sum_{k=1}^{K} |x_{ij} - x_{H_k(i),j}|} \tag{4.11}$$

(Kira and Rendell, 1992 [23])

The features in Table 4.5 are the top 20 features important to Relief. The Rank column describes the features rank according to Relief, and the Feature No. column is the number of the feature in the full feature vector found in Appendix D. The full feature vector of the Relief ordered features can also be found in Appendix D.

Top 22 Relief Ordered Feature Vector		
Rank	Feature No.	Feature
1–2	3–4	Damerau–Levenshtein distances between sentences 3-4
3	156	Maximum word duration in sentence 7
4	133	Minimum word duration in sentence 5
5	144	Maximum word duration in sentence 6
6	123	Mediam word duration in sentence 4
7	11	Cosine Similarity distance in sentence 4
8	85	Minimum word duration in sentence 1
9	86	Mean word duration in sentence 1
10	47	Damerau–Levenshtein distance in syllablic transcription in sentence 1
11	8	Cosine Similarity distance in Sentence 1
12	45	Article count per each read passage
13	126	Maximum non–common word duration in sentence 4
14	9	Cosine Similarity distance in sentence 2
15	142	Summation of the non–common word durations sentence 5
16	94	Summation of the non–common word durations in sentence 1
17	10	Cosine Similarity distance in sentence 1
18	53	Cosine Similarity distance in syllabic transcription of sentence 7
19	82	Standard Deviation of the pitch per passage
20	102	Maximum common word duration in sentence 2
21	49	Damerau–Levenshtein distance in syllablic transcription in sentence 3
22	121	Minimum common word duration in sentence 4

Table 4.5: **Top 22 Relief Ordered Features**. A full breakdown of the feature vector and the Relief order feature vector can be found in Appendix D. The feature number in the table corresponds to the feature number from the fully featured SVM vector.

Manual Feature Selection

The features selected manually were selected through trying many different configurations, a ”trial and error” approach. At first all the features were tried, and one by one, each feature array was added or discarded in trying to reach an optimal set of features with the highest accuracy rates. Unlike Relief, which chose the features individually, the manual selection method chose the features in groups. Each set of features were grouped with other sets of features which were similar. Table 4.6 has a full breakdown of the features selected manually.

The Rank column describes the features rank in the manually ordered feature vector, and the Feature No. column is the feature number in the full feature vector found in Appendix D. The numbers the Feature No. column in Table 4.5 correspond to the same feature number and same feature in Table 4.6. A copy of this table can also be found in Appendix D in order to easily compare the manual and the relief ordered full feature vectors.

Manually Ordered Feature Vector		
Rank	Feature No.	Feature
1–7	27–33	Damerau–Levenshtein distances between sentences 1–7 without the common words
8–14	34-40	Cosine Similarity distance in sentence 1–7 without the common words
15	41	Ratio of Out of Vocabulary word to In Vocabulary words per passage
16	46	A count of all the common word syllables per passage
17-23	47 – 53	Damerau–Levenshtein distances between syllabic transcriptions 1–7 and the best scored native English speaker

24 – 29	77 – 82	Pitch statistics for the words per passage: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
30	83	Numbers of pitches per passage
30–65	90–167 *	Word Durations per sentence: maximum, minimum, mean, and median Common Word durations per sentence: summation and standard deviation **every other line between 90–167 in the full feature vector*
66–72	168–174	Common word count per sentence
73	175	Average common word duration per passage
74–125	181–251*	Duration statistics for the syllables per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation **every other line between 181–251 in the full feature vector*
126	252	Maximum pause duration per passage
127	252	Minimum pause duration per passage
128	256	Summation of all the pauses per passage
129	257	Standard deviation of all the pauses per passage
130	258	Common word count for the passage

Table 4.6: **Full Manually Selected Feature Set**. These features are not the full feature set due to the manual selection and elimination process. Some features in the automatic selection process were eliminated in the manual process. The feature number in the table corresponds to the feature number from the fully featured SVM vector.

4.6 Experimental Configuration

To create a baseline for later use, only the read data was evaluated. Each speaker‘s file was compared against the transcript they read. Five groups of different configurations of 80% of the non-native English data were used as training data and the remaining 20% of the non native English data were used as the testing data. While the actual story text was used as the baseline comparison for the recognized text, the rough phonemic transcription of the native English speakers were used as a baseline of comparison for the L2 English speakers.

Fifty runs of the system were conducted to establish statistics, 5–fold cross validation tests without replacement, and to get a testing configuration. The selected configuration for testing had an accuracy percentage close to the mean of the 50 runs. Each of the fifty runs, except for the selected testing configuration, had a different configuration. In total, 51 different configurations were tested during each system run, with the selected test configuration used consistently during each run to get directly comparable results. The table of the experimental configuration can be found in Appendix C.

During each run, groups 1-5 were tested against all the other combined groups. Group 6 was always used in the training dataset, while each consecutive group 1-5 were each tested against the other groups in the resulting training dataset. The user designated with an ”ot”, meaning the person was not manually classified by the people collecting the data, were not used in any of the training datasets. In total there were 46 data samples used, and the breakdown of the data is as follows: 39 were used in different training configurations, 7 were classified as others and always used in a testing configuration, and there were 15 different L1 languages. All the super-learners were German L1 speakers, and around 62% were classified in the ”c1” or the beginner category for all the L1 languages.

L2 Language Speakers		
L1 Language	Classes	Total
Arabic	c1	1
Chinese	c1	1
English	na	3
French	c1	1
German	c1	11
	c2	4
	s1	7
	ot	1
Hungarian	c1	2
	c2	2
Ibibio	ot	1
Igbo	ot	1
Korean	c1	2
	ot	1
Persian	c1	1
Russian	c1	3
Spanish	ot	2
Yoruba	ot	1

Table 4.7: **L2 Language Speakers**. All the e1 classifications were combined with the c1s, and all the c3s were combined with the c2s, in order to result in 3 possible classes: c1, c2, s1. The na class was only used for training and combined with the s1 training classes. The ot classes were not used in training, but only in testing for further evaluation at another time.

Chapter 5

RESULTS AND ANALYSIS

In the Results chapter, the test configuration used in each of the automatic and manual sections refers to the experimental configuration described in Chapter 4. The full testing configuration can be found in Appendix C. The same testing configuration dataset was used for both the automatic and manually configured tests.

In the Relief ordered, automatic configuration, all the features were used in an optimal configuration pattern found by Relief. The automatic configuration has a higher number of features than the manual configuration in order to achieve the same accuracy results. The top 20 features for the automatic configuration can be found in Table 4.5, and the full breakdown of the feature vector can be found in Appendix D. The experimental testing configuration used the features as dictated by Relief. In the manual configuration, the missing features in Table 4.6, as described in the full feature vector found in Appendix D, were added onto the end of the feature vector. Table 4.6 describes the first 130 features used and how they correspond to the features in the full feature vector.

5.1 Automatic Configuration

5.1.1 Relief Ordered, Test Configuration

Each feature is cumulative, and the accuracy results are achieved at each feature is due to all the features preceding it added into the feature vector. For example, in order to achieve an overall accuracy rating of 71% at automatic feature 138, the features had to be ordered 1–137 preceding feature 138 in the configuration dictated by Relief. Figure 5.1 shows the accuracy results that resulted from incrementally increasing through the Relief ranked and ordered features. For the first set of features 1–10,

there was not enough information for the SVM to make a decision, so it defaulted to the result of the largest class, c1, at 62%. Features 11–60 only confused the SVM, which resulted in accuracy numbers lower than the largest class, although the numbers steadily improved until they again reached the default at 62%. The accuracy numbers rose and fell around the default class accuracy until it reached its highest value at 71%, at features 138–146. Between 146 until the last feature, the accuracy remained steady at 68%. All the numbers in Figure 5.1 are for the test configuration.

The features which gave the highest accuracy to the test configuration were the common word durations for sentence five, ”But the tiger put his paw on the cheese and said: ”It‘s mine!”. The maximum, minimum, mean, median, sum, and standard deviation for the common word durations in sentence five pushed the test configuration to an accuracy rate of 71%. The common words in sentence five were: but, the, his, on, and, said, and its.

5.1.2 Class Accuracies

Both classes, c1 and s1, have positive relationships with the overall test accuracy. As each one set of accuracies go up, they all go up. Class c2 is not featured in the chart due to its constant value of ”0”. Classes c1 and s1 have a very minor negative correlation (the linear dependance between two variables X and Y) to each other. Relief ordered the features into an optimal configuration for class separation, and it found the optimal configuration for class separation between c1 and s1, while being unable to detect c2. Relief finds the optimal features for data separation, and will disregard the features which are not optimal. Due to the classifier being unable to detect c2, it is assumed the features needed to detect c2 are not optimal.

Instead of displaying a chart with the individual lines comparing the classes with the overall accuracy, Table 5.1 is neater to display. The Relief ordered automatic configuration found good data separation between the lowest and the highest class. The

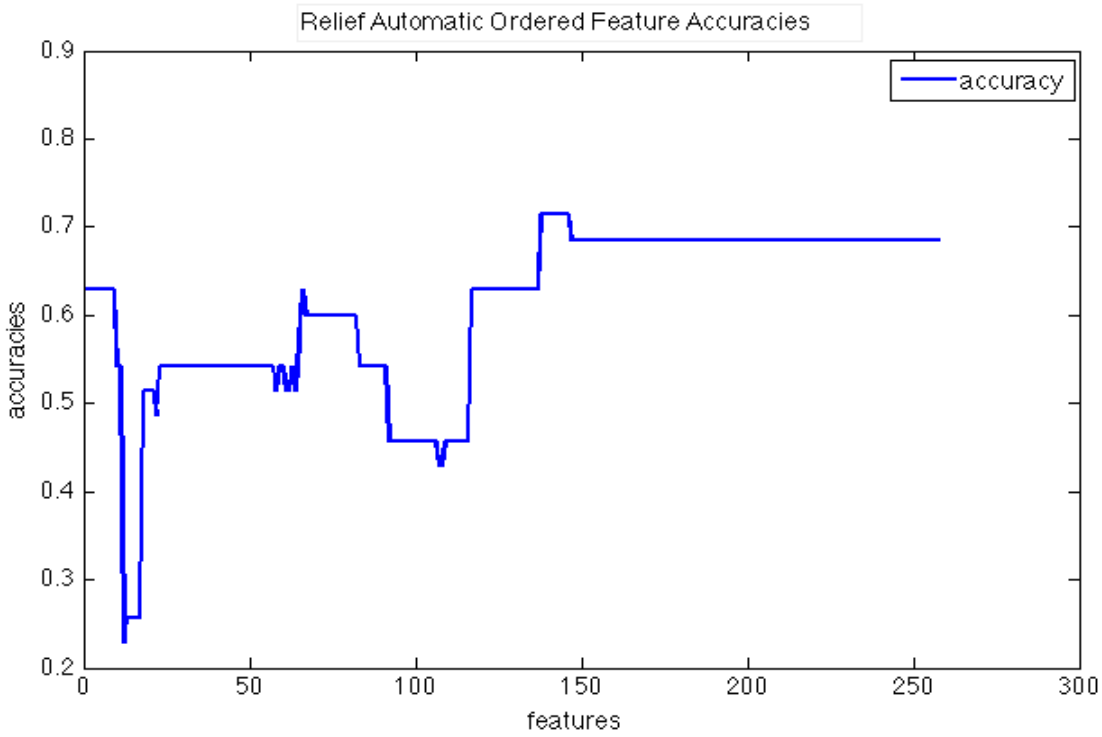

Figure 5.1: **Relief Ordered Test Accuracy Chart**. The features on the x–axis correspond to the Rank column in Table 4.5.

Relief ordered configuration also helped the overall accuracy.

The correlations between the classes and the overall test accuracy were computed using the Pearson product–moment correlation coefficient, also known as PPMCC or PCC. The formula measures the correlation or a relation, and gives a value between +1 and -1, and is a measure of strength of the reliance between two variables. It is referenced as ”Pearson‘s *r*”, the same letter used to denote the *sample correlation coefficient*. The *sample correlation coefficient* is used where the sample size is finite, versus a *population correlation coefficient* that is used as a representative for a whole

population and is not finite.

$$r = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \bar{X})^2} \times \sqrt{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}} \tag{5.1}$$

(Rodgers and Nicewander 1988, [32])

Accuracies	Correlations
overall vs c1	0.9178
overall vs s1	0.5263
c1 vs s1	0.1456

Table 5.1: **Relief Ordered – Class Correlations**. Both c1 and s1 helped the overall testing configuration accuracy go up, as the overall accuracy improved, so did c1 and s1 individually. C1 and S1 had a very minor to almost no correlation to each other in this configuration.

In Table 5.1 it is obvious there is a strong correlation between the overall system accuracy and the accuracy for the c1 class. There is a minor correlation for the s1 class to the overall system accuracy. This does show the effectiveness of the Relief algorithm in being able to find an optimal configuration for data separation. Table 5.1 also shows how the features for both the lowest and the highest class in the automatic Relief-ordered configuration had almost no relationship to each other in order to achieve the highest possible system accuracy. The table also shows the highest effect on the overall system accuracy

5.2 Manual Configuration

5.2.1 Manually Ordered Test Configuration

As with the Relief ordered test configuration, each feature in the manually ordered test configuration is also cumulative, and the accuracy results are achieved at each

feature is due to all the features preceding it added into the feature vector. Figure 5.2 shows the accuracy results that resulted from incrementally increasing through the manually ordered features. For the first set of features 1–18, the SVM did not have enough information in order to make a decision, so it defaulted to the result of the largest class, c1, at 62%. Features 11–45 confused the SVM, which resulted in accuracy numbers lower than the largest class, and then improved until they again reached the default at 62%. The accuracy numbers rose and fell around the default class accuracy until it reached its highest value at 71%, at feature 75. All the numbers in Figure 5.2 are for the test configuration. Unlike the Relief ordered and ranked features, the accuracy numbers for the manual configuration are more unstable, and fall below the default value more often than it reaches above it.

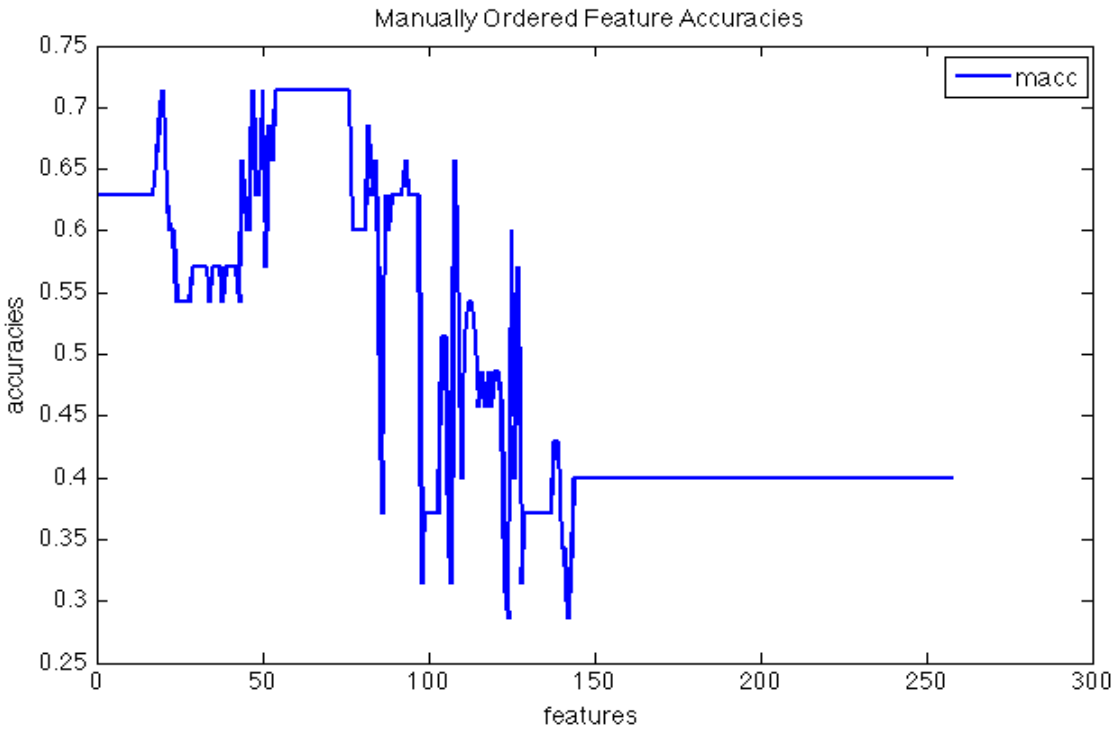

Figure 5.2: **Manually Ordered Test Accuracy Chart**. The features on the x–axis correspond to the Rank column in Table 4.6

5.2.2 Class Accuracies

Unlike the Relief ordered features, the manually ordered and selected features were able to detect and separate out the c2 class from the other classes: c1 and s1. While c1 and c2 both had small positive correlations with the overall accuracy, s1 had a very slight negative correlation. Comparing the classes to each other, c1 vs c2, and c1 vs s1 had strong negative correlations, while c2 vs s1 had a good positive correlation.

Accuracies	Correlations	Accuracies	Correlations
overall vs c1	0.9142	c1 vs c2	–0.1166
overall vs c2	0.2160	c1 vs s1	–0.3416
overall vs s1	0.0418	c2 vs s1	0.5296

Table 5.2: **Manually Ordered – Class Correlations**. The overall accuracy had a strong correlation to c1‘s accuracy, and no correlation with s1. S1 improved its score as c2 improved, and c1 decreased.

In Table 5.2 it is obvious there is a strong correlation between the overall system accuracy and the accuracy for the c1 class. There is a minor correlation for the c2 class to the overall system accuracy. There is no correlation between the overall system accuracy and the s1 class. This does show the effectiveness of the manually selected features in being able to find a good configuration for data separation with less features. The surprising result of Table 5.2 is while the c1 class had minor negative correlations to both the c2 and s1 classes, the c2 and s1 classes had a good positive correlation to each other. Both tables, 5.1 and 5.2, also show the highest effect on the overall system accuracy for both systems was the c1 class.

5.3 Test Configuration Accuracies

The most important features in getting the highest classification accuracies for both the automatic and the manual configurations, and both their overall and class scores were the durational features. While in the overall, c1, and c2 accuracies, all the words' durational features, c2 in particular heavily relied on the common word durations in order to make its decisions. The overall, c1, and c2 also found the Damerau-Levenshtein distances between syllabic transcriptions and the best scored native English speakers important. The syllabic and the common word durations were particularly import in getting high accuracies in the s1 class.

It was surprising to see that in the top 22 features for both configurations, some of the features found important in the manual configuration were missing from the automatic configuration and vice versa. For instance, feature number 41, the ratio of the out of vocabulary words to the in vocabulary words per passage was ranked number 15 in the manual configuration, while it was absent from the top 22 in the automatic configuration. The same for feature number 26, the count of all the common word syllables per passage, was ranked number 16 in the manual configuration, and also absent from the top 22 of the automatic configuration. For the automatic configuration, feature number 45, the article count per each read passage, is ranked 12th, while in the manual configuration is has been deleted from the important feature list.

In both the automatic and the manual configurations, the first feature with the highest accuracy is the Damerau-Levenshtein distances between syllabic transcriptions and the best scored native English speakers. Sentence 4 in the automatic configuration has the highest accuracy, while sentence 6 has the highest accuracy in the manual configuration. For both the automatic and the manual configurations, the statistics for the word durations both had a significant impact on the overall system accuracy. While for the manual configuration, the common word durational statistics had the most effect, the automatic configuration was effected by the summations and

the standard deviations of the all the words‘ durational statistics. The automatic configuration differs from the manual configuration by its use average passage pitch and the Out of Vocabulary (OOV) syllable count in order to find good data separation and high overall system accuracies.

In Table 5.3 the automatic configuration arrived at the highest accuracy when it placed 138 features into the automatic configuration, but the manual configuration started reaching its highest accuracy at 20 features. In Table 5.3 the rank number is the feature‘s placement within the respective automatic or manual configuration, and the feature number is the ranking of the feature in the full featured SVM vector in Appendix D. The dash between the numbers is trying associate the rank of the feature within its respective configuration to the rank in the fully featured SVM vector.

5.3.1 Class Accuracies

In the class accuracies for the testing configuration, the automatic has the automatic configuration in the same top 10 spots. The manual configuration has different features in the top 10 spots depending on the individual class. As in the overall accuracies in the c1 class for the automatic and the manual configurations, the first feature with the highest accuracy is the Damerau-Levenshtein distances between syllabic transcriptions and the best scored native English speakers. Sentence 4 in the automatic configuration, and sentence 2–4 in the manual configuration. For the automatic configuration, the summations and the standard deviations of the word durations had a significant impact on the c1 class. In the manual configuration, the mean syllable duration for sentence 6 and the statistics of the word durations proved important.

The automatic configuration did not detect the c2 class, so all the information in Table 5.5 is the manual configuration. Due to the absence of the automatic configuration results, Table 5.5 has a slightly different format and shows the results for all the class sorted in descending order of the c2 class. The common word median in

10 Highest Accuracy Features				
	Automatic		Manual	
No.	A. Accuracy	Rank – Feature	M. Accuracy	Rank – Feature
1	0.7142	138 – 52	0.7142	20 – 50
2	0.7142	139 – 143	0.7142	53 – 131
3	0.7142	140 – 124	0.7142	54 – 150
4	0.7142	141 – 76	0.7142	55 – 151
5	0.7142	142 – 101	0.7142	56 – 152
6	0.7142	143 – 81	0.7142	57 – 153
7	0.7142	144 – 72	0.7142	58 – 154
8	0.7142	145 – 103	0.7142	59 – 155
9	0.7142	146 – 120	0.7142	60 – 162
10	0.6857	147 – 78	0.7142	61 – 163

Table 5.3: **Testing Accuracies – Overall System – Top 10**. The Rank number refers to that features rank within either the automatic or the manual configuration. The feature numbers can be associated with their feature names in Appendix D. Before reaching the highest accuracy, for the highest first highest ranked feature, there were n-1 features in front of that specific feature in order to reach the highest accuracy numbers.

sentence 2 was the feature giving the best separation for c2. The common words in sentence 2 are “the, said, let, me, have, it”. The common syllable count per passage, the word and syllabic durations, and the Damerau-Levenshtein distances between syllabic transcriptions and the best scored native English speakers also proved important.

All top 10 features which were important for the s1 class were either syllable or common word durations. The same features in the automatic configuration which proved important for the overall and c1 class accuracies also proved important for

C1 – 10 Highest Accuracy Features				
	Automatic		Manual	
No.	A. Accuracy	Rank – Feature	M. Accuracy	Rank – Feature
1	1.0	138 – 52	1.0	19 – 49
2	1.0	139 – 143	1.0	21 – 51
3	1.0	140 – 124	0.9545	107 – 238
4	1.0	141 – 76	0.9090	20 – 50
5	1.0	142 – 101	0.8636	18 – 48
6	1.0	143 – 81	0.8181	53 – 131
7	1.0	144 – 72	0.8181	54 – 150
8	1.0	145 – 103	0.8181	55 – 151
9	1.0	146 – 120	0.8181	56 – 152
10	1.0	147 – 78	0.8181	57 – 153

Table 5.4: **C1 – Testing Accuracies – Top 10**. The Rank number refers to that features rank within either the automatic or the manual configuration. The feature numbers can be associated with their feature names in Appendix D. Before reaching the highest accuracy, for the highest first highest ranked feature, there were n-1 features in front of that specific feature in order to reach the highest accuracy numbers.

the s1 class accuracies. The standard deviation for the syllable durations for sentence 2, and the mean and standard deviation for the syllable durations for sentence 6 were important in differentiating the s1 class. The common word summation for sentence 3, the minimum common word duration for sentence 4, and common word durational statistics for sentence 6 also proved important.

C2 – 10 Highest Accuracy Features Manual Configuration					
No.	Rank	Feature	C1	C2	S1
1	105	105	0.1818	0.8333	0.4285
2	46	46	0.7272	0.6666	0.7142
3	85	197	0.0909	0.6666	1.0
4	122	248	0.1363	0.6666	0.5714
5	49	127	0.7727	0.5	0.7142
6	52	130	0.6818	0.5	0.7142
7	83	195	0.6818	0.5	0.7142
8	44	116	0.5909	0.5	0.7142
9	45	117	0.5909	0.5	0.7142
10	53	131	0.8181	0.3333	0.7142

Table 5.5: **C2 – Testing Accuracies – Top 10**. The automatic configuration did not differentiate the c2 class. Only the manual configuration is included in this table. The feature numbers can be associated with their feature names in Appendix D. Before reaching the highest accuracy, for the highest first highest ranked feature, there were n-1 features in front of that specific feature in order to reach the highest accuracy numbers.

5.3.2 Language Accuracies

In Table 5.7, each individual L1 language correlation is calculated to its matching class, to see how dependent the class accuracy is to that particular language‘s same class. The overall correlation shows how dependent the overall accuracy is to that particular languages class. Finally, the class and overall dependancies are shown at the end of each class to measure how related each class is to the overall accuracy. Not surprisingly, except for hungarian, each individual L1 language in the c1 class had a strong relationship with the overall accuracy in the automatic configuration, and

S1 – 10 Highest Accuracy Features				
	Automatic		Manual	
No.	A. Accuracy	Rank – Feature	M. Accuracy	Rank – Feature
1	0.7142	138 – 52	1.0	85 – 197
2	0.7142	139 – 143	0.85714	108 – 239
3	0.7142	140 – 124	0.85714	109 – 240
4	0.7142	141 – 76	0.7142	46 – 118
5	0.7142	142 – 101	0.7142	49 – 127
6	0.7142	143 – 81	0.7142	53 – 131
7	0.7142	144 – 72	0.7142	54 – 150
8	0.7142	145 – 103	0.7142	55 – 151
9	0.7142	146 – 120	0.7142	56 – 152
10	0.7142	147 – 78	0.7142	57 – 153

Table 5.6: **S1 – Testing Accuracies – Top 10**. The Rank number refers to that features rank within either the automatic or the manual configuration. The feature numbers can be associated with their feature names in Appendix D. Before reaching the highest accuracy, for the highest first highest ranked feature, there were n-1 features in front of that specific feature in order to reach the highest accuracy numbers.

all the languages working together have a very strong relationship with the overall accuracy. This is a good example of how features working together can really provide good data separation and really improve a system‘s working performance. Not all the languages listed in Table 4.7 are listed in Table 5.7. All the omitted languages do not have a speaker in one of the three classes “c1, c2, and s1”, but only a speaker or speakers in the “ot” class.

Language / Class Correlations					
L1 Language	Class	Class Correlation		Overall Correlation	
		Automatic	Manual	Automatic	Manual
arabic	C1	0.7725	0.8634	0.7333	0.9095
chinese	C1	0.7094	0.2283	0.8173	0.1669
french	C1	0.7268	0.7616	0.5656	0.7459
german	C1	0.8252	0.7542	0.8253	0.5379
hungarian	C1	0.3210	0.2090	0.1417	0.0915
korean	C1	0.8626	0.7432	0.7621	0.7324
persian	C1	0.6815	0.7353	0.6818	0.8113
russian	C1	0.8766	0.9095	0.8767	0.9359
C1	C1			0.9178	0.9142
german	C2	0.0	0.8830	0.0	0.2966
hungarian	C2	0.0	0.6052	0.0	–0.0489
C2	C2			0.0	0.2160
german	S1	1.0	1.0	0.5263	0.0418
S1	S1			0.5263	0.0418

Table 5.7: **Language / Class Correlations**. All these numbers were generated from the one testing configuration. Not all the languages listed in Table 4.7 are listed here, due to some of the languages not having a c1, c2 or s1 class.

Chapter 6

CONCLUSION

In this study, the type of features with the highest overall impact on the system for both the automatic and manual configurations are the durational features. For the overall, c1, and c2 accuracies, all the words‘ durational features were important for the classifier to make its decision, while c2 in particular heavily relied on the common word durations. Both the syllabic and the common word durations were particularly important in getting high accuracies for the s1 class. The c2 classified speakers were only detected by the manual configuration, and had zero results in the automatic configuration.

The system was able to classify speakers into the three different classes with good success although there were a relatively small amount of features. In the work of Chen, Evanini and Sun, formant and vowel features boosted the accuracy of their systems (2010 [7]), and I believe the addition of formant and vowel features could possibly boost this overall system accuracy above 80%. I would like to see the addition of phonetic features used in an overall automatic tutoring system or into CALL systems, where the system will be able to detect a speakers English ability, and then automatically tailor its teaching style to that particular speaker. Most L2 learners of English do not have ready access to a social support system of native English speakers who will be able to reinforce, and repeat the words with a correct pronunciation. Most L2 English learners have the most contact with other L2 English learners who are making the same pronunciation mistakes they are. Although each L2 English learners is learning English through the ”affective filter” mentioned in the Introduction, there are similarities can be quantified and used to automatically detect their proficiency in English.

As the system progresses and matures, lexical semantic and syntactic features can be incorporated into the system for a more fine-grained approach. As the English learner progresses in their learning levels, the tutoring or CALL system should also detect this, and automatically tailor itself to the learners level at that particular time. I would like to see the system ultimately develop into an interactive artificial intelligence system, which could possibly independently learn from the English learners, teach the English learners, and adapt to each persons learning style in many different subjects. Eventually, because both children and adults have similar learning issues when they are trying to learn second, third or more languages, the system should also be perfected with data from students under and over the age of 18, incorporating the data from children below the age of 13 as well. Also, the system should be expanded to include both adults and children with or without learning or physical disabilities.

BIBLIOGRAPHY

[1] T. Abeel, Y.V. de Peer, and Y. Saeys. Java-ml: A machine learning library. *Journal of Machine Learning Research*, 10:931–934, 2009.

[2] A. Ben-Hur and J. Watson. A users guide to support vector machines. *Methods in Molecular Biology*, 609:223–239, 2010.

[3] P. Boersma and D. Weenink. Praat: A system for doing phonetics by computer, 2011.

[4] D.K. Burnham, L.J. Earnshaw, and J.E. Clark. Development of categorical identification of native and non-native bilabial stops: infants, children and adults. *Journal of Child Language*, 18:231–260, 1991.

[5] J. Carson-Berndsen, U. Gut, and R. Kelly. *Discovering Regularities in Non-native Speech*. Rudopi, 2006.

[6] Chang, Chih-Chung, and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[7] L. Chen, K. Evanini, and X. Sun. Assessment of non-native speech using vowel space characteristics. *IEEE Workshop on Spoken Language Technology*, December 2010.

[8] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[9] K. Zechner D. Higgins, X. Xi and D. Williamson. A three-stage approach to the automated scoring of spontaneous spoken responses. *Computer Speech and Language*, 25(1):282–306, June 2011.

[10] L. Wong Fillmore. When learning a second language means losing the first. *Early Childhood Research Quarterly*, 6:323–346, 1991.

[11] J.E. Flege, O.S. Bohn, and S. Jang. Effects of experience on non-native speakers production and perception of english vowels. *Journal of Phonetics*, 25:437–470, 1997.

[12] J.B. Gleason. *The Development of Language*. Macmillan Publishing Company, New York, 1993.

[13] U. Gut. The leap corpus, 2004.

[14] I. Guyon and A. Elisseeff. An introduction to feature extraction. In *Feature Extraction, Foundations and Applications*, pages 1–12. Springer, 2006.

[15] M.K. Halliday. *Learning how to mean: Explorations in the development of language*. Edward Arnold, London, 1975.

[16] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, San Diego, 2006.

[17] T.J. Hazen. Direct and latent modeling techniques for computing spoken document similarity. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*, pages 366 –371, dec. 2010.

[18] Chih-Wei Hsu, C C Chang, and Chih-Jen Lin. A practical guide to support vector classification. *Bioinformatics*, 1(1):1–16, 2003.

[19] X. Huang, A. Acero, and H.W. Hon. *Spoken Language Processing*. Prentice Hall PTR, Upper Saddle River, 2001.

[20] Y. Liu J. Kolar and E. Shriberg. Speaker adaptation of language and prosodic models for automatic dialog act segmentation of speech. *Speech Communications*, 52:236–245, 2010.

[21] T. Joachims. Text categorization with support vector machines: learning with many relevant features. *Proc. of the European Conference on Machine Learning*, pages 137–142, 1998.

[22] A. Juneja and C. Espy-Wilson. Speech segmentation using probabilistic phonetic feature hierarchy and support vector machines. *Proceedings of International Joint Conference on Neural Networks*, 2003.

[23] K. Kira and L. Rendell. A practical approach to feature selection. In D. Sleeman and P. Edwards, editors, *International Conference on Machine Learning*, pages 294–256, Aberdeen, July 1992. Morgan Kaufman.

[24] S.D. Krashen. *Principles and practice in second language acquisition*. Prentice-Hall International, Englewood Cliffs, 1987.

[25] K.P. Li and J.E. Porter. Normalizations and selection of speech segments for speaker recognition scoring. *Proceedings of ICASSP 88*, 1:595–598, 1988.

[26] J. Liljencrants and B. Lindblom. Numerical simulation of vowel quality systems: The role of perceptual contrast. *Language*, 48:839–862, 1972.

[27] C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[28] MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

[29] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March 2001.

[30] F. Paluncic, H.C. Ferreira, T.G. Swart, and W.A. Clarke. Modelling distances between genetically related languages using an extended weighted levenshtein distance. *Southern African Linguistics and Applied Language Studies*, 27(4):381–389, 2009.

[31] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. Section 16.5 support vector machines. In *Numerical Recipes: The Art of Scientific Computing*, New York, 2007. Cambridge University Press.

[32] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, February 1988.

[33] S. Schwarm and M. Ostendorf. Reading level assessment using support vector machines and statistical language models. *Proceedings of the 43rd Annual Meeting of the ACL*, pages 532–530, 2005.

[34] J.K. Shah, B.Y. Smolenski, R.E. Yantorno, and A.N. Iyer. Sequential k-nearest neighbor pattern recognition for usable speech classification. *European Signal Processing Coference*, sept 2004.

[35] E. Shriberg, L. Ferrer, S. Kajarekar, N. Scheffer, A. Stolke, and M. Akbacak. Detecting nonnative speech using speaker recognition approaches. *Proc. Odyssey Speaker and Language Recognition Workshop*, Jan 2008.

[36] D.Q. Thang and P.T. Huy. Determining restricted damerau–levenshtein editdistance of two languages by extended automata. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on*, pages 1 –6, nov. 2010.

[37] Esko Ukkonen. On approximate string matching. In Marek Karpinski, editor, *Foundations of Computation Theory*, volume 158 of *Lecture Notes in Computer Science*, pages 487–495. Springer Berlin / Heidelberg, 1983.

[38] L. Wang, X. Feng, and H. Meng. Mispronunciation detection based on cross-language phonological comparisons. *Proc. IEEE IET Int. Conf. Audio Lang. Image Process*, pages 301–311, 2008.

[39] J.F. Werker and R.C. Tees. Cross-language speech perception: Evidence for perceptual reorganization during the first year of life. *Infant Behavior and Development*, 7:49–64, 1984.

[40] R. Wright. Factors of lexical competition in vowel articulation. *Phonetic Interpretation*, 2003.

Appendix A

COMMON WORDS USED

a	able	about	across	after	all	almost
also	am	among	an	and	any	are
as	at	be	because	been	but	by
can	cannot	could	dear	did	do	does
either	else	ever	every	for	from	get
got	had	has	have	he	her	hers
him	his	how	however	i	if	in
into	is	it	its	just	least	let
like	likely	may	me	might	most	must
my	neither	no	nor	not	of	off
often	on	only	or	other	our	own
rather	said	say	says	she	should	since
so	some	than	that	the	their	them
then	there	these	they	this	tis	to
too	twas	us	wants	was	we	were
what	when	where	which	while	who	whom
why	will	with	would	yet	you	your

Table A.1: Common Word List

Appendix B

THE TIGER AND THE MOUSE

The story below was recited by the LeaP study participants during the LeaP Corpus recordings:

"A tiger and a mouse were walking in a field when they saw a big lump of cheese lying on the ground. The mouse said: "Please, tiger, let me have it. You don't even like cheese. Be kind and find something else to eat." But the tiger put his paw on the cheese and said: "It's mine! And if you don't go I'll eat you too." The mouse was very sad and went away.

The tiger tried to swallow all of the cheese at once but it got stuck in his throat and whatever he tried to do he could not move it. After a while, a dog came along and the tiger asked it for help. "There is nothing I can do." said the dog and continued on his way. Then, a frog hopped along and the tiger asked it for help. "There is nothing I can do." said the frog and hopped away.

Finally, the tiger went to where the mouse lived. She lay in her bed in a hole which she had dug in the ground. "Please help me," said the tiger. "The cheese is stuck in my throat and I cannot remove it." "You are a very bad animal," said the mouse. "You wouldn't let me have the cheese, but I'll help you nonetheless. Open your mouth and let me jump in. I'll nibble at the cheese until it is small enough to fall down your throat." The tiger opened his mouth, the mouse jumped in and began nibbling at the cheese. The tiger thought: "I really am very hungry.." "

(Gut 2004 [13])

Appendix C

EXPERIMENTAL GROUPINGS

Group 1	Test	ger bc c2, ger be c2, ger by s1, ger bz s1, ger ev ot, hun ay c1, hun ay c2, kor bg c1, spa ch ot
	Train	ara ax c1, chi bb c1, eng ai na, eng ca na, eng cb na, fre bj c1, ger aw c1, ger bc c1, ger be c1 ger be c3, ger bl c1, ger bl c2, ger bm e1, ger bn e1, ger bo e1, ger bp e1, ger bq e1, ger br e1, ger bs e1, ger bt s1, ger bu s1, ger bv s1, ger bw s1, ger bx s1, hun az c1, hun az c2, kor bi c1 per bk c1, rus ba c1, rus bd c1, rus bh c1, rus dv c1
Group 2	Test	ger be c3, ger bl c1, ger bt s1, ger bv s1, ger bw s1, hun az c2, ibb cm ot, ibo aj ot, rus bd c1
	Train	ara ax c1, chi bb c1, eng ai na, eng ca na, eng cb na, fre bj c1, ger aw c1, ger bc c1, ger bc c2 ger be c1, ger be c2, ger bl c2, ger bm e1, ger bn e1, ger bo e1, ger bp e1, ger bq e1, ger br e1, ger bs e1, ger bu s1, ger bx s1, ger by s1, ger bz s1, hun ay c1, hun ay c2, hun az c1, kor bg c1, kor bi c1, per bk c1, rus ba c1, rus bh c1, rus dv c1
Group 3	Test	fre bj c1, ger aw c1, ger bc c1, ger bm e1, ger bp e1, ger bx s1, kor ci ot, rus ba c1, spa cg ot
	Train	ara ax c1, chi bb c1, eng ai na, eng ca na, eng cb na, ger bc c2, ger be c1, ger be c2, ger be c3 ger bl c1, ger bl c2, ger bn e1, ger bo e1, ger bq e1, ger br e1, ger bs e1, ger bt s1, ger bu s1 ger bv s1, ger bw s1, ger by s1, ger bz s1, hun ay c1, hun ay c2, hun az c1, hun az c2, kor bg c1, kor bi c1, per bk c1, rus bd c1, rus bh c1, rus dv c1

Group 4	Test	ara ax c1, ger bl c2, ger bo e1, ger bq e1, ger br e1, kor bi c1, rus bh c1, rus dv c1
	Train	chi bb c1, eng ai na, eng ca na, eng cb na, fre bj c1, ger aw c1, ger bc c1, ger bc c2, ger be c1 ger be c2, ger be c3, ger bl c1, ger bm e1, ger bn e1, ger bp e1, ger bs e1, ger bt s1, ger bu s1 ger bv s1, ger bw s1, ger bx s1, ger by s1, ger bz s1, hun ay c1, hun ay c2, hun az c1, hun az c2, kor bg c1, per bk c1, rus ba c1, rus bd c1
Group 5	Test	chi bb c1, ger be c1, ger bn e1, ger bs e1, ger bu s1, hun az c1, per bk c1, yor cq ot
	Train	ara ax c1, eng ai na, eng ca na, eng cb na, fre bj c1, ger aw c1, ger bc c1, ger bc c2, ger be c2 ger be c3, ger bl c1, ger bl c2, ger bm e1, ger bo e1, ger bp e1, ger bq e1, ger br e1, ger bt s1 ger bv s1, ger bw s1, ger bx s1, ger by s1, ger bz s1, hun ay c1, hun ay c2, hun az c2, kor bg c1, kor bi c1, rus ba c1, rus bd c1, rus bh c1, rus dv c1

Table C.1: **Testing Configuration**. The full testing and training configuration for the experimental dataset.

Appendix D

SVM AUTOMATICALLY SELECTED FEATURES

Full SVM Feature Vector	
Sentence Features	
1–7	Sentence: Damerau–Levenshtein Distances Sentences 1-7 from the original text
8–14	Sentence: Cosine Similarity Distances Sentences 1-7 from the original text
15–20	Sentence: Damerau–Levenshtein Distance Statistics (in order): Summation, Minimum, Maximum, Median, Mean, Standard Deviation
21–26	Sentence: CosineSimilarity Distance Statistics (in order): Summation, Minimum, Maximum, Median, Mean, Standard Deviation
27–33	Sentence: Damerau–Levenshtein Distance Non–Common Word Sentences 1-7 from the original text
34–40	Sentence: Cosine Similarity Distance Non–Common Word Sentences 1-7 from the original text
41	Passage: Ratio of the words said Out of Vocabulary to In Vocabulary (only the words from the story text are considered In Vocabulary)
42	Passage: Ratio of the counted words In Vocabulary to the total amount of the characters for the words
43–45	Passage: Count of the Out of Vocabulary words, the count of the In Vocabulary words, the count of the articles per each sentence

Full SVM Feature Vector	
Syllable Features	
46	Passage: Common syllable count
47–53	Sentence: Damerau–Levenshtein Distances syllalble transcribed sentences 1-7 from the original text
54–60	Sentence: Cosine Similarity Distances syllalble transcribed sentences 1-7 from the original text
61–67	Sentence: Damerau–Levenshtein Distance syllable transcribed statistics (in order): Summation, Minimum, Maximum, Median, Mean, Standard Deviation
68–74	Sentence: CosineSimilarity Distance syllable transcribed tatistics (in order): Summation, Minimum, Maximum, Median, Mean, Standard Deviation
75	Passage: Count of all the syllables per sentence which were In Vocabulary
76	Passage: Count of all the syllables per sentence which were Out of Vocabulary
Pitch Features	
77–82	Passage: pitch statistics (in order): Summation, Minimum, Maximum, Median, Mean, Standard Deviation
83	Passage: the total length of the pitch array
Duration Features	
84–89	Sentence 1: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
90–95	Sentence 1: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
96–101	Sentence 2: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation

Full SVM Feature Vector	
102–107	Sentence 2: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
108–113	Sentence 3: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
114–119	Sentence 3: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
120–125	Sentence 4: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
126–131	Sentence 4: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
132–137	Sentence 5: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
138–143	Sentence 5: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
144–149	Sentence 6: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
150–155	Sentence 6: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
156–161	Sentence 7: Word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
162–167	Sentence 7: Common word durations per sentence: Maximum, Minimum, Mean, Median, Sum, Standard Deviation
168–174	Sentences 1–7: Average common word count per sentence
175	Passage: Average common word duration for passage
176–180	Sentence 1: Syllable durations: last five per sentence
181–186	Sentence 1: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
187–191	Sentence 2: Syllable durations: last five per sentence

Full SVM Feature Vector	
192–197	Sentence 2: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
198–202	Sentence 3: Syllable durations: last five per sentence
203–208	Sentence 3: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
209–213	Sentence 4: Syllable durations: last five per sentence
213–218	Sentence 4: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
219–223	Sentence 5: Syllable durations: last five per sentence
224–229	Sentence 5: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
230–234	Sentence 6: Syllable durations: last five per sentence
235–240	Sentence 6: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
241–245	Sentence 7: Syllable durations: last five per sentence
246–251	Sentence 7: Syllable duration statistics per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
252–257	Passage: Pause durations, Maximum, Minimum, Mean, Median, Sum, Standard Deviation
258	Passage: Common Word Count

Table D.1: **Full SVM Feature Vector**. Row by row breakdown of each vector feature for the SVM.

Relief Automatically Ordered Feature Vector	
Ranks	Feature Numbers
1 – 20	3, 4, 156, 133, 144, 123, 11, 85, 86, 47, 8, 45, 126, 9, 142, 94, 10, 53, 82, 102
21 – 40	49, 121, 43, 44, 125, 59, 109, 106, 50, 26, 130, 118, 18, 97, 89, 57, 15, 77, 30, 155
41 – 60	116, 122, 21, 24, 105, 140, 40, 69, 110, 80, 127, 148, 38, 128, 149, 68, 58, 139, 66, 65
61 – 80	51, 88, 63, 60, 56, 36, 79, 99, 152, 91, 131, 135, 145, 33, 150, 20, 104, 1, 23, 2
81 – 100	35, 96, 70, 146, 75, 29, 17, 74, 64, 147, 22, 54, 153, 100, 132, 32, 6, 83, 117, 136
101 – 120	87, 92, 95, 31, 19, 41, 61, 34, 5, 67, 98, 14, 108, 93, 107, 13, 46, 90, 151, 37
121 – 140	28, 39, 12, 42, 134, 129, 48, 27, 154, 113, 7, 55, 73, 112, 84, 115, 16, 52, 143, 124
141 – 160	76, 101, 81, 72, 103, 120, 78, 137, 25, 141, 138, 71, 114, 111, 62, 119, 157, 158, 160, 164
161 – 179	43, 161, 162, 163, 165, 159, 166, 167, 168, 170, 171, 172, 173, 174, 175, 176, 177, 178, 169

Table D.2: **Relief Ordered Feature Ranking**. All the missing numbers 179–258 from the full feature vector are placed at the end of the feature vector and had very little impact on the automatic vector accuracies.

Manually Ordered Feature Vector		
Rank	Feature No.	Feature
1–7	1–7	Damerau–Levenshtein distances between sentences 1–7 without the common words
8–14	8–14	Cosine Similarity distance in sentence 1–7 without the common words
15	41	Ratio of Out of Vocabulary word to In Vocabulary words per passage
16	46	A count of all the common word syllables per passage
17-23	47 – 53	Damerau–Levenshtein distances between syllabic transcriptions 1–7 and the best scored native English speaker
24 – 29	77 – 82	Pitch statistics for the words per passage: Summation, Minimum, Maximum, Median, Mean, Standard Deviation
30	83	Numbers of pitches per passage
30–35 36–41 42–47 48–53 54–59 60–65	90–95 102–107 114–119 126–131 150–155 162–167	Word Durations per sentence: maximum, minimum, mean, and median Common Word durations per sentence: summation and standard deviation
66–72	168–174	Common word count per sentence
73	175	Average common word duration per passage
74–79 80–85 86–91 92–97 98–103 104–109	181–186 192–197 203–208 213–218 224–229 235–240	Duration statistics for the syllables per sentence: Summation, Minimum, Maximum, Median, Mean, Standard Deviation

110–115	246–251	
116	252	Maximum pause duration per passage
117	252	Minimum pause duration per passage
118	256	Summation of all the pauses per passage
119	257	Standard deviation of all the pauses per passage
120	258	Common word count for the passage

Table D.3: **Full Manually Selected Feature Set**. The missing features from the full feature vector are placed at the end of the vector and had very little impact on the manual vector accuracies.

Appendix E

PRAAT SCRIPT

```
# This script goes through sound and TextGrid files in a directory,
# opens each pair of Sound and TextGrid, calculates the duration
# of each labeled interval in the phone tier,
# the pitch maximum at the center of the phone, and
# the duration of the corresponding interval in the syllable tier,
# and then saves the results to a text file.
#
# To make some other or additional analyses, you can modify the script
# yourself... it should be reasonably well commented! ;)
#
# This script is distributed under the GNU General Public License.
# Copyright 25.11.2004 Mietta Lennes
# Edited Stella Podgornik January 18, 2011
#
# To run this script:  .../Praat <scriptname>

# ../../../../../../../../Applications/Praat.app/Contents/MacOS/Praat get-syllables.p
wave/ara_ax/ wave/ara_ax/ ara_ax

form Analyze durations of cvs and the corresponding syllables
comment Directory of sound files
text resultsdir ""
text sound_directory ""
text textgrid_directory ""
text lang ""
endform
```

```
# All the settings that don't get picked up in form if run on command line
#log files and directory settings
        logfile$ = "'resultsdir$'log/'lang$'_log.txt"
        sound_file_extension$ = ".wav"
        textgrid_file_extension$ = ".TextGrid"
        resultfile$ =  "'resultsdir$'data/'lang$'_pitchformants.txt"
#comment Which tier contains the speech sound segments?
        cv_tier$ = "vowels"
        cv_tier_int = 0
        tone_tier$ = "tones"
        words_tier$ = "words"
#comment Which tier contains the syllable segments?
        syllable_tier$ = "syll"
#comment Pitch analysis parameters
        time_step = 0.01
        minimum_pitch = 70
        maximum_pitch = 700
#comment Formant analysis parameters, adult female
maximum_number_of_formants = 5
maximum_formant = 5500
window_length = 0.025
preemphasis_from_hz = 50
formant_step = 0.0005

printline 'resultsdir$' 'sound_directory$' 'textgrid_directory$'
printline 'resultfile$' 'logfile$'

#Get a listing of all the sound files in the sound_directory
#Sound files
Create Strings as file list...
list 'sound_directory$'*'sound_file_extension$'
numberOfSoundFiles = Get number of strings
for sfile to numberOfSoundFiles
```

```
filesname$ = Get string... sfile
fileappend 'logfile$' 'filesname$'
fileappend 'logfile$' 'newline$'
endfor

fileappend 'logfile$' 'newline$'

#files with a TextGrid annotation file.  not all the wav files are annotated
#good to use sound files with annotation
Create Strings as file list... list 'textgrid_directory$'*
'textgrid_file_extension$'
numberOfFiles = Get number of strings
for tfile to numberOfFiles
filetname$ = Get string... tfile
fileappend 'logfile$' 'filetname$'
fileappend 'logfile$' 'newline$'
endfor
fileappend 'logfile$' 'newline$'

# Check if the result file exists, comment out if you want to keep the
# old com.stellamarie.thesis.many:
if fileReadable (resultfile$)
filedelete "'resultfile$'"
endif

# Check and see if log file exists, comment out if you want to keep the
# old com.stellamarie.thesis.many:
if fileReadable (logfile$)
        filedelete "'logfile$'"
endif


# Write a row with column titles to the result file:
```

```
# (remember to edit this if you add or change the analyses!)

titleline$ = "Filename word_label$ syllable_label$ syl_start syl_end tones
pitchmin pitchmed pitchmax onset opitch of1 of2 of3 midpoint mpitch
mf1 mf2 mf3 glide gpitch gf1 gf2 gf3"
titleline$ = titleline$ + "'newline$'"
fileappend 'resultfile$' 'titleline$'

# Go through all the sound files, com.stellamarie.thesis.many
# by com.stellamarie.thesis.many:

for ifile to numberOfFiles
    cv_tier_int = 0
    tone_tier_no = 0
    select Strings list
filename$ = Get string... ifile
    name$ = filename$ - "'textgrid_file_extension$'"
# Starting from here, you can add everything that should be
# repeated for every TextGrid file that was opened:
fileappend 'logfile$' 'name$'
    fileappend 'logfile$' 'newline$'
    #Open the sound file:
    sname$ = "'textgrid_directory$''name$''sound_file_extension$'"
    fileappend 'logfile$' 'sname$'
    fileappend 'logfile$' 'newline$'
    Read from file... 'sname$'
    soundname$ = selected$ ("Sound", 1)
    To Formant (burg)... time_step maximum_number_of_formants
     maximum_formant window_length preemphasis_from_hz
     # Open a TextGrid by the same name:
    gridfile$ = "'textgrid_directory$''name$''textgrid_file_extension$'"
    fileappend 'logfile$' 'gridfile$'
    fileappend 'logfile$' 'newline$'
```

```
if fileReadable (gridfile$)
Read from file... 'gridfile$'
# Find the tier number that has the label given in the form:
                select TextGrid 'soundname$'
                nTiers = Get number of tiers
                cv_tier = 0
                syllable_tier = 0
                word_tier = 0
            for i from 1 to 'nTiers'
    tname$ = Get tier name... 'i'
    if tname$ == "'cv_tier$'"
                call GetTier 'cv_tier$' cv_tier
    endif
    if tname$ == "'syllable_tier$'"
                 call GetTier 'syllable_tier$' syllable_tier
    endif
    if tname$ == "'words_tier$'"
                 call GetTier 'words_tier$' word_tier
    endif
                if tname$ <> "tones"
                    cv_tier_int = 0 + cv_tier_int
                else
    cv_tier_int = 1 + cv_tier_int
                endif
                start = Get starting time
                end = Get finishing time
                fileappend 'logfile$' 'tname$' 'cv_tier_int' 'tone_tier_no'
                  'newline$'
            endfor

            if cv_tier_int >= 1
                call GetTier 'tone_tier$' tone_tier_no
            elsif cv_tier_int < 1
```

```
      tone_tier_no = 0
            endif

            fileappend 'logfile$' 'tone_tier$' 'cv_tier_int' 'tone_tier_no'
            fileappend 'logfile$' 'newline$'

   if syllable_tier > 0
numberOfSylIntervals = Get number of intervals... syllable_tier
preceding_label$ = ""
#based on the Hirst procedure below,
# attempts to get realistic speaker range
            call calculate_min_max_f0
            select Sound 'soundname$'
            To Pitch... time_step min_f0 max_f0
            select Sound 'soundname$'
            To Intensity... 50 0 yes
# Pass through all intervals in the selected syllable tier:
for interval to numberOfSylIntervals
    word_label$ = ""
    vowels$ = ""
    pitchmax = 0
    pitchmin = 0
    pitchmed = 0
    select TextGrid 'soundname$'
syllable_label$ = Get label of interval... syllable_tier interval

# if the interval has an unempty label, get its start and end:
syl_start = Get starting point... syllable_tier interval
syl_end = Get end point... syllable_tier interval
iCV_start = Get interval at time... cv_tier syl_start
                iCV_end = Get interval at time... cv_tier syl_end

# get the duration of the syllable segment
```

```
sylldur = syl_end - syl_start

# get the time at the middle of the syl:
syllcenter = (syl_start + syl_end) / 2
#printline 'syllcenter'

#get the time at 75 of the syl
                syllend = sylldur/4 + syllcenter

# get number of points in the tone tier
                if tone_tier_no > 0
                    select TextGrid 'soundname$'
                    numberPoints = Get number of points... tone_tier_no
else
numberPoints = 0
endif

                tones$ = "tobi"
                select TextGrid 'soundname$'
if cv_tier_int >= 1 and numberPoints > 0
                    startTonePoint = Get nearest index from time...
                     tone_tier_no syllcenter-sylldur/4
                    endTonePoint = Get nearest index from time...
                     tone_tier_no syllend
           startToneTime = Get time of point...
                 tone_tier_no startTonePoint
          endToneTime = Get time of point...
  tone_tier_no endTonePoint
                    firstPoint = Get low index from time...
                     tone_tier_no syl_start
                    lastPoint = Get high index from time...
                     tone_tier_no syl_end
                    pointLabel_one$ = Get label of point...
```

```
                    tone_tier_no startTonePoint
                   pointLabel_two$ = Get label of point...
                    tone_tier_no endTonePoint
                   tones$ = pointLabel_one$
endif

if tones$ == ""
    tones$ = "tobi"
endif

select TextGrid 'soundname$'
                #printline 'soundname$'

                # get the syllable interval number at the cv center:
                if syllable_label$ = ""
    syllable = Get interval at time... cv_tier syllcenter
    syllable_label$ = Get label of interval... cv_tier syllable
    #if syllable_label$ = "PAUSE"
    #    syllable_label$ = "pause"
    #endif
    syllable_label$ = "pause"
    word_label$ = syllable_label$
else
    syllable = Get interval at time... syllable_tier syllcenter
    syllable_label$ = Get label of interval... syllable_tier syllable
    word = Get interval at time... word_tier syllcenter
    if word > 0
        word_label$ = Get label of interval...  word_tier word
        if word_label$ = ""
            last_word = word
            word =  Get interval at time... word_tier syl_end-formant_step
            word_end =  Get end point... word_tier word
            word_label_end$ = Get label of interval...  word_tier word
```

```
            if last_word > word and word_end <= syl_end
                word_label$ = word_label_end$
            endif
        endif
    else
        last = syl_start - 0.05
        word = Get interval at time... word_tier last
        word_label$ = Get label of interval...  word_tier word
                    endif
    printline "'word' 'word_label$'"
                    printline "'syllable' 'syllable_label$'
endif
                vowel$ = ""
                intense$ = ""
    if syllable_label$ <> "pause"
    for iCV from iCV_start to iCV_end
    select TextGrid 'soundname$'
                        iCV_start_time = Get starting point... cv_tier iCV
    iCV_end_time = Get end point... cv_tier iCV
        vowel$ = Get label of interval... cv_tier iCV
        if iCV_end_time > syl_end
            iCV_end_time = syl_end
        endif
        if iCV_start_time < syl_start
                            iCV_start_time = syl_start
        endif
     if vowel$ <> "PAUSE" and vowel$ <> "pause"
                           vowels$ = vowels$ + "'vowel$' 'iCV_start_time'
                                'iCV_end_time' "
                        endif
                    if vowel$ = "V"
                            #get gross measures
                            select Pitch 'soundname$'
```

```
                                pitchmax = Get maximum... iCV_start_time
                                 iCV_end_time Hertz Parabolic
                                pitchmin = Get minimum... iCV_start_time
                                 iCV_end_time Hertz Parabolic
                                pitchmed = Get mean... iCV_start_time
                                 iCV_end_time Hertz
                                pitches$ = "'pitchmin' 'pitchmed' 'pitchmax' 'newline$'"
                                printline 'pitches$'
                                call get_pitches
                                vowels$ = vowels$ + "'onset' 'opitchmin' 'opitchmed'
                                 opitchmax' 'of1' 'of2' 'of3' 'midpoint' 'mpitchmin'
'mpitchmed' 'mpitchmax' 'mf1' 'mf2' 'mf3' 'glide'
'gpitchmin' 'gpitchmed' 'gpitchmax' 'gf1' 'gf2' 'gf3' "
                                call get_intensity
                            endif
        endfor
        #vowels$ = vowels$ + "'newline$'"
                    #printline 'vowels$'
                endif
                resultline$ = "'soundname$' 'word_label$' 'syllable_label$'
                 'syl_start' 'syl_end' 'tones$' 'pitchmin' 'pitchmed'
'pitchmax' "
                resultline$ = resultline$ + intense$
                resultline$ = resultline$ + vowels$
                # Save result to text file:
                resultline$ = resultline$ + "'newline$'"
                #printline 'resultline$'
                fileappend "'resultfile$'" 'resultline$'
            endfor
# Remove the Pitch object
select Pitch 'soundname$'
Remove
#Remove the Formant object
```

```
select Formant 'soundname$'
Remove
# Remove the Intensity object
select Intensity 'soundname$'
Remove
endif
# Remove the TextGrid object from the object list
select TextGrid 'soundname$'
Remove
endif
# Remove the sound object from the object list
select Sound 'soundname$'
Remove
select Strings list
# and go on with the next sound file!
fileappend 'logfile$' 'newline$'
endfor

Remove


#-------------
# This procedure finds the number of a tier that has a given label.

procedure GetTier name$ variable$
        numberOfTiers = Get number of tiers
        itier = 1
        repeat
                tier$ = Get tier name... itier
                itier = itier + 1
        until tier$ = name$ or itier > numberOfTiers
        if tier$ <> name$
                'variable$' = 0
```

```
        else
                'variable$' = itier - 1
        endif

if 'variable$' = 0
printline The tier called 'name$' is missing from the file
'soundname$'!
endif

endproc

#This procedure was developed by Daniel Hirst for the MoMel project.
#version: 2008:07:06
#author  Daniel Hirst
#email: daniel.hirst@lpl-aix.fr
#  estimate of newMaxF0 as 1.5 * quantile 75
#  and newMinF0 as 0.5 * quantile 25
#  rounded to higher (resp. lower) 10

procedure calculate_min_max_f0
        select Sound 'soundname$'
To Pitch... time_step minimum_pitch maximum_pitch
q75 = Get quantile... 0.0 0.0 0.75 Hertz
q25 = Get quantile... 0.0 0.0 0.25 Hertz
max_f0 = 10*ceiling((1.5*q75)/10)
min_f0 = 10*floor((0.75*q25)/10)
Remove
endproc

#! Praat script http://www.fon.hum.uva.nl/paul/p2/speakers/measureVowels.praat
# Paul Boersma, April 24, 2001

#-------------------------
```

```
# get formant values

procedure get_pitches
    # get number points between start and end
    start = iCV_start_time
    end = iCV_end_time
    phonedur = end - start
    # 20
    onset = start + (phonedur/5)
    # 50
    midpoint = start + (end-start)/2
    # 80
    glide = end - (phonedur/5)

    #formants - 20
    select Pitch 'soundname$'
    oc = 0
    opitchmax = Get maximum... start onset Hertz Parabolic
    opitchmin = Get minimum... start onset Hertz Parabolic
    opitchmed = Get mean... start onset Hertz
    if opitchmax = undefined
        while opitchmax = undefined and oc < 5 and onset < midpoint
            if midpoint > onset
             onset = onset + 'formant_step'
         start = start + 'formant_step'
         select Pitch 'soundname$'
                opitchmax = Get maximum... start onset Hertz Parabolic
                opitchmin = Get minimum... start onset Hertz Parabolic
                opitchmed = Get mean... start onset Hertz
            endif
            oc = oc + 1
     endwhile
    endif
```

```
    select Formant 'soundname$'
    of1 = Get quantile... 1 start onset Hertz 0.5
    of2 = Get quantile... 2 start onset Hertz 0.5
    of3 = Get quantile... 3 start onset Hertz 0.5
    fileappend 'logfile$' 'word_label$' 'syllable_label$' 'onset' 'opitchmin'
     'opitchmed' 'opitchmax' "printed formant 20"
    fileappend 'logfile$' 'newline$'

    #formants - 80
    select Pitch 'soundname$'
    gc = 0
    gpitchmax = Get maximum... glide end Hertz Parabolic
    gpitchmin = Get minimum... glide end Hertz Parabolic
    gpitchmed = Get mean... glide end Hertz
    if gpitchmax = undefined
        while gpitchmax = undefined and gc < 5 and glide > midpoint
     if end > midpoint
                    glide = glide - 'formant_step'
                    end = end - 'formant_step'
              select Pitch 'soundname$'
              gpitchmax = Get maximum... glide end Hertz Parabolic
                    gpitchmin = Get minimum... glide end Hertz Parabolic
                    gpitchmed = Get mean... glide end Hertz
          endif
     gc = gc + 1
        endwhile
    endif
    select Formant 'soundname$'
#this is to prevent undefineds
    gf1 = Get quantile... 1 midpoint end Hertz 0.5
    gf2 = Get quantile... 2 midpoint end Hertz 0.5
    gf3 = Get quantile... 3 midpoint end Hertz 0.5
    fileappend 'logfile$' 'word_label$' 'syllable_label$' 'glide' 'gpitchmin'
```

```
     'gpitchmed' 'gpitchmax' "printed formant 80"
    fileappend 'logfile$' 'newline$'

    #formants - 50
    select Pitch 'soundname$'
    mpitchmax = Get maximum... onset glide Hertz Parabolic
    mpitchmin = Get minimum... onset glide Hertz Parabolic
    mpitchmed = Get mean... onset glide Hertz
    select Formant 'soundname$'
    mf1 = Get quantile... 1 onset glide Hertz 0.5
    mf2 = Get quantile... 2 onset glide Hertz 0.5
    mf3 = Get quantile... 3 onset glide Hertz 0.5
    fileappend 'logfile$' 'word_label$' 'syllable_label$' 'midpoint' 'mpitchmin'
     'mpitchmed' 'mpitchmax' "printed formant 50"
    fileappend 'logfile$' 'newline$'
    fileappend 'logfile$' 'newline$'
endproc

#--------------------------
# get intensity values

procedure get_intensity
    select Intensity 'soundname$'
    # get number points between start and end
    start = iCV_start_time
    end = iCV_end_time
    phonedur = end - start

    # estimate noise floor
    minint = Get minimum... start end Parabolic

   # estimate noise max
   maxint = Get maximum... start end Parabolic
```

```
   #get median of Intensity: limits influence of high peaks
   medint = Get quantile... start end 0.5

   intense$ = "'minint' 'medint' 'maxint' "
   printline 'intense$'

endproc
```

Appendix F

PERSONAL

- My page.

  `http://www.stellamarie.com`

VITA

Stella Podgornik is a graduate student at the University of Washington. She is a part time baker with a passion for sourdough breads. Other hobbies include concert production, reading books, and playing guitar.

.