Shibboleth:

An Automated Foreign Accent Identification Program

by

Wende Frost

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved November 2013 by the
Graduate Supervisory Committee:

Elly van Gelderen, Chair
Dennis Perzanowski
Elisabeth Gee

ARIZONA STATE UNIVERSITY

December 2013

ABSTRACT

The speech of non-native (L2) speakers of a language contains phonological rules that differentiate them from native speakers. These phonological rules characterize or distinguish accents in an L2. The Shibboleth program creates combinatorial rule-sets to describe the phonological pattern of these accents and classifies L2 speakers into their native language. The training and classification is done in Shibboleth by support vector machines using a Gaussian radial basis kernel. In one experiment run using Shibboleth, the program correctly identified the native language (L1) of a speaker of unknown origin 42% of the time when there were six possible L1s in which to classify the speaker. This rate is significantly better than the 17% chance classification rate. Chi-squared test (1, N=24) =10.800, p=.0010 In a second experiment, Shibboleth was not able to determine the native language family of a speaker of unknown origin at a rate better than chance (33-44%) when the L1 was not in the transcripts used for training the language family rule-set. Chi-squared test (1, N=18) =1.000, p=.3173 The 318 participants for both experiments were from the Speech Accent Archive (Weinberger, 2013), and ranged in age from 17 to 80 years old. Forty percent of the speakers were female and 60% were male. The factor that most influenced correct classification was higher age of onset for the L2. A higher number of years spent living in an English-speaking country did not have the expected positive effect on classification.

ACKNOWLEDGMENTS

I would like to thank my committee chair, Dr. Elly van Gelderen, for her patience as an advisor and continued encouragement throughout the Ph.D. process. It takes a special kind of chair to hear one of my schemes and trust that the finished product will be anything but a resounding failure, and now she's done it twice. This project would also not have been possible without the assistance of Dr. Dennis Perzanowski, my mentor and boss at the Naval Research Laboratory. He was available as a sounding board throughout the building of Shibboleth, and spent significant time editing and revising with me. Any leftover quirks are entirely my own fault. Dr. Betty Gee stepped in at the last minute to complete my committee, and for that I am very grateful. Dr. Roy Major provided me with much needed aid in background research and his questioning led to some interesting research directions.

Shibboleth was made possible by funding under the Pathways program at the Naval Research Laboratory in Washington, D. C., Work Unit #55-4290. I am greatly appreciative of my collaborators at the lab, Dr. Sam Blisard and Mr. Rob Page. Without their experience, advice, and programming skill, this project would have taken me years longer, if it ever got completed at all. Additionally, I would like to express my gratitude to Juno Schaser for her help in creating all of the graphics in a fraction of the time it would have taken me.

I would also like to thank Greg and Joe Freye, Keith Meinerts, Stacy Kuczyk, Patrick Francis, and Mercedes Murrieta for always making sure I had a place to stay, food to eat, beer to drink, and a ride to the airport, usually during rush hour. Whoever said musicians were unreliable never met you guys.

Finally, thanks to my mom, dad, and Trin for the lunches, dog-sitting, house-cleaning, support, and occasional distraction. I love you all.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1

INTRODUCTION

This thesis describes the design of a computer program, Shibboleth, which automatically recognizes the native language (L1) or dialect of a non-native speaker (NNS) from written phonetic transcriptions of articulatory features in the non-native language (L2). The thesis begins with a brief history of the problem, followed by a chapter describing the design of the Shibboleth modules. After this description, the results of two experiments testing the accuracy of the program are reported. The thesis finishes with a discussion of the results of the experiments, issues found while conducting the experiments, and future work on Shibboleth.

While human listeners can usually tell if someone is a NNS of their own language, they can usually only place the original language of the speaker if the speaker displays a high degree of accentedness and if the accent belongs to an L1 with which the listener is familiar. This program aims to classify NNSs of a language by their L1 for any language with sufficient speech samples in the Speech Accent Archive (Weinberger, 2013). Not only will accurate classification be a useful outcome in itself, it is likely that more pronunciation information and developmental processes from a variety of interlanguages will be discovered.

The program described in this thesis and all related experiments were conducted at the Naval Research Laboratory in Washington, D.C., under the Pathways program. The work was funded under Work Unit #55-4290. I was responsible for designing the Shibboleth program and conducting the experiments, while the underlying Java software was coded by two colleagues at the laboratory.

Since Shibboleth arises from a multi-disciplinary background in linguistics and machine learning, some key terms used in this thesis may be unfamiliar. The definitions used for key terms in this thesis are located in Appendix A.

CHAPTER 2

BACKGROUND

2.1 History

Foreign accent, according to Major (2013), is “a pronunciation deviating from what a [native speaker] expects another [native speaker] to sound like.” Almost all NNSs who have acquired a language past very early childhood have some trace of foreign accent in their speech, even though a younger age of acquisition seems to be highly correlated with a lower perceived amount of foreign accent to native speakers (Flege, Munro, & MacKay, 1995; Flege, Frieda, & Nozawa, 1997; Piske, MacKay, & Flege, 2001; Flege, Yeni-Komshian, & Liu 1999; Long, 1990). Even children whose age of onset is as young as three years old still are not always wholly native in their pronunciation (Flege et al., 1999). In addition, speakers of different dialects within the same L1, for example, Northern and Southern American English, will be perceived as having accented speech when speaking with a member of the other dialect.

There are two avenues through which the accent of a NNS may differ from that of a native speaker: transfer from the NNS’s L1 and developmental processes based on language universals (Major, 2008).

Positive transfer from the L1 of the NNS that results in the NNS achieving a native-like pronunciation of a segment will not be found by Shibboleth, as the segment should be identical to the representative sample. Negative transfer at the level of phonic interference will be one of the main areas of interest. There are four basic types of phonic interference from the L1 that affect the secondary sound system. These include under-differentiation of phonemes, over-differentiation of phonemes, reinterpretation of distinctions, and phone substitution (Weinreich, 1968). In addition to these four basic types, phonological processes, phonotactic interference, and prosodic interference also occur. Shibboleth is able to use phone substitution, under-differentiation of phonemes, some types of phonotactic interference, and some cases of reinterpretation of distinctions. Syllable structure changes are more difficult, due to automatic syllable alignment issues. Shibboleth cannot use over-differentiation of phonemes or prosodic interference at this time.

In addition to transfer from a NNS's L1, language universals also play a part in the production of foreign accent. These are substitutions that are not explained by transfer from the L1, but are explainable in terms of first language acquisition. For example, in one study, a six year old Icelandic child used developmental processes more often than transfer processes to produce novel affricates and fricatives while learning English as an L2. However, he also seemed to use transfer from his L1 to determine the difficulty of the novel segments (Hecht & Mulford, 1987). Developmental processes can also be used to explain Saudi Arabian Arabic speakers in a study by Flege (1980) whose voice onset times for stops gradually became more native-like as their L2 proficiency in English increased. This gradually improving approximation also occurs in L1 acquisition of English stops (Macken & Barton, 1977). It is likely that L2 speakers of higher proficiency will tend to use universal substitutions more often than transfer from the L1, and that these substitutions will gradually decrease as the speakers gain proficiency (Major, 2008). Shibboleth is able to use universal substitutions at the segmental level.

2.2 Related Work

While there have been a few automated accent recognition projects attempted, none were identical to what is done in Shibboleth. The most similar precursors to this work are the STAT system at George Mason University (GMU) (Kunath & Weinberger, 2009) and the accent classification work done by Angkititrakul and Hansen (2006). STAT also made automated comparisons between written phonetic transcriptions from the GMU Speech Accent Archive (Weinberger, 2013) and phonological speech patterns of a NNS with English as an L2, but, unlike Shibboleth, no classification attempt of a speaker with an unknown L1 was attempted. The classification system used by Angkititrakul and Hansen (2006) operated similarly in theory to Shibboleth, but a larger amount of data was needed for training and a different classification algorithm was used over a smaller number of languages.

Other systems, such as the dialect recognition program at the Lincoln Laboratory at the Massachusetts Institute of Technology use acoustic data and a very small number of dialects or languages are analyzed (Shen, Chen, & Reynolds, 2008). When using acoustic data, the analyst has to derive the useful phonological data from the proposed acoustic difference suggested by the system. Additional similar approaches include language or dialect recognition from acoustic data (Campbell, 2008; Choueiter, Zweig, & Nguyen, 2008) and individual speaker recognition from acoustic data (Campbell, Campbell, Reynolds, Singer, & Torres-Carrasquillo, 2006).

2.3 Participants and Data

The Shibboleth project currently relies on samples of speech gleaned from the GMU Speech Accent Archive (2013), compiled by the director of linguistics at GMU, Dr. Steven Weinberger. The archive is composed of both native and non-native English speakers each reading the same 69-word passage in English, discussed in section 3.1.6.1. This passage was written by Weinberger to maximize the number of phonemes and phoneme combinations in a relatively short utterance, though it does not contain data for all phoneme combinations common in American English.

The samples in the Archive were collected by means of personal interviews, unmonitored recording on the GMU campus (a recording device was set up in a hallway near the linguistics department), and through samples submitted in email over the internet. Since there were many different collection techniques used by several individuals over a span of 15 years, the manner and means of recording varied greatly. The participants were also more likely to be individuals who had access to a computer with internet access and audio recording capabilities. They were also more likely to be students at GMU or a friend or relative of a student at GMU than the average member of the population.

The transcribers of the data were graduate students at GMU in an advanced phonology class with at least one previous semester of phonological transcription. Each transcription was done by two students and Weinberger, with disagreements handled by group discussion until a transcription could be agreed upon. Each student did three to four transcriptions over a semester, and, in the latter years of the project, used programs such as Praat (Boersma & Weenink, 2013) for assistance.

The 1542 participants in the Speech Accent Archive range in age from five to 97 years, with 698 female participants and 844 male participants. The participants' average age was 33.1 years at the time of recording. English is the L1 for 449 of the participants and an L2 for the other 1,093 participants. The average age of English onset for the NNSs is 12.3 years. The specific demographics of the subset of the Archive participants used in each Shibboleth classification study are shown within Chapter 4, Language Classification Experiment, and Chapter 5, Language Family Classification Experiment.

CHAPTER 3

DESIGN AND PROCESS

The Shibboleth program is comprised of two main modules, the rule-set creation module and the speaker comparison module. This design is suitable for practical reasons, since it means either module can be altered and improved independently of the other, as long as the expected output of the rule-set creation module and the expected input of the speaker comparison module remain compatible. Easy, independent alteration of the modules means that if, for example, there are advances in classification algorithms, a new algorithm can be plugged into the speaker comparison module with no or minimal changes to the rule-set creation module. Likewise, if there are improvements to the input of the rule-set creation module, it will not greatly affect the speaker comparison module. At the present time, the output from the rule-set creation module, a set of Microsoft Spreadsheet Format (XLS) files, must be input into the speaker comparison module as a separate, non-automated process by the user. Eventually, the data transfer process between modules will be streamlined to be automatic, so it will appear to the user as one inclusive procedure.

3.1 Rule-Set Creation Module.

The rule-set creation module's main function is to create a rule-set consisting of a collection of possible rules which, when taken together, illustrate how a non-native accent in a language or dialect differs from the native-like pronunciation of that language or dialect. This accent could be due to negative transfer from the NNS's native language or dialect or it could be due to developmental processes based on language universals. Negative transfer at the level of phonic interference will be one of my main areas of interest. As mentioned earlier, I will be able to use phone substitution, under-differentiation of phonemes, some types of phonotactic interference, and some cases of reinterpretation of distinctions in this system. I will not be finding rules based on the over-differentiation of phonemes, prosodic interference, or syllable structure changes at this time. I will also be able to use universal substitutions at the segmental level in this program, though most other developmental processes are not as easily captured.

The rules created are described as a difference between articulatory features in the speech of a speaker in the training group and a reference text. No acoustic features are taken into account at this time, largely because the data set I am using did not contain acoustic features.

A *rule-set* is any collection of rules that describes a speaker's or a group of speakers' feature changes when compared to a given reference sample. A rule-set for a language, family, or dialect is created from the changes of multiple speakers from the same native language, family, or dialect speaking the same L2 (e.g. multiple native German speakers speaking American English). The rule-set characterizing the accent of these speakers in the L2 will comprise the training rule-set for this L1 when they are speaking the same L2.

All of the phonetic differences between a speaker and a reference sample are represented in a spreadsheet (shown in Appendix C), which includes the articulatory features of a phone before it was changed (the input), the articulatory features of the phone after the change (the output), the environment (i.e. phone) preceding the phone in question (the pre-environment), the environment following the phone in question (the post-environment), how frequently this change occurred per speaker and per rule-set, and when a phone was inserted or deleted (a special category of change).

3.1.1 File types and compatibility. The rule-set creation module takes as input strings of International Phonetic Alphabet (IPA) characters encoded by Unicode escape sequences and stored in Rich Text Format (RTF) files and gives as output a Comma Separated Value (CSV) file. The IPA text is a broad transcription of an audio file. The transcription is created by a team of transcribers, in the case of speech samples to be included in the rule-set, or the IPA text from a dictionary or other language standard, in the case of the reference transcript. A future goal for the program is the capability of the rule-creation module to accept audio files as input in addition to written transcriptions.

The rule-set creation module is written in Java, which enables it to run on almost every computer platform that is currently available, as well as being possible to port to Android-enabled smartphones and tablets.

3.1.2 Phones and phone representation. In this paradigm, phones are always viewed as a collection of features, never as a single, inseparable entity. These features are represented in three different ways: the full representation consisting of place of articulation, manner of articulation, voicing and/or rounding; all partial representations of these features; and a consonant/vowel distinction. These multiple representations for the same phone change are to ensure that Shibboleth captures what causes a change in a speaker's phonetic output in a given instance. For example, final stop (or obstruent) devoicing is a common quality of native German speakers speaking American English. This devoicing is not a characteristic of American English when spoken by native speakers, so it should register as a characteristic that deviates from standard American

English. However, previous research (Brockhaus, 1995) indicates that word-final voiced stops trigger the devoicing. It is not possible to determine from limited data if the trigger for this occurrence of negative transfer is the word position, the manner of articulation, the place of articulation, the voicing, or some combination of these factors. As many of the feature changes that comprise a foreign accent are not supported by the extensive research that this particular change does, all combinations of possible feature changes are collected by Shibboleth and ultimately compared to pinpoint what constitutes a change, and what features of the phone and surrounding environments trigger that change. This combinatorial data is necessary to discover what a change actually involves. Without instances of multiple native German speaker devoicing all word-final voiced stops ([b], [d], and [g]) after a wide combination of pre-environments when speaking American English, it would be uncertain that "devoicing word-final stops" is the change, and not, instead, some more restrictive version of it, such as "devoicing all word-final voiced velar stops preceded by an [ɑ]," as could arise if the only sample was an example of one person changing the pronunciation of [fɹɑg] to [fɹɔk].

The types of features that are collected for consonants include place of articulation, manner of articulation, and voicing. The types of features that are collected for vowels include openness, centralness, and roundedness. The specific features used for both types are enumerated in Figure 1. A representation of the phone [ɛ] is shown in Figure 2.

Place of articulation	**Manner of articulation**	**Voicing**	**Open**	**Central**	**Roundedness**
Bilabial	Stop	Unvoiced	Close	Back	Unrounded
Labiopalatal	Tap	Voiced	Semi-close	Near-back	Rounded
Labiovelar	Affricate		Close-mid	Central	
Labiodental	Fricative		Mid	Near-front	
Dental	Nasal		Open-mid	Front	
Alveolar	Trill		Semi-open		
Alveolo-palatal	Approximant		Open		
Palato-alveolar					
Retroflex					
Palatal					
Velar					
Uvular					
Glottal					
Pharyngeal					

Figure 1. Articulatory features collected for consonant and vowel representation in Shibboleth.

Vowel Feature 1 (Openness)	**Vowel Feature 2 (Centralness)**	**Vowel Feature 3 (Roundedness)**
Open-mid	Front	Unrounded

Figure 2. Specific articulatory features collected for [ɛ].

Glides are encoded as consonants rather than as vowels in this algorithm. Affricates are recognized as one phone, while diphthongs are treated as two independent phones. This inconsistency is due both to the lack of Unicode characters for diphthongs, as well as the difficulty of representing diphthongs within the feature constraints of the other phones. While affricates can be represented, more or less, by a single place and manner of articulation, diphthongs represent a transition between two positions. Since there is no satisfactory way to appropriately encode the phenomenon of transition within a single phone using the current Shibboleth framework, I opt for the two-phone representation of diphthongs.

There is a similar issue with transcribers' representations of long vowels in the data used for this study that I, by necessity, treat differently. Over the years that the transcripts have been created, the standard used by the GMU linguistics department for long vowels has changed. The result of this was that some transcripts represented long vowels with the [ː] symbol (Unicode number 02D0), some transcripts represented them with the [:] symbol (Unicode number 003A), and some doubled the vowel itself. I chose to assume that all doubled vowels were, in fact, long vowels, and these three forms were interpreted as long vowels in Shibboleth. However, there was another complication in describing vowel length. Not all transcribers consistently chose to describe vowel duration. While vowel length is easy to implement in Shibboleth, I chose not to implement it at this time, given the irregularity and inconsistency of the transcriptions in this regard. I chose to ignore this feature which might have resulted in irregular analyses and representations and perhaps produced erroneous results.

Shibboleth, as mentioned, can perform comparisons between broad transcriptions of speech. By this, I mean that all phones that can be encoded in IPA are included in the speech comparison, but all diacritical markings and suprasegmental features that may have been included in the transcription are removed before the comparison takes place. This is due to further inconsistency in the transcriptions. Without a very high level of transcription regularity, mathematical comparison of the probability of a diacritic occurring, not occurring, or changing cannot be done.

In addition to the obvious drawback of not retaining as informative a description of the utterance, in some cases, this also has the drawback of being biased toward L1s whose phonemic inventories do not contain any diacritical markings in the phonemes, such as English. Languages such as Hindi, that have a phonemic differentiation between aspirated and unaspirated stops, affricates, and taps, would not serve as accurately as reference languages due to this limitation. While it is not a structural limitation of the program (code related to removing diacritical markings can be taken out easily), it makes sense in the short term to retain this constraint, as the data at my disposal uses English as the reference language.

In the long term, I am working toward automatic transcription from a spoken utterance, which should produce transcriptions that are regular across speakers with no human transcriber needed. This will also enable analysis of vowel length changes between speakers and contain the diacritical markings necessary to represent phonemic inventories of languages where diacritics represent a phonemic distinction.

3.1.3 Phone change and rule representation. The core of every rule is the phone change. Without a phone change, there will be no rule created (hence why it is impossible to capture positive transfer from an L1 or recognize over-differentiation of phonemes – I cannot tell *why* a speaker does not make a phonetic change in a specific instance, only that they did not). Shibboleth makes no distinction between phonemes and allophones for a language, so even a native speaker of a language using a different allophone than what is encoded as the "standard" will trigger rule formation.

The rules that Shibboleth creates to describe phonetic changes are based on the format A->B/C_D, where A is the input, B is the output, C is the preceding environment, D is the succeeding environment, and the underscore represents the positioning of the input and eventual output (Chomsky and Halle, 1968). Each of the variables may correspond to one of the three types of representations mentioned previously: every distinctive feature of a phone, some of the phone's distinctive features, or simply the consonant/vowel distinction.

The program creates the rules by comparing each phone a speaker utters to the phone that occurs at the same location in the reference sample. Each time these phones do not match, it will create rules from each possible feature combination, or *feature set* of the phone, that could have caused that change, in addition to all feature sets of the phone's environment that could be triggering the change. Shown in Figure 3 are examples of some of the possible changes if an /ɛ/ changes to /ɪ/, as in "pen" [pɛn] becoming "pin" [pɪn]. The full set of changes is not shown.

	ɛ				**ɪ**	
open-mid	front	unrounded	→	semi-close	near-front	unrounded
open-mid	null	null	→	semi-close	null	null
null	front	unrounded	→	semi-close	null	null
open-mid	front	null	→	null	near-front	null
null	null	unrounded	→	null	near-front	unrounded
null	front	null	→	semi-close	null	unrounded

Figure 3. Subset of possible feature sets to describe the /ɛ/ and /ɪ/ merger in Southern American English.

Even though creating combinatorial representations of every phone change is necessary, this leads to a large number of possible inputs and outputs for every phone change. For example, when considering the common merger of /ɛ/ and /ɪ/ (Labov, Ash, & Boberg, 2006) in Southern American English, as demonstrated by an average native Southern American English speaker who changes the pronunciation of [pɛn] to [pɪn], 62 combinations of the feature sets of the inputs and outputs would be created in Shibboleth (Appendix B).

Theoretically, the largest possible number of proposed combinations of the input and output feature sets created by a phone change would be 64, though this many combinations are rare. Since there are three different feature slots for both the input and output and all of the combinations within a mathematical set are determined by 2^n, there are eight possibilities, 2^3, of feature sets that can occur as the input and eight, 2^3, possibilities of feature sets that can occur as the output. All further combinatorial results in this thesis are derived similarly. Multiplying the possible inputs by the possible outputs according to the rule of product produces 64 possible combinations of feature sets. However, the combinations V->V and C->C will not occur because those are not changes. The other possibility, V->C, occurs extremely infrequently because Shibboleth usually does not align vowels with consonants. It generally considers the probability of a vowel deletion followed by a consonant insertion (two separate changes) more likely than a vowel in the reference sample aligning with a consonant in a speaker's input sample. The same is true for the combination C->V. In describing the change [ɛ]->[ɪ], the two changes not seen are V->V and V->C.

Whenever the *null* feature is used in the feature set on the left-hand side of the equation, it denotes that that particular feature of the phone may be unnecessary to trigger the change. For example, the change (bilabial, null, voiced)->(bilabial, stop, unvoiced) says that a voiced bilabial phone, regardless of manner of articulation, changes to a unvoiced bilabial stop. Whenever the null feature is used in the feature set on the right-hand side of the equation, it means that this feature was not changed due to the feature set described on the left-hand side of the equation. For example, the change (bilabial, stop, voiced)->(null, stop, unvoiced), says that a voiced bilabial stop will change to a unvoiced stop, which may or may not be bilabial, and that the place of articulation does not depend on the input of a voiced bilabial stop.

Even though Shibboleth creates 62 proposed changes to accurately express what could be happening when [ɛ]->[ɪ] in Southern American English, it is not suggested that all 62 proposed changes are correct. In fact, these changes are mutually exclusive, and only one should be correct. However, until Shibboleth determines which change to assign the highest likelihood to, likely through the input of more data from more occurrences of this phone change, these 62 changes simultaneously exist as inputs and outputs in the rule-set for Southern American English.

3.1.4 Pre- and post-environment. Although locating a phone change is the most important step in creating a rule, taking note of the phones that constitute the environment of that change is nearly as crucial. In an earlier example, stop devoicing occurs specifically in the word final position for some native speakers of German speaking American English. This devoicing does not occur everywhere a stop occurs in speech. In this case, the post-environment of the phone change rule would be #, to denote word-final position.

The pre- and post-environments on either side of a phonetic change are very similar in that they can possess similar feature sets as the feature sets of the input phone and output phone, with two added types. In addition to a full feature representation, partial feature representations, and a consonant/vowel distinction for each environment, environments can also contain the symbol '#,' to denote the word-initial or word-final position, and the symbol *NULL*. NULL has two uses. The more common use indicates that the segment so marked is unimportant to the change. This NULL is not quite functionally equivalent to every feature of a phone being represented by null. Rather, it means that the presence or absence of a particular phone, feature, or word boundary in that position simply does not matter. To return to the previous example, native German speakers devoicing word-final stops are triggered by the presence of a voiced stop in word-final position. The pre-environment to the voiced stop is irrelevant for this rule, or NULL.

The second way NULL is used is when a phone is inserted and it is preceded or followed by another change, insertion, or deletion. In this case, the phone change is taking place prior to an environment which did not previously exist. Since, computationally, the non-existence of a phone means that that phone did not trigger the change, it still follows the main use of NULL discussed previously, to denote that the phone change is not dependent on that particular environment. This does not happen frequently in the data. Phone changes are much more common than phone insertions, and phone insertions preceded immediately by another phone change are not common. The average number of rules that create or use a new environment in the rule-sets for the language classification experiment and the language family classification experiment is roughly 10%, though this number varies by native language or language family.

It should be noted that phone changes and insertions can be anticipated by the speaker, changing a prior phone by assimilation (Bloomfield, 1984). Due to computational limitations, however, Shibboleth assumes that something that is inserted or deleted later does not affect earlier distant phonetic changes. In addition, without knowing what the speaker's L1 is, it is impossible to follow the series of interlanguage changes that may have taken place, even if Shibboleth did not operate along a strictly sequential paradigm of change. Since Shibboleth's goal is to ascertain what a speaker's native language or dialect is from an unidentified accent, I cannot simultaneously use the native language or dialect in judgments about which rules should be used. Thus, assigning an order to the process of rule creation is necessary, leading to this rare use of NULL.

After the pre- and post-environments, the input and output have been considered, for the sound change we have been looking at thus far, namely the [pɛn]->[pɪn] change, the rule appears as in Figure 4. The twelve features (a_n-l_n) correspond to the twelve vectors used by the support vector machines (SVMs) discussed in section 3.2.3 in the speaker comparison module.

Input (ɛ)				**Output (ɪ)**		
Feature 1 d_1	Feature 2 e_1	Feature 3 f_1		Feature 1 g_1	Feature 2 h_1	Feature 3 i_1
open-mid	front	unrounded	→	semi-close	near-front	unrounded

Pre-environment (p)				**Post-environment (n)**		
Feature 1 a_1	Feature 2 b_1	Feature 3 c_1		Feature 1 j_1	Feature 2 k_1	Feature 3 l_1
stop	bilabial	unvoiced		nasal	alveolar	voiced

Figure 4. Twelve features (a_n-l_n) in the feature sets corresponding to twelve SVM vectors.

Including the pre- and post-environments of a phone change creates an even larger number of feature set combinations to consider. While the number of combinations of inputs and outputs is maximally 64, this must be multiplied by the number of feature set combinations of both environments, which is calculated using the combinatorial process previously discussed. The minimum number of environments is four, as outlined by the process in section 3.1.3. This occurs when the phone change has no surrounding environment besides # and NULL. Each environment can have a maximum of eight feature sets based on phone features, then an additional option for NULL, which creates up to nine feature sets for each environment of the phone change, for a combined 81 options between the pre- and post-environments. Multiplying the maximum combinations of phone changes, 64, by the maximum number of options for the environment, 81, gives the maximum number of rules that can be created based on one phone change, i.e. 5,184 rules. While having the maximum number of rules for a phone change may be unusual, even very normal changes, such as [pɛn]->[pɪn], can easily have almost as many. In this particular case, 62 change possibilities for [ɛ]->[ɪ] multiplied by 81 possible environment descriptions for [p] and [n] results in 5,022 rules. Ideally, Shibboleth would be able to determine, given enough data, which one of those 5,022 most accurately describes the phonetic rule that takes place, [ɛ] changing to [ɪ] in front of the nasal stops. Since every NNS of a language will likely make more than one phone change in an utterance of any length, the total number of resulting rules would easily be too large for a human researcher to create and study without some measure of automated analysis.

3.1.5 Frequency of change. Shibboleth stores four types of frequency information on each rule: the number of times a specific change in a specific environment was made by all speakers in a group, the number of times each individual speaker of a group made the change in that environment, how many times it was possible for the group to have collectively made the change in that environment, and how many times the environment and input for the change occurred in the reference text. Examples of this can be seen in columns P through Q and T through Z in Appendix C.

These four measures are necessary to measure how frequently a particular rule is used by a group of speakers, and to measure if the rule usage is regular across the group or confined to a small number of group members.

3.1.6 Process of rule and rule-set creation. The process of generating a rule-set that can be used in the Shibboleth speaker comparison module can be broken down into three phases: before, during, and after rule creation. Before the program is run and to begin generating a rule-set, the user must input the transcript of a standard native speaker's accent in an L1, or the reference transcript, and transcripts of NNSs from the same L2 speaking the L1, or the speaker transcripts. Next, both automatic and computer-aided human pre-processing must be done on the transcripts. Finally, it is widely known that alignment is one of the more difficult tasks. Shibboleth performs it by using a modified version of the ALINE algorithm (Kondrak, 2000), followed by further user editing of any necessary transcripts. During rule-set creation, Shibboleth notes each phone change, new rules are created for new changes, incremented for previously seen changes, and the frequency statistics are updated. After rule-set creation for a speaker or group of speakers, Shibboleth gives the user information about the more unique phone changes discovered in the data, including insertions, deletions and rare cases of vowels changing to consonants or vice versa. At this point, the user can make any necessary edits to the transcripts and rerun the module. Finally, after the user is satisfied that the output correctly reflects the phone changes that occurred between reference and speaker transcripts, the rules must be sorted by frequency before they are input into the speaker comparison module.

3.1.6.1 Reference transcript. Since Shibboleth works by comparisons to a baseline, some baseline standard version of a native speaker's pronunciation of their language must be input. As the data source used in this study, the GMU Speech Accent Archive is currently focused on comparisons to American English, and the reference transcript contains phrases in American English. For the purposes of this investigation, the Northern American English accent as defined by Labov et al. (2006) is the representative accent for a native American English speaker. This accent is common throughout the northern area of the United States in New York (excepting New York City), Michigan, Wisconsin, northern Illinois, Iowa, Minnesota, North Dakota, Vermont, Connecticut, western Massachusetts, and St. Louis. This dialect was chosen for two reasons: practicality and lack of markedness. The Northern American English dialect is one of the least marked of the American English dialects, with the most unmarked accent belonging to the Western American English dialect. However, most American dictionaries and standardized pronunciation guides base their pronunciation upon the Northern American English accent, with Midland and Southern pronunciations as marked secondary and tertiary options. Because of this, I created a reference transcript for a phrase where the standard speaker is speaking the Northern American English dialect as characterized by the Random House pronunciations, given on Dictionary.com (2013) as the standard. Since most major world languages as well as many less common languages have a dictionary that includes standardized pronunciations, this method of creating a reference sample can remain consistent when I need to create reference transcripts for languages and dialects other than American English.

It is possible to automate the creation of a reference transcript using standardized reference materials. This means that eventually automatic reference creation will be part of the solution to entering extemporaneous speech into Shibboleth for training and testing. The capability to collect informal speech would allow many more phone combinations that naturally occur to be captured more easily for study. Obtaining casual utterances also would create the ability to ascertain the accent of L2 speakers in very close to real-time without the necessity for reading fluency in the L2 or requiring a specific utterance for testing the speaker.

In cases where the dictionary definition gives alternate pronunciations, generally the first, most common, pronunciation is used. However, in cases where two alternate pronunciations are given for the stressed and unstressed forms of the word (e.g. *the* – [ði], [ðə]), the form recommended by the dictionary for the situation is chosen.

The reference transcript for American English is created using this method. It is shown along with the elicitation paragraph used in this study in Figure 5. This transcript is stored as a string of IPA text in an RTF file. This requires the use of a Unicode font that contains IPA characters (the preferred method), or an RTF editor with the ability to enter Unicode encoding for the IPA characters. For this study, I used the Doulos SIL font (SIL International, 2010) in Microsoft Word 2010.

Please call Stella, ask her to bring these things with her from the store: six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake and a big toy frog for the kids. She can scoop these things into three red bags and we will go meet her Wednesday at the train station.

[pliz kɔl stɛlə æsk həɹ tə bɹɪŋ ðiz θɪŋz wɪθ həɹ fɹəm ðə stɔɹ sɪks spunz əv fɹɛʃ snoʊ piz faɪv θɪk slæbz əv blu ʧiz ənd meɪbi ə snæk fəɹ həɹ bɹʌðəɹ bɑb wi ɔlsoʊ nid ə smɔl plæstɪk sneɪk ənd ə bɪg tɔɪ fɹɑg fəɹ ðə kɪdz ʃi kən skup ðiz θɪŋz ɪntə θɹi ɹɛd bægz ənd wi wɪl goʊ mit həɹ wɛnzdeɪ æt ðə tɹeɪn steɪʃən]

Figure 5. Elicitation paragraph used in Speech Accent Archive data collection, followed by the IPA reference transcript for this text in Northern American English dialect.

While it is convenient and quick to compile a transcript of how a phrase would be pronounced by the hypothetical standard native speaker of a language, this method of creating a reference transcript word by word using a dictionary or other formal standardized material has the drawback of generating a transcript with artificially precise lexical segmentation where it would not necessarily occur in speech. An example of this will be shown when discussing word alignment.

Another factor to keep in mind when creating a reference transcript is that even though most languages have a number of common associated dialects, there is usually only one standard form of the language. However, pluricentric languages, such as English, French, Spanish, Portuguese, and Serbo-Croatian have no single standard form (Clyne, 1992). Rather, there are two or more standard forms, usually following geopolitical or cultural boundaries. With English, for example, there can be a clear distinction made between standard American English, British English, South African English, and the English of the Oceanic region. There are, of course, several dialects that diverge from these standards within each geographic region. For pluricentric languages, there will never be only one reference transcript, but, instead, there will be separate reference transcripts for each major variety that the Shibboleth user wishes to test as an L2.

3.1.6.2 Speaker transcripts. The transcripts input for building classification rule-sets (training transcripts), the transcripts input for testing unknown speakers to determine their native language or dialect (testing transcripts), and the reference transcript as identified above are identical in format. Speaker transcripts (any transcript which comes from a speaker and not a standardized source) are also stored as a string of IPA text in individual RTF files, one for each utterance. The utterance in the training transcripts and the testing transcript should both match the utterance from a reference transcript. As the language classification experiment and the language family classification experiment were conducted on transcripts from the GMU Speech Accent Archive, all data was compared to the same reference transcript, shown in Figure 5. Though the training and testing transcripts must match a reference transcript (otherwise there is no direct comparison possible), it is not theoretically necessary that there be only one reference transcript used. Currently, however, Shibboleth can only use one reference transcript per rule set. If the training transcript phrases are all as short as the one used for the Speech Accent Archive, it is ideal to have additional phrases spoken and additional matching reference transcripts created; consequently, later work on Shibboleth will entail adding the capability of using different reference transcripts per rule-set. The process of transcribing the speaker transcripts for the Speech Accent Archive was described in more detail in section 2.3.

3.1.6.3 Pre-processing transcripts. Reference and speaker transcripts must be standardized via pre-processing before they can be used for the sake of speech comparison or classification. Preparing the transcripts for analysis is a step that is easily overlooked when considering the process of locating differences in accent and creating possible phonetic rules to describe accents, but it can also be a very difficult step to automate. Currently, part of the pre-processing necessary for Shibboleth input has been automated, but there are some critical elements that still require a human.

The automatic pre-processing of transcripts by Shibboleth includes removing unnecessary white space, removing diacritical markings, converting Unicode encoding into the appropriate Unicode to represent IPA characters, standardizing vowel length, removing any unnecessary text and markup added by an outside transcript editor, creating a transcript editor for the human portion of pre-processing, and creating an array of every possible feature combination of the phones that exists in the reference text.

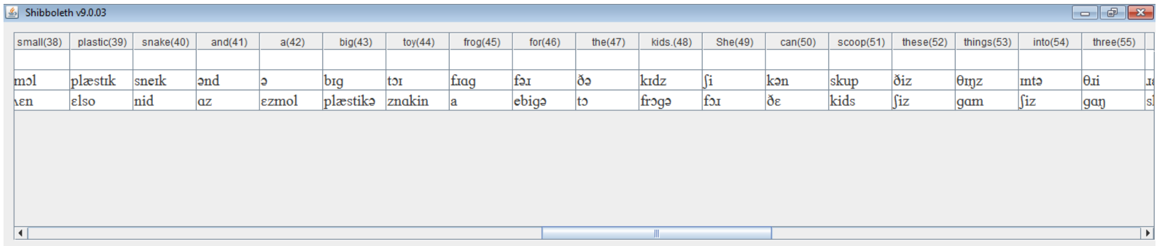
Shibboleth reads in data from the RTF file using the built-in Java class RTFEditorKit. This class removes unnecessary markup and state information added by outside editors. This data was not purposely added by the user. Many word processing programs, such as Microsoft Word, have markup information, state information, and metadata included invisibly in every document even if the file is saved as an RTF file. This sort of information is not unique to Microsoft Word and is not pertinent to the RTF text, so it is automatically removed. The first step Shibboleth does after receiving the RTF text from the RTFEditorKit is to clean up the file itself. This includes replacing all Unicode encoding that does not correspond to IPA characters but *looks* like it might to a human transcriber, such as the Cyrillic form of æ, with the Unicode number of 04D4, as opposed to the Latin form used by IPA, æ, with a Unicode number of 00E6. Cleaning up the file also includes removing white space (extra spaces, tabs, line breaks, and non-break spaces) or text outside of the '[' and ']' symbols that usually denote a complete transcript in the Speech Accent Archive. If the string is missing these characters, the beginning of the string of input text is considered to be the beginning of the transcript and the end of the string is considered to be the end of the transcript. The resulting string of text that represents the transcript should have no outstanding formatting issues and should be made up of Unicode encoding that corresponds to IPA characters. Further changes will create a more regular character format which will correct for some of the irregularities caused by human transcribers.

The next step is for Shibboleth to combine all appropriate adjacent characters that have been encoded as separate phones into their corresponding phones, namely affricates, as discussed in 3.1.2. This is closely followed by the previously mentioned encoding of all long vowels, no matter the representation, as a single vowel of identical representational length. Finally, all Unicode encoding that corresponds to diacritical markings is removed. Rarely, there are some Unicode characters that have been entered by a human transcriber that are not recognizable as a common accidental substitute for any IPA character, such as the symbol ©. Shibboleth gives an error message to the user so that he or she knows there are potentially faulty characters in the transcription.

The last step of pre-processing is for Shibboleth to create an array of all of the possible combinations of features of the phones in the reference text, both as environments and as phones that could be changed. This array can be used during processing of a speaker transcript to aid in the frequency calculations for phone changes. For example, a speaker may change [pɛn]->[pɪn], but without Shibboleth retaining knowledge of the features in the entire reference text, it is hard to know if that is significant. The speaker may make the change 100% of the time that [ɛ] appears in the appropriate environment in the reference text, indicating it is a part of the speaker's accent, or the speaker may only do it once out of 20 possible instances, indicating it was perhaps a minor disfluency or transcriber's error. The array created in this step keeps track of all of the possible instances, leading to the frequency statistics found in Appendix C.

All other pre-processing will need to be done by a human, preferably one who also has access to the audio recording upon which the transcript is based. To aid the human with this processing, Shibboleth has a built-in editor to visually compare alignment and IPA characters, change the transcript with an onscreen IPA keyboard, and automatically save and reload the edited transcripts.

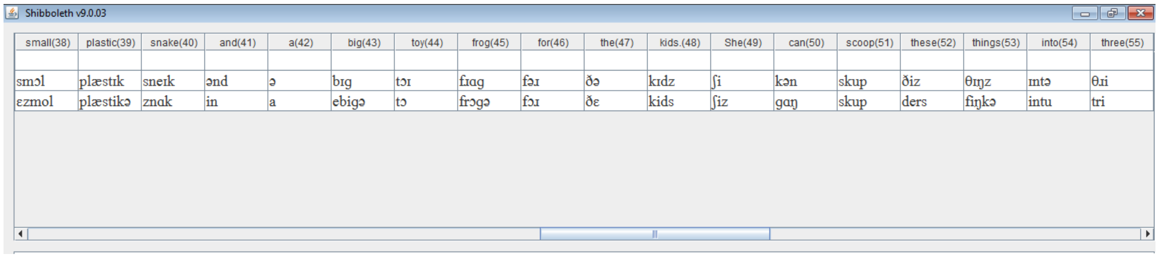
The most common problem requiring human pre-processing is that of speech disfluency which produces stammering, word deletion, word substitution, or word insertion during speech production, which is then expressed in the transcript. Most of the non-fluent speech does not hinder comprehension and would likely be overlooked by a human listener, but since we are not using an automatic word alignment algorithm, this causes difficulty with aligning speaker transcripts with the reference transcript. I found that about 20% of speakers in the tested languages in the Speech Accent Archive, including native American English speakers, have some disfluency in their speech sample. Further, removing duplicated words and incorrect contractions from speech is relatively simple for a human user of Shibboleth, but it is a difficult issue to automate. While the program can tell when a speaker transcript is the incorrect length in comparison to a reference transcript or when a phone has changed from a vowel to a consonant (a sign of an alignment issue, due to the rarity of the occurrence), the current version cannot correct the issue. Instead, Shibboleth displays the reference transcript and the speaker transcript in a table in the GUI, which will give the human user an easy visual aid to see where the alignment problems may lie. An example of mis-aligned data due to disfluency is shown in Figure 6.



small(38)	plastic(39)	snake(40)	and(41)	a(42)	big(43)	toy(44)	frog(45)	for(46)	the(47)	kids.(48)	She(49)	can(50)	scoop(51)	these(52)	things(53)	into(54)	three(55)	
mɔl	plæstɪk	sneɪk	ənd	ə	bɪg	tɔɪ	fɹɑg	fəɹ	ðə	kɪdz	ʃi	kən	skup	ðiz	θɪŋz	ɪntə	θɹi	ɹ
ʌɛn	ɛlso	nid	ɑz	ɛzmol	plæstikə	znɑkin	a	ebigə	tɔ	frɔgə	fɔɹ	ðɛ	kids	ʃiz	gɑm	ʃiz	gɑŋ	s

Figure 6. Mis-aligned data due to disfluency shown in Shibboleth GUI.

When a word is completely missing, the human processing the data adds a hyphen to the location of the missing word in the speaker text before running a comparison. When the speaker has repeated a phrase, word, or stuttered at the beginning of a word, the last utterance is used and all previous versions are deleted. Sometimes the speaker will create a contraction from two adjoining words (usually contracting “we will” [wi wɪl] to “we’ll” [wil] in the Speech Accent Archive). In this instance, the contraction is deleted and the two missing words are treated as deleted. Similarly, sometimes a speaker will conjoin two words that, in “perfect” speech, would be separate. This happens frequently in the sample with “need a” [nid ə] becoming “needa” [nidə]. In these instances, the human processing the data must make a judgment call on listening to the speech if the human transcriber was representing a standard speech pattern more accurately than the standardized reference. In this case the human Shibboleth user will separate the two sets of phonemes into their corresponding morphemes (e.g. “needa” [nidə] will get separated into “need a” [nid ə] for processing), or if the human transcriber was representing a feature of the non-standard speech (e.g. “we’ll” [wil] should not be matched to “will” [wɪl]). In this case, the human Shibboleth user will keep the phonemes together as one “word” for Shibboleth to process, and add a hyphen to represent the deleted word following or preceding the conjoined word, whichever is deemed closer to creating rules that represent the actual phonological change. Occasionally, the speaker apologizes for their disfluency during the utterance, and this is also deleted from the transcript. An example of the earlier mis-aligned text in Figure 6 is shown after human processing in Figure 7.



Shibboleth v9.0.03

small(38)	plastic(39)	snake(40)	and(41)	a(42)	big(43)	toy(44)	frog(45)	for(46)	the(47)	kids.(48)	She(49)	can(50)	scoop(51)	these(52)	things(53)	into(54)	three(55)
smɔl	plæstɪk	sneɪk	ənd	ə	bɪg	tɔɪ	fɹɑg	fəɹ	ðə	kɪdz	ʃi	kən	skup	ðiz	θɪŋz	ɪntə	θɹi
ɛzmol	plæstikə	znɑk	in	a	ebigə	tɔ	frɔgə	fɔɹ	ðɛ	kids	ʃiz	gɑŋ	skup	ders	fiŋkə	intu	tri

Figure 7. Correctly aligned text shown in Shibboleth GUI.

Frequently, examining alignment issues is all of the human pre-processing needed. Once in a while, the automatic pre-processing will return a character error when a Unicode character has been used in the transcript that does not correspond to any IPA character or even any character that is commonly mistaken for an IPA character. In these cases, the user needs to determine whether or not he or she can tell from context which IPA character was intended and, if no option seems plausible, delete the offending word (it is not possible to delete a single phone and properly process the word at this time).

3.1.6.4 Phone alignment. Shibboleth contains a modified implementation of an early version of the ALINE algorithm (Kondrak, 2000). This implementation was modified based upon an implementation of ALINE that was used within the STAT program at GMU (Kunath & Weinberger, 2009).

ALINE is an algorithm that was originally created for identifying corresponding segments of related phone sequences for cognate identification. It contains a function for comparing the difference between a pair of phones, based on a numerical value assigned to a feature and the salience of that feature, and assigns similarity or difference scores to phone pairs. Kondrak (2000) uses multivalued features to describe features along a continuum (i.e. a dental phone would receive a value of 0.9, which is significantly closer to the bilabial value of 1.0 than it is to the glottal value of 0.1, since the place of articulation for the bilabial and dental phones are physically closer in the oral cavity than the place of articulation for the bilabial and the glottal phones), and a salience value to describe how important that type of feature is to the matching algorithm. For example, manner and place of articulation have very high salience ratings, of 50 and 40, respectively, while aspiration and length have much lower salience ratings of five and one. This means that manner of articulation will rate much higher than length when matching phones. Matching is done by finding the optimal score for an input based on the algorithm in Figure 8.

```
algorithm Alignment
    input: phonetic sequences x and y
    output: alignment of x and y
    define $S(i,j) = -\infty$ when $i < 0$ or $j < 0$

    for $i \leftarrow 0$ to $|x|$ do
        $S(i, 0) \leftarrow 0$
    for $j \leftarrow 0$ to $|y|$ do
        $s(0, j) \leftarrow 0$
    for $i \leftarrow 1$ to $|x|$ do
        for $j \leftarrow 1$ to $|y|$ do
            $S(i, j) \leftarrow$ max(
                $S(i- 1, j) + \sigma_{skip}(x_i)$,
                $S(i, j-1) + \sigma_{skip} (y_j)$,
                $S(i-l, j-1) + \sigma_{sub} (x_i,y_j)$,
                $S(i-1, j-2) + \sigma_{exp} (x_i, y_{j- l}y_j )$,
                $S(i-2, j-1) + \sigma_{exp} (x_{i-l}x_i,y_j)$, 0)
    $T \leftarrow (1 - \varepsilon)$ x $\max_{i,j} S(i,j)$

    for $i \leftarrow 1$ to $|x|$ do
        for $j \leftarrow 1$ to $|y|$ do
            if $S(i, j) > T$ then
                Retrieve(i, j, 0)
```

Figure 8. ALINE algorithm for finding the optimal score for an input word.

Since ALINE was initially created for considering pairs of words based on a word list and comparing phones inside of these words, it only has the ability to match phones within words. ALINE cannot match words within utterances. Kunath and Weinberger's 2009 version of ALINE made minor changes to Kondrak's algorithm, some of which

Shibboleth uses, and some of which were reverted back to Kondrak's original algorithm. Shibboleth's version of the algorithm also contains edits to the methods for comparing two vowels, two consonants, and a mixed vowel/consonant pair. Other minor changes to the algorithm include feature values which were added and phones which were omitted by Kondrak (2000) and Kunath and Weinberger (2009). The feature values and salience ratings are shown in Figures 9 and 10, respectively.

Feature name	**Phonological term**	**Numerical value**
Place		
	bilabial	1.0
	labiodental	0.95
	dental	0.9
	labiopalatal	0.86
	alveolar	0.85
	labiovelar	0.82
	retroflex	0.8
	palato-alveolar	0.75
	alveolo-palatal	0.73
	palatal	0.7
	velar	0.6
	uvular	0.5
	pharyngeal	0.3
	glottal	0.1
Manner		
	stop	1.0
	nasal	0.95
	affricate	0.9
	fricative	0.8
	approximant	0.6
	tap	0.55
	trill	0.5
Height		
	close	1.0
	semi-close	0.8
	close-mid	0.65
	mid	0.5
	open-mid	0.35
	semi-open	0.2
	open	0.0
Back		
	front	1.0
	near-front	0.85
	central	0.5
	near-back	0.15
	back	0.0

Figure 9. ALINE values for articulatory distinctive features.

Feature Name	Salience Setting
Syllabic	5
Voice	10
Lateral	10
High	5
Manner	50
Long	1
Place	40
Nasal	10
Aspirated	5
Back	5
Retroflex	10
Round	5

Figure 10. ALINE salience values for articulatory distinctive feature categories.

3.1.6.5 Change recognition and rule creation. Once Shibboleth has a reference transcript and a speaker transcript aligned and pre-processed, it begins the phone by phone comparison of the transcripts. It first pulls up the pair of words in the first position of the alignment and compares the Unicode characters of the first phones in these words. If these characters match, it does nothing else, and moves on to comparing the second phones in the first word. Similarly, if the initial phone value in the speaker transcript is '-,' indicating an omitted word, Shibboleth moves to the next word.

When Shibboleth finds a pair of phones whose characters do not match, or a Unicode character it recognizes as an insertion from the phone alignment, it converts the Unicode characters, as well as the characters on either side, the pre- and post-environments, into feature sets. The feature sets contain all of the combinations of features possible for each one of the six characters (the original phone pair with the change, plus both of their environments), based upon the four features used to describe vowels (openness, centralness, roundness, and vowel) and the four features used to describe consonants (place and manner of articulation, voiced, and consonant). There will be nine feature sets found for each environment that is a phone, two feature sets found for each environment that is word final or word initial position, and eight feature sets for each phone involved as input or output, as seen by the rows in Figure 11. The input for an insertion will have only one feature set, NULL, as will the output for a deletion. As previously mentioned, all other uses of the feature set NULL refer to the feature set in that location being irrelevant to the phonetic change. The types of features making up the feature sets are described in Figure 11.

Environment of Change					
Phone					
Consonant			Vowel		
Place of artic.	Manner of artic.	Voiced	Openness	Centralness	Roundedness
Place of artic.	Manner of artic.	-	Openness	Centralness	-
Place of artic.	-	Voiced	Openness	-	Roundedness
-	Manner of artic.	Voiced	-	Centralness	Roundedness
Place of artic.	-	-	Openness	-	-
-	Manner of artic.	-	-	Centralness	-
-	-	Voiced	-	-	Roundedness
C	-	-	V	-	-
NULL	-	-	NULL	-	-
Word final/initial					
#	-	-			
NULL	-	-			

Input/Output Phone					
Consonant			Vowel		
Place of artic.	Manner of artic.	Voiced	Openness	Centralness	Roundedness
Place of artic.	Manner of artic.	-	Openness	Centralness	-
Place of artic.	-	Voiced	Openness	-	Roundedness
-	Manner of artic.	Voiced	-	Centralness	Roundedness
Place of artic.	-	-	Openness	-	-
-	Manner of artic.	-	-	Centralness	-
-	-	Voiced	-	-	Roundedness
C	-	-	V	-	-

Figure 11. Types of features within the feature set for input phones, output phones, and their pre- and post-environments.

This means that for every word-internal phone change or insertion, there are 52 feature sets generated; for every change or insertion in word-initial or word-final position, there are 38 feature sets generated; and for every change of a phone that is a full morpheme, there are 24 feature sets generated. These are the sums of the input phone's feature sets plus its pre-environment and post-environment feature sets and the output phone's feature sets plus its pre-environment and post-environment feature sets. The feature sets for each Unicode character have been created independently of the information in the other feature sets, so these numbers will remain consistent, regardless of which IPA phone changes, or the feature similarity of this phone with the output phone.

Next, these feature sets that have been created are refined into rules. Rules are generated by combining four feature sets: the pre-environment of the input phone, the post-environment of the output phone, the input phone, and the output phone. Within the rules created, any rule having identical inputs and outputs is deleted. For example, by combining all feature sets, regardless of their contents, the input and output could both have a feature set containing only 'V,' for vowel. When the 'V's' are compiled into a rule, this compilation would create a rule containing no change, hence, it would be deleted. Similarly, the closer the input and output phones are, the more feature sets they would have in common. Just as Shibboleth does not need a rule that says V->V, it does not need a rule that says unrounded->unrounded.

However, this method of creating phonetic rules leads to some rules that are valid within Shibboleth, but not a common representation in phonology. Due to the program generating all of the feature sets and creating a complete set of combinations, there will be rules created that read, for example, (front, null, unrounded)->V/_#. Obviously, there is no common interpretation for that rule that makes sense. However, in reality, when an L2 contains a phone that was not in the L1 for a group of L1 speakers, they may all try to produce this phone in different ways, such as with the unfamiliar front, unrounded vowel in the example. The speakers that are unsuccessful in producing the front, unrounded vowel will likely produce vowels, but their outputs may vary. In Shibboleth, the interpretation would be “in the word-final position, an unrounded, front vowel is changed into a different vowel.” (In this case, the feature value “null” means that the feature of “height” is immaterial to the change.) Since a rule cannot be created with the same phone as input and output, the interpretation of the output is always as an implied difference from the input. This does not apply to the situation mentioned earlier of unrounded->unrounded. The feature that differentiates this change will appear in one of the other feature set combinations, such as (front, null, unrounded)->(near-front, null, unrounded) or (close-mid, null, unrounded)->(near-close, null, unrounded). “Vowel” and “consonant,” on the other hand, are categories that stand apart from the other feature combinations.

Once these new rules have been created, they are compared to a rule-set for the speaker containing all previously created rules. If a rule does not already exist in this rule-set, it is added to the rule-set, the value for the number of occurrences of this rule is set at one, and the input, pre-environment, and post-environment are matched against the phones from the reference transcript processing to determine how many times this change had the potential of occurring within this specific utterance.

If a rule does match against an existing rule in the rule-set, the number of occurrences of the rule is incremented by one and no further change is made to the rule-set. An example of the earlier rule from Figure 4 in the rule-set, along with the corresponding frequency information, is shown in Figure 12.

Input (ε)				**Output (ɪ)**		
Feature 1 d_1	Feature 2 e_1	Feature 3 f_1		Feature 1 g_1	Feature 2 h_1	Feature 3 i_1
open-mid	front	unrounded	→	semi-close	near-front	unrounded

Pre-environment (p)				**Post-environment (n)**		
Feature 1 a_1	Feature 2 b_1	Feature 3 c_1		Feature 1 j_1	Feature 2 k_1	Feature 3 l_1
stop	bilabial	unvoiced		nasal	alveolar	voiced

Number of occurrences	**Number possible**	**Number possible in passage**	**Occurrences for speaker**
4	5	1	1

Figure 12. Feature set with frequency information for the rule [ε]->[ɪ]/[p_n].

In the case where a phone was inserted, changed, or deleted directly after an inserted, changed, or deleted phone in the rule-set, it is difficult to represent the statistics for the corresponding environment. Even making the assumption that changes occur strictly linearly in time (which is not true, as stated earlier when referencing such phonological phenomena as assimilation), there is still the problem of how to count the frequency of the change. For example, if a phone, B, is changed from phone A between two existing phones, C and D, both of which did not change, it is easy to count how many times the feature sets of CAD occurred in the reference to get a number of how many times it was possible to change A in the environment C and D.

However, if the same phone, A, is changed into phone B after a deletion of phone C, what should the statistics for the environment of A be? Should it be phone previous to C or the word boundary that replaced C, which I will refer to as E, or should it be C? Neither option seems ideal. If one chooses to use the environments of E and D, it is possible EAD may not even exist in the reference transcript. If one chooses to treat the pre-environment of A as being the original C, even after it is deleted, and counts the occurrences of CAD, that is not taking into account the fact that at the time A changed into B, C did not exist and likely shouldn't have as much influence on A as those situations where C remains unchanged. Neither option seems likely to represent the appropriate frequency of the change.

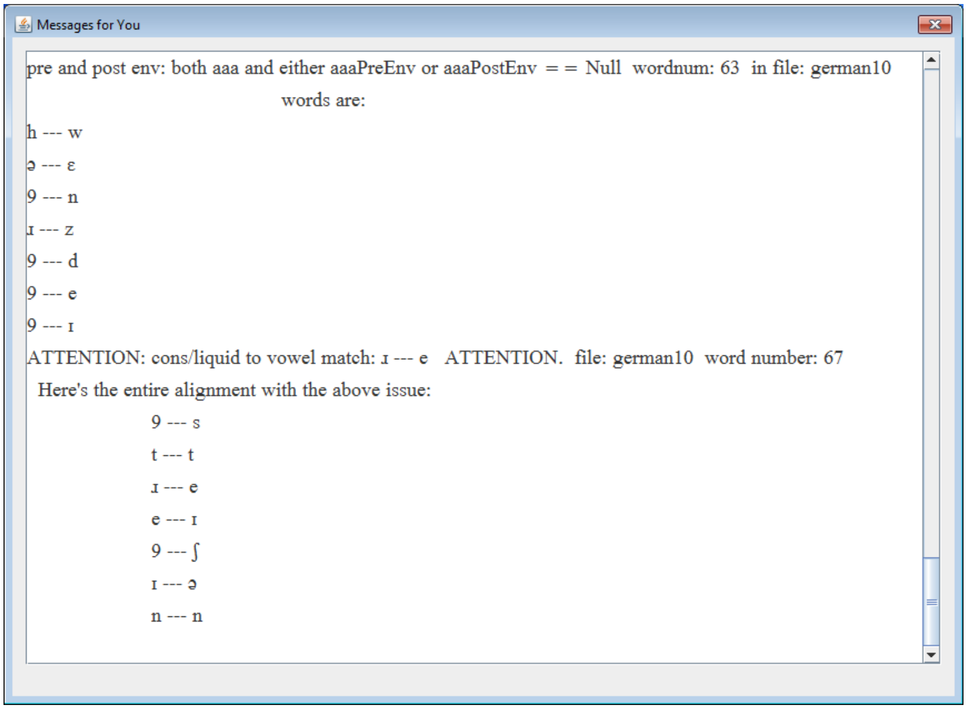
Since there is no satisfactory way to calculate the likelihood of an insertion, change, or deletion following an insertion, change, or deletion, the rules referencing these situations are created, but they are stored with default frequency values of -11111 in the rule-set. Since their frequencies are, thus, less than zero, these rules are not frequently used for speaker classification. Presently, this does not have that much impact on the classifications, as two changes in a row are rare in all of the data used so far, but effectively deleting these rules from the classification process is far from an elegant solution.

If a speaker is a native speaker of an unknown language, dialect, or language family that the Shibboleth user wishes to classify by comparing him or her to existing rule-sets, this is the last step of the Shibboleth rule-set creation module for this speaker. The user can save the rule-set and pre-process it for input to the speaker comparison module at this point.

If the speaker is going to be included in a classification rule-set, the last step is to add this speaker's rule-set to the rule-set for the entire language family, language or dialect. This is done by one last loop, which compares all of the rules in the rule-set to the existing larger classification rule-set, adding rules when they did not previously exist, and incrementing those rules that did exist, similar to the speaker's rule-set creation process. However, for a rule-set containing more than one speaker, Shibboleth also keeps track of which speaker exhibits each rule in the rule-set, as well as how frequently the rule occurs per speaker. This is a future safeguard against one speaker exhibiting a rule very strongly overriding several other L1 speakers of an L2 who do not exhibit the rule at all. In those cases, the rule should not be considered indicative of that particular L1.

3.1.6.6 Output of module and further processing. There are four different types of output from the rule-set creation module: 1) problems or irregularities during processing shown as a list of errors displayed in a pop-up window from the Shibboleth GUI; 2) normal processing information, including a list of all phones and words aligned and all rules created; 3) data output given to the user as a CSV file of the rules for input to the speaker comparison module; and 4) command line information, which shows the basic file input and output information from a command window.

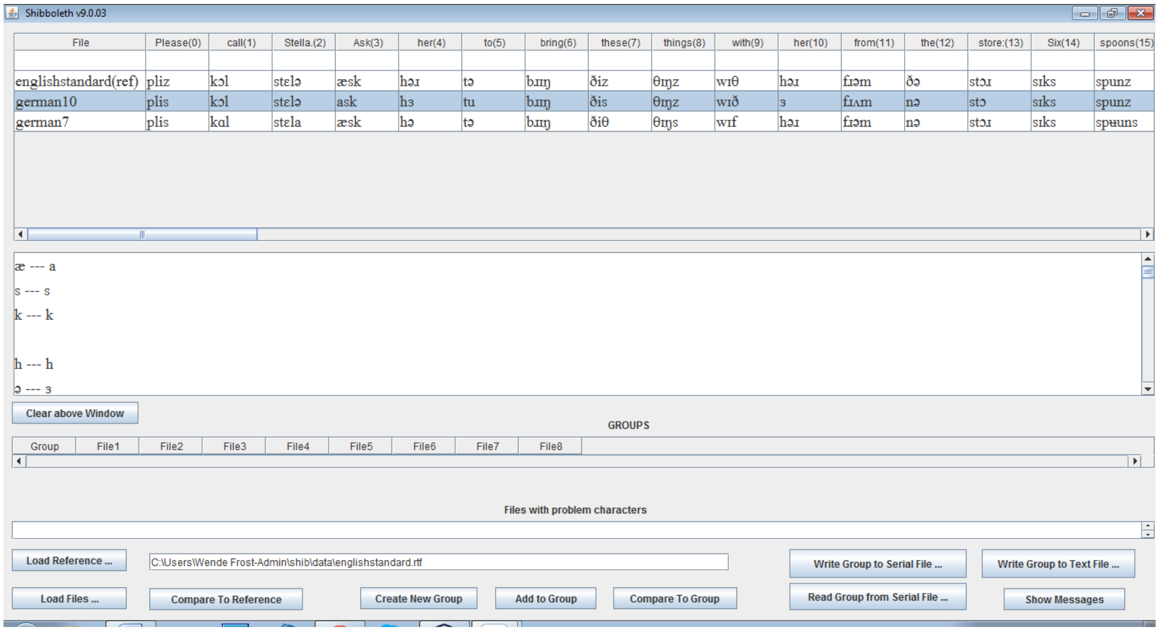
A pop-up window in the Shibboleth GUI displays all unusual alignments, words and phones skipped, and the location of the words or phones in the input. An example of this output is shown in Figure 13.



```
pre and post env: both aaa and either aaaPreEnv or aaaPostEnv = = Null  wordnum: 63  in file: german10
                                   words are:
h --- w
ə --- ɛ
9 --- n
ɹ --- z
9 --- d
9 --- e
9 --- ɪ
ATTENTION: cons/liquid to vowel match: ɹ --- e   ATTENTION.  file: german10  word number: 67
  Here's the entire alignment with the above issue:
                 9 --- s
                 t --- t
                 ɹ --- e
                 e --- ɪ
                 9 --- ʃ
                 ɪ --- ə
                 n --- n
```

Figure 13. Shibboleth GUI pop-up window showing unusual alignments.

The list of every phone alignment matched during Shibboleth's run for every speaker in the group is shown in the Shibboleth GUI output window. These are displayed as phones and not as a collection of features, as they are once they have been converted into feature sets for inclusion in rule-sets. An example of this phone output is in Figure 14. The rules that have been created are also displayed in the Shibboleth GUI. This feature is useful for debugging individual phones or morphemes before they are saved in a CSV file or added to a group. The rules displayed in the GUI do not show the frequency statistics for the rules in rule-sets containing more than one speaker. The GUI only displays each rule with the frequency statistics for an individual speaker's rules as Shibboleth loops through the individual speakers. An example of rules in the Shibboleth GUI, including rules after a speaker's transcript was processed, are shown in Figure 15.



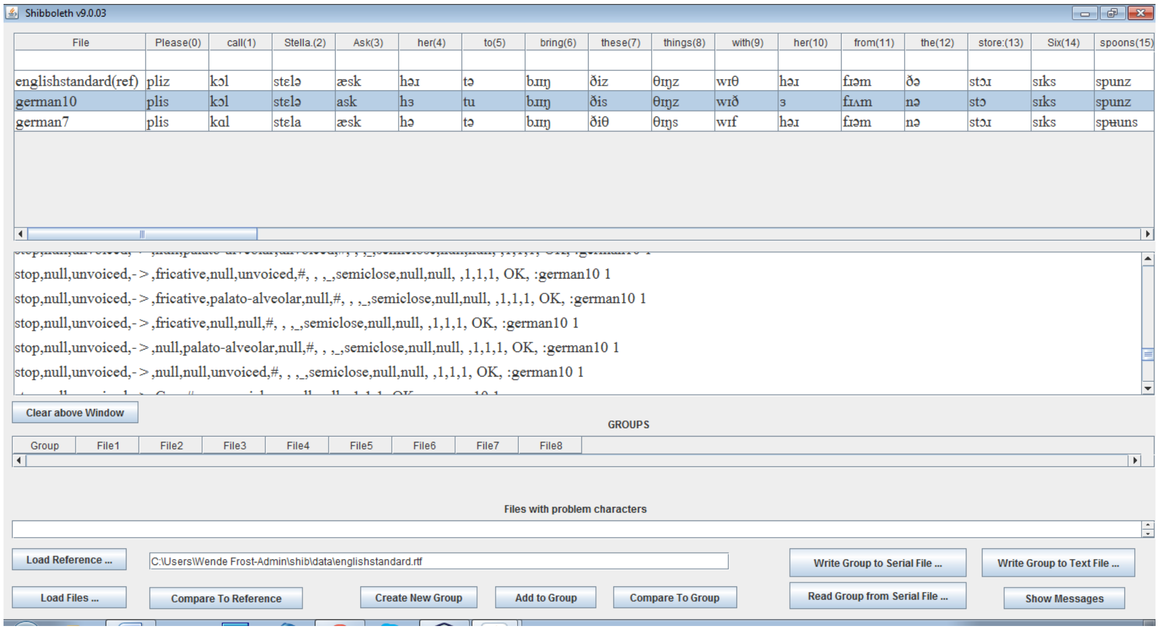
File	Please(0)	call(1)	Stella.(2)	Ask(3)	her(4)	to(5)	bring(6)	these(7)	things(8)	with(9)	her(10)	from(11)	the(12)	store:(13)	Six(14)	spoons(15)
englishstandard(ref)	pliz	kɔl	stɛlə	æsk	həɹ	tə	bɹɪŋ	ðiz	θɪŋz	wɪθ	həɹ	fɹəm	ðə	stɔɹ	sɪks	spunz
german10	plis	kɔl	stɛlə	ask	hɜ	tu	bɹɪŋ	ðis	θɪŋz	wɪð	ɜ	fɹʌm	nə	stɔ	sɪks	spunz
german7	plis	kɑl	stɛla	æsk	hə	tə	bɹɪŋ	ðiθ	θɪŋs	wɪf	həɹ	fɹəm	nə	stɔɹ	sɪks	spʉuns

```
æ --- a
s --- s
k --- k

h --- h
ə --- ɜ
```

C:\Users\Wende Frost-Admin\shib\data\englishstandard.rtf

Figure 14. Phone alignment in Shibboleth GUI output window.



File	Please(0)	call(1)	Stella.(2)	Ask(3)	her(4)	to(5)	bring(6)	these(7)	things(8)	with(9)	her(10)	from(11)	the(12)	store:(13)	Six(14)	spoons(15)
englishstandard(ref)	pliz	kɔl	stɛlə	æsk	həɹ	tə	bɹɪŋ	ðiz	θɪŋz	wɪθ	həɹ	fɹəm	ðə	stɔɹ	sɪks	spunz
german10	plis	kɔl	stɛlə	ask	hɜ	tu	bɹɪŋ	ðis	θɪŋz	wɪð	ɜ	fɹʌm	nə	stɔ	sɪks	spunz
german7	plis	kɑl	stɛla	æsk	hə	tə	bɹɪŋ	ðiθ	θɪŋs	wɪf	həɹ	fɹəm	nə	stɔɹ	sɪks	spʉuns

```
stop,null,unvoiced,->,fricative,null,unvoiced,#, , ,_,semiclose,null,null, ,1,1,1, OK, :german10 1
stop,null,unvoiced,->,fricative,palato-alveolar,null,#, , ,_,semiclose,null,null, ,1,1,1, OK, :german10 1
stop,null,unvoiced,->,fricative,null,null,#, , ,_,semiclose,null,null, ,1,1,1, OK, :german10 1
stop,null,unvoiced,->,null,palato-alveolar,null,#, , ,_,semiclose,null,null, ,1,1,1, OK, :german10 1
stop,null,unvoiced,->,null,null,unvoiced,#, , ,_,semiclose,null,null, ,1,1,1, OK, :german10 1
```

C:\Users\Wende Frost-Admin\shib\data\englishstandard.rtf

Figure 15. Feature sets shown in Shibboleth GUI output window.

The CSV file a user can create with the data from the Shibboleth program being run on two or more transcripts is the most complete collection of the output. This is the only place to see the frequency statistics for rules in rule-sets containing more than one speaker. It is easily sortable by feature value or frequency, and is already formatted for the speaker comparison module with minimal pre-processing. An example displaying rule-set columns for multiple Swedish speakers is shown in Appendix C.

3.2 Speaker Comparison Module.

The function of the speaker comparison module is to use descriptions of the accents of NNSs of a language or dialect, captured as rule-sets, to classify the languages or dialects of speakers whose L1 is unknown. Comparisons can be run at the dialect, language, or language family level, if the L1 exists as a robust rule-set. In the current version of the speaker comparison module, classification is run by training SVMs on rule-sets of a NNS speaking an L2 whose first language is already known. This will determine which rules seem most indicative of the characterization of the accent of that L1 in that L2. Unknown speakers can then be tested by the speaker comparison module to compare the rules the unknown speaker demonstrates to those from a variety of trained L1 rule-sets. The rule-set the speaker seems to match with the most closely is predicted to be the unknown speaker's L1.

3.2.1 Different types of rule-sets for classification. Even though the format of all rule-sets obtained from the rule-set creation module is identical, there are some underlying differences between the rule-sets for dialects, languages, and language families.

3.2.1.1 Language rule-sets. The rule-sets for monocentric languages are the least marked of the rule-sets: there is one rule-set for each L1 (currently created from one reference transcript); the reference transcript for the L2 was created from a standardized source, such as dictionary pronunciations; and the speakers within the rule-set are classified by self-description as being native speakers of the language, often with corresponding birthplace information. Examples of these types of rule-sets include Italian, Polish, and Dutch.

Rule-sets for pluricentric languages are very similar, but, as previously noted in 3.1.6.1, these types of languages will have one rule-set for each standard form. Currently, for the pluricentric language of French, there will never be a case where a speaker of an unknown L1 is classified as a "native French speaker." Rather, the result will be either "native Parisian French speaker" or "native Canadian French speaker." This format had two motivations. The first motivation for treating languages without a standard form as unique accents is that without a much larger number of training speakers somewhat equally distributed between the different standard forms of the language, the support-vector machines will rate the rules of the form with the greater number of speakers as being more indicative of the language. This is regardless of which variety those speakers spoke. The second motivation is that from a language classification point of view, each main form of a pluricentric language has an accent that is considered "standard," thus indicating a need for a different rule-set to describe the accent in that dialect standard. This differs from dialects in that, for monocentric languages, there will be one dialect that is considered the least marked and most standard for the language.

3.2.1.2 Language family rule-sets. Language family rule-sets contain rules for multiple languages within the same family or branch. Due to the amount of data the Speech Accent Archive contains for languages from the Indo-European family, the current experiments were done on three different branches of the Indo-European family: Germanic, Romance, and Slavic. A language family rule-set will contain a rule-set made up of the most frequently occurring rules of multiple standard languages within these branches. When a pluricentric language is used in a language family, all varieties of the language are used.

Language family rule-sets are very similar to language rule-sets, except they are currently limited in size due to computer memory limitations. On tests on computers with less than 2 GB of RAM, it was necessary to limit each language family rule-set to between 100 and 150 speakers total (a number far below the ideal) so that the rule-set creation module would not fail to finish running due to lack of memory. A single language rule-set, however, has not yet reached the limit of speakers it can contain by using the data in the Speech Accent Archive, as the highest number of speakers of any language other than English in the Archive is 100, for native Spanish speakers. However, even if the Speech Accent Archive contained more data, Shibboleth could generate rules for more speakers on the same hardware for a single language rule-set than it would be able to for a language family rule-set. This is because single language rule-sets have more instances of incremented rules rather than new rules. Incrementing takes less memory than creating and comparing all of the new rules generated by each language in the language family rule-set. Due to this limitation, running the rule-set creation module will eventually have to be done on more powerful hardware, with the results being returned to personal computers and mobile devices. Because of the current limitation on memory, the current number of speakers of an individual language within a language family consisting of *n* languages is no more than 100/n.

3.2.1.3 Dialect rule-sets. Dialect rule-sets are also similar to language rule-sets, but the speakers for these rule-sets are chosen in a slightly different way. Speakers for language rule-sets can be chosen based upon the individual's birthplace and self-reported L1. This provides rough, but seemingly accurate, information. However, it is much harder for an individual to self-report on his or her own major dialect area or for a researcher to assign a dialect area based on a scant amount of personal data and a short utterance. This is due to four main reasons:

1) The major dialect areas for a particular language or language variety may not have been studied and defined, or if they have been defined, these boundaries may be widely contested or not exact.
2) It is much more likely that an individual will move between geographical dialect areas over the course of his or her life than it is that they will move to another

geographical area with a different standard language, making birthplace information and self-reported speech much more accurate for languages than for dialects.

3) Without a detailed personal history of an individual who does move between dialect areas (something that the Speech Accent Archive does not give), along with a trained sociolinguist and abundant speech samples, it is difficult to assign the individual to a specific dialect for inclusion in a rule-set.
4) It is possible the individual has acquired characteristics of several dialects in his or her speech, to the extent that it is simply not possible to isolate the speaker into a single dialect.

Because of these issues, determining a way to choose speakers for dialect rule-sets in such a way that I could test them using Shibboleth was problematic. It was not possible to classify dialects using the basic personal information in the Speech Accent Archive. Instead, three native English-speaking individuals from three different dialect regions, listened to all American English samples that were going to be used in the American English dialect rule-sets and assigned them to one of five major American English dialect groups. While this method is certainly somewhat subjective and prone to error, in the absence of speakers who have lived their entire lives in the same geographic location with exposure to the same dialect, there should be a human assessment element when classifying speakers into dialects to create rule-sets for that dialect.

3.2.2 Training and testing data. Before the speaker comparison module can train SVMs on language, family, or dialect rule-sets or test unknown speakers against the trained SVMs, every speaker's transcript must be processed through the rule-set creation module. Speakers who are used for training SVMs are processed sequentially, going through the full process described in section 3.1.6.5, with the results being saved in the same CSV file as a training rule-set. Speakers who are being used to test Shibboleth's classification proficiency are processed individually, with Shibboleth stopping before the last loop of the rule-set creation module cycle (the step where the rules of multiple speakers are assembled together), and saved as a rule-set consisting of a single speaker.

For both types of rule-set, the last processing step between the rule-set creation module and the speaker comparison module is sorting the rules in each set by frequency, so that the higher frequency rules are at the beginning of the file and the lower frequency rules are at the end. This enables the Java library used in the speaker classification module, libsvm-3.16, to use only the higher frequency rules, since libsvm-3.16 pulls the data for analysis from the beginning of the file until the user specifies a stopping point (Chang & Lin, 2011). This is done easily in the spreadsheet by dividing the number of times a rule occurred in the rule-set by the number of times the input and environment appeared in the reference transcript, assigning the result as the frequency, and sorting the spreadsheet with the highest frequency rules appearing first and the lower frequency rules appearing later. Generally, the frequency will be a number between 0 and 1, inclusive, but in cases where an environment was created and this environment was the same as an existing environment in the reference transcript, the frequency could be higher than 1. This is due to the number of times the change actually occurred being higher than the number of times the environment and input occurred originally. Currently, the distribution of rule frequency for individual speakers in a group is not used in the calculation for frequency of a rule. In the next implementation of Shibboleth, rules that have a higher median value will factor more highly than rules with the same overall frequency but a lower median value. This will promote rules that have occurred across more speakers over rules that are unique to one or a few speakers, even if the mean frequency is the same. Sorting by frequency is done for all training and testing rule-sets. Currently, the user completes this step by sorting the CSV file in a spreadsheet application, such as Excel 2010, but it is an easy process that can be automated and included in the rule-set creation module. Once the rule-sets are sorted, the new CSV files can be input into the speaker comparison module.

3.2.3 Support vector machine algorithm. The basic SVM (Vapnik & Lerner, 1963) takes as input a set of data as vectors, and separates this data into two classes by finding the line which has the largest distance to the nearest vector of either class while separating the classes. This separation is done by a maximum-margin hyperplane (shown in Appendix D).

The separation can be visualized as a line separating two sets of points, where the line is as far from all of the points simultaneously as possible, as shown in Figure 16.

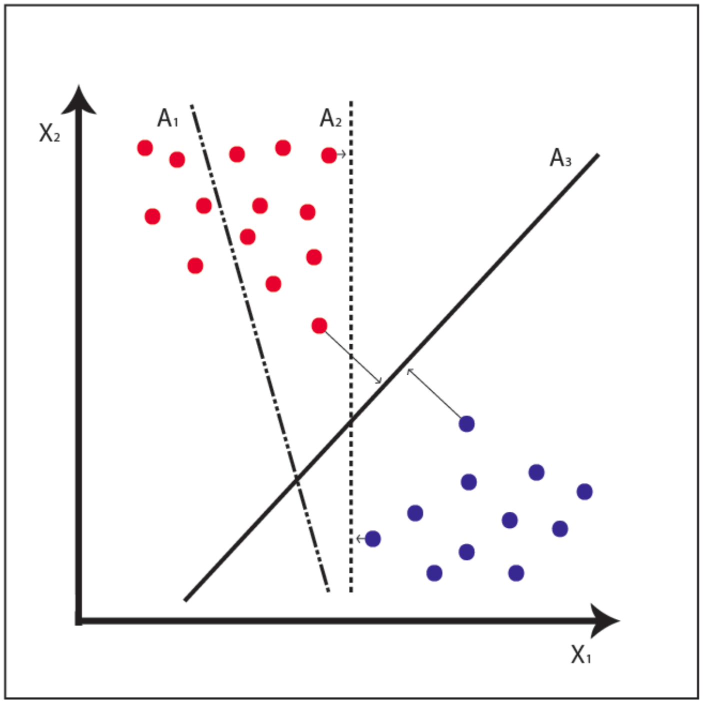


Figure 16. Maximum-margin hyperplane. Line A_1 does not correctly separate the data, line A_2 separates the data, but not with the maximum margin, and line A_3 separates the data with the maximum margin of separation between the classes. Line A_3 is the maximum-margin hyperplane.

An expansion of the original algorithm for maximum-margin hyperplanes is to use kernels instead of a dot product equation to create a nonlinear classifier from the original algorithm (Boser, Guyon, & Vapnik, 1992). The equation for the nonlinear SVM classifier is shown in Appendix D. This allows the feature space to be transformed into n dimensions, enabling a hyperplane to linearly separate data in the new space while remaining non-linear in the original input space. Shibboleth uses a Gaussian radial basis kernel (shown in Appendix D).

An example of using three dimensional vectors to create the hyperplane between two classes is shown in Figure 17. The number of dimensions that is used to separate n-dimensional data is n-1.

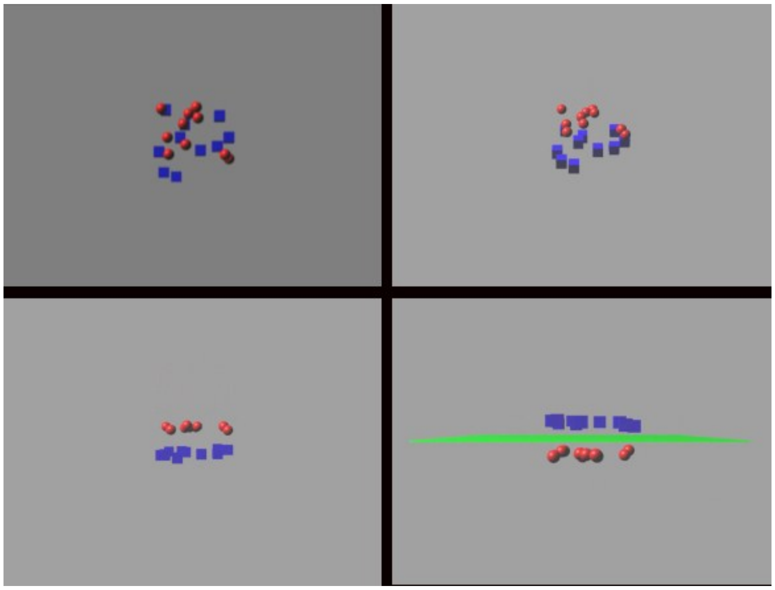


Figure 17. Demonstration of hyperplane separation in three dimensions. Top left: Vectors that cannot be separated by a plane in two-dimensional space. Top right: The same vectors rotated 45°. Bottom left: The same vectors now rotated 90° from top left, showing that a separation is possible in three-dimensional space. Bottom right: The vectors separated by the maximum-margin hyperplane in three-dimensions.

The SVM is by its very nature designed for two-class problems. For multi-class problems, an approach must be chosen to allow vectors to be classified into more than two classes. Two common strategies are the winner-takes-all approach and the max-wins approach. Currently, Shibboleth uses the Java library libsvm-3.16 to classify the rule-sets from unknown speakers' transcripts with the hyperplanes created by the support vectors in the trained model. The winner-takes-all approach is the strategy implemented by libsvm-3.16. This strategy finds the best overall set of hyperplanes to separate all of the pairs of classes at the same time.

Another strategy is the max-wins strategy, where each classifier model assigns a vector to one of two classes in the two-class pairwise comparison, and increments a "winning" vote each time a class is assigned. The class with the most winning votes at the end of classification determines to which class the vector is ultimately assigned. This is the strategy used for unknown speaker classification in Shibboleth. The reasons for this choice that diverges from libsvm-3.16 will be expanded upon in section 3.2.4.3.

The rationale for using SVM classification comes from the unique requirements of the task. While the rule-set creation module builds rule-sets containing thousands of rules to use in classification, there are simultaneously a dearth of speakers to use for training data, and a lack of repetition of many phonemes within the reference sample in the Speech Accent Archive. Any algorithm that requires a large (hundreds or thousands) group of speakers to use as training data is not feasible (Reynolds, 2008). Neural network algorithms give multiple solutions for local minima, which is not a desired attribute, since classifying each speaker of unknown origin to a single L1 is the ideal solution in Shibboleth. Neural networks also require more data than the Speech Accent Archive contains. SVMs provide the ability to use all of the rules created for classification, while still retaining a measure of robustness given the small number of speakers in the training data (Meyer, Leisch, & Hornik, 2003).

3.2.4 Support vector machine classification procedure. Between the rule-set creation module and the speaker comparison module, there are a few parameters for the user to input.

Before the module is run, it needs to be told how many of the 12 features, as shown in Figure 4, the SVM should use in its analysis, whether the rule-set being input is a training or a test rule-set, which rule-sets are being compared (in the case of training), whether only rules of certain frequencies should be included in the analysis, and whether only a certain number of rules should be compared. The ability to input these preferences will eventually be added to the Shibboleth GUI. Currently, the user changes them in the source code of the file before each run. All experiments in this thesis use all 12 features. The rule-sets being compared and the number of rules included in the analysis change according to what experiment is being run. The frequency of the rules is currently set to include frequencies between 0 and 2.01 in the analysis. (Without the inclusion of 2.01, Shibboleth would not include rules that had frequencies of 2.)

For the rules to be in the correct format to operate as vectors in the SVM, they are translated from rules containing features such as “alveolar” and “voiced” to scaled twelve-dimensional vectors in an automated pre-processing procedure. The first step of this procedure is assigning twelve columns that contain feature information (i.e. the three columns which contain features of the input phone, the three columns which contain features of the output phone, the three columns that contain features of the pre-environment, and the three columns that contain features of the post-environment) to the dimensions 1 through 12, as shown in Figure 4. Next, the module converts all features into numerical values from 0 to 20 according to Figure 18, which it then assigns to their corresponding numbered dimension, as shown in Figure 19. In the example shown in Figure 19, for example, 1:19 would be the input value for the distinctive feature “stop.” The values are sorted by general similarity of consonant and vowel features and place of articulation. (Another practical option would be to sort the values by place of articulation without regard for consonant and vowel differences.) The combination of dimension labels and feature values comprises one “vector.” For a rule-set that is going to be used to train the SVM, each vector also contains an initial numerical label to specify which rule-set the vector is from (e.g. the initial identifier for a vector from the German training rule-set may be “1” and the identifier for Italian may be “2”). These numbers are created arbitrarily, but must remain consistent across models. This file of rules represented as vectors can now be processed by the libsvm-3.16 Java library.

Numerical Value	Feature 1	Feature 2	Feature 3
0	NULL	null	null
1	null	-	-
2	-	back	unrounded
3	#	near-back	rounded
4	C	central	unvoiced
5	V	near-front	voiced
6	close	front	
7	semi-close	bilabial	
8	close-mid	labio-palatal	
9	mid	labiovelar	
10	open-mid	labiodental	
11	semi-open	dental	
12	open	alveolar	
13	approximant	alveolo-palatal	
14	trill	palato-alveolar	
15	nasal	retroflex	
16	fricative	palatal	
17	affricate	velar	
18	tap	uvular	
19	stop	glottal	
20		pharyngeal	

Figure 18. Numeric assignment chart for feature values in the support vectors.

```
1 1:19 2:0 3:5 4:0 5:1 6:1 7:13 8:12 9:5 10:3 11:1 12:1
```

Figure 19. Example vector for the rule 0->(stop, null, voiced)/(approximant, alveolar, voiced)_#.

Using a method, svm-scale, within the libsvm-3.16 library, the user inputs the preference to scale the data. Scaling the data will convert the values assigned to the dimensions into a proportional value between 0 and 1. Scaled data tends to give more accurate results, though the SVM can be run on the vectors without scaling them. All data used in experiments for this thesis have been scaled.

Finally, if the SVM is being used to train a model, the user inputs into the svm-train method in the libsvm-3.16 library the preferences for the cost value, the kernel type used, and any parameters specific to that kernel. For these experiments, I use a radial basis function for the kernel, with a cost value of 32, and a gamma value of 0.5. These are average values for these parameters.

3.2.4.1 Output of SVM Training. Once the user has run the SVM to train two or more rule-sets, the speaker classification module outputs a model file (viewable as text) containing all of the vectors that created the best separation between the rule-sets. Since the vectors on the margins of this hyperplane, as seen in Figure 20, border the separation between the rule-sets drawn by the hyperplane, they are, in this case, also the vectors most likely to belong to multiple rule-sets and to be weak indicators of unique rules for the accents in question.

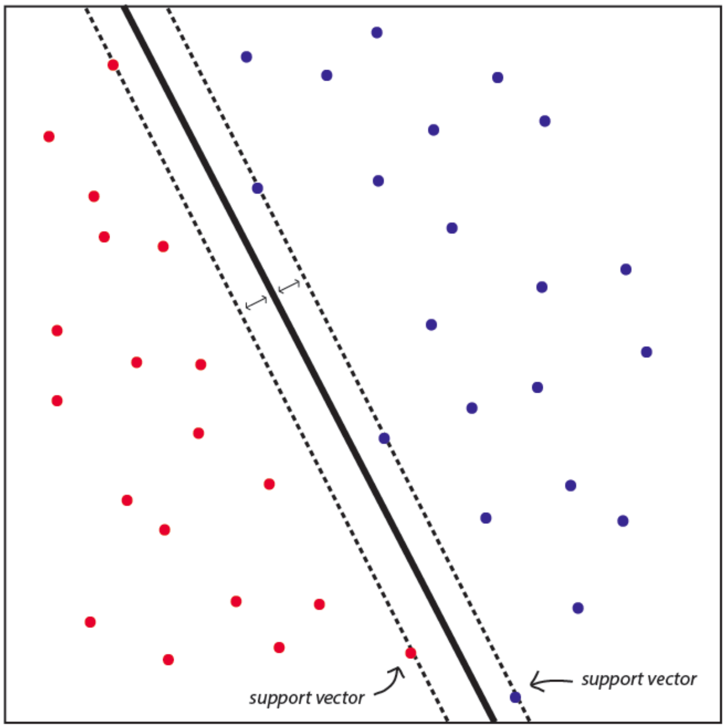


Figure 20. Support vectors lying on the margins of the hyperplane.

Future work will look at two different methods of altering the libsvm-3.16 output to give higher importance to rules that seem more strongly indicative of the language in question. The first alteration to the current method will discard these “border” rules in the models from the rule-sets which they were trained on, and the system will retrain on the new rule-sets to get a new model, with, presumably, greater separation between the rule-sets in question. This is shown in Figure 21.

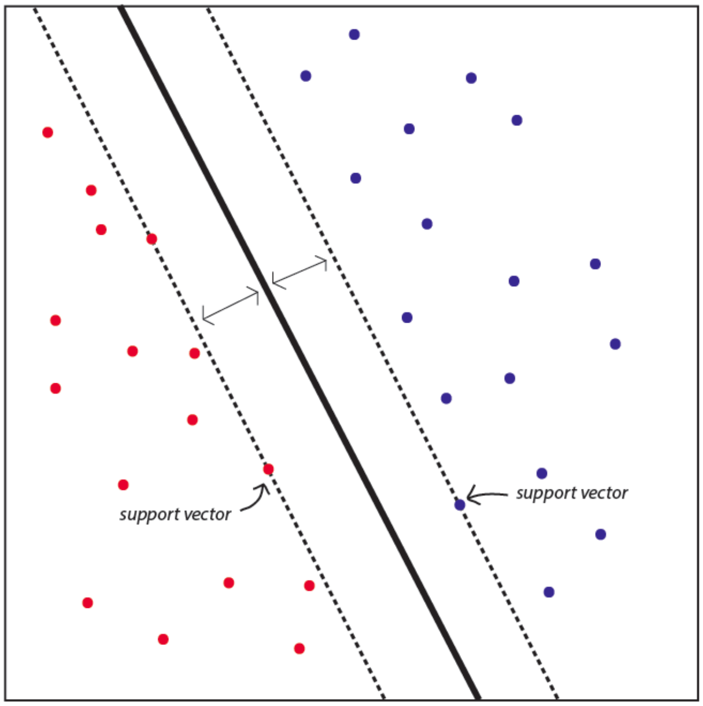


Figure 21. New margins to the hyperplane once the original support vectors were discarded.

This retraining is the easiest alteration to implement. A more difficult alteration is to use a distance function to give rules higher weights the further they are from the rules that are used as support vectors in the trained models. This would not result in the border rules being discarded entirely, but it would give each rule or vector a higher value the more distant they are from the hyperplane. Those values would then be more indicative of a particular L2, rather than what occurs in the current system, where all rules are weighted equally. A visualization of this technique is shown in Figure 22.

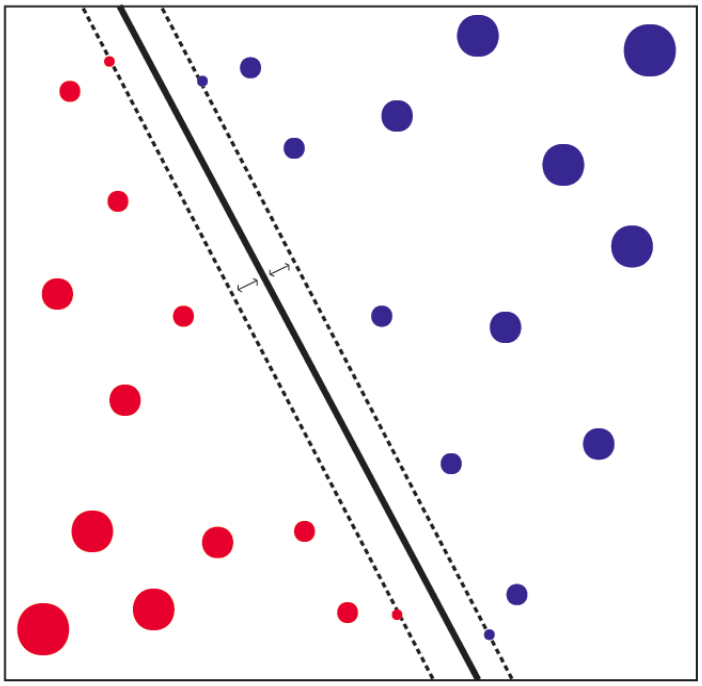


Figure 22. Vectors on either side of a hyperplane with sizes (weights) corresponding to the distance from the hyperplane.

The current model file created by libsvm-3.16 containing all of the support vectors for the combination of rule-sets from training can be saved and used indefinitely, until the user wishes to change the training parameters or add or subtract rules to the rule-set. This is useful from a practical perspective, as testing unknown speakers to see which rule-set they most likely fall into is generally a matter of a few minutes of computation, but many models can take hours, days, or weeks to train. The training time increases exponentially for every rule-set added or for roughly every 20,000 rules added to an individual rule-set.

3.2.4.2 Output of SVM testing. The procedure for pre-processing a rule-set from a testing transcript to be compared against the model files is the same as the procedure for pre-processing rule-sets to be used in creating the model files, with one difference. For data that will be used to test classification, the vectors created for the input based on the rules in the rule-set are no longer labeled with the initial numerical rule-set identifier, since the numerical identifier will not be known for speakers of unknown origin.

After the rule-set has been converted into a file usable by libsvm-3.16 and scaled accordingly, the user can run the method svm-predict on the rule-set file and whichever previously-trained model to which he or she wishes to compare the rule-set file. The output from svm-predict is a list of the numerical rule-set classifications of every vector within the testing rule-set according to where the SVM classifies them using the boundaries created by the support vectors in the model file. The classification of an unknown speaker is based upon where a majority of its rules fall in the 12-dimensional space within and across any pertinent model files and the rule-sets within them. Each run of svm-predict is considered a "test." One test, depending upon the type of test, may not be enough to classify a test rule-set correctly against the available rule-sets. This does not necessarily mean that the test result is incorrect, but it may mean that unless it is part of a series of tests, the results are meaningless. This occurs frequently in the pairwise comparisons discussed later.

Currently, the output of the svm-predict method is the output of the speaker comparison module, and, hence, the Shibboleth program. However, the text file that is output with the classification of each vector for a specific test is not easily readable and does not quickly feed in to further testing. Future implementations of Shibboleth will give the user the ability to see the classification of a user within each model file at a glance and enable the user to compare a speaker's classification across model files.

3.2.4.3 Pairwise and k-way classification. SVMs naturally create a separation between two classes of objects, or a pairwise separation. When libsvm-3.16 is separating more than two classes of objects, it begins by doing the pairwise separation for all classes and finding the vectors that create the hyperplanes to delineate the classes. After this initial separation, it does parameter comparison to find the best hyperplanes for the overall k-way comparison, even if that hyperplane is not the best for a specific pairwise separation. When the likely best hyperplanes have been located, all pairwise comparisons are automatically run again. If the accuracy goes down significantly, libsvm-3.16 corrects to slightly different parameters to eventually find the vectors that create the hyperplanes that maximize the overall percentage of objects separated correctly. This means that while both a manual pairwise separation done by the user and the automated k-way separation done by libsvm-3.16 both begin with the same pairwise separation, the k-way separation proceeds through several iterations to find the best hyperplanes, leading to anywhere from a two- to eight-fold increase in time spent training the SVM.

In a hypothetical world where training model files with as many rule-sets as desired and testing against this model took no perceptible amount of time, it would be ideal to create a complete model consisting of rule-sets of all of the world's languages as L1s. Any time a new rule-set was created or an old one was updated, a new, complete model would be created immediately. However, in reality, the memory required and the time it would take makes this method highly infeasible for the average personal computer.

Instead, decomposing the process of training and testing to the underlying pairwise, or two-way, comparisons provides trained models and language classifications in a more tractable amount of time. At this time, each model and test is run or created individually by the user, but future updates to Shibboleth will train a complete series of pairwise comparison models automatically, as well as enable the user to test an unknown speaker against every model in a specified set without further input. Decomposing every k-way comparison into a series of pairwise comparisons creates k(k-1)/2 model files and between (k-1) and k(k-1)/2 tests for every speaker of unknown origin. However, it takes less time to train the set of models than to create a single k-way comparison model for any k greater than two, because of the previously mentioned iterations. It takes an equal to slightly greater amount of time to run a single test. In this case, "slightly greater" refers to a difference of seconds to minutes, whereas the time saved on training the models has been shown to be on the order of several hours to days in experiments conducted on training up to seven rule-sets at a time. In addition, it is faster to incorporate changes in a rule-set or add new rule-sets, because pairwise models only need to be created or re-created to compare each individual rule-set to the new rule-set. In the previously mentioned experiment, it was so unwieldy to create a model of a seven-way comparison between rule-sets (over a week), that adding an eighth rule-set to the comparison model, at the cost of re-running the entire model, would likely take multiple weeks on the average home computer. This assumes the computer doesn't run out of memory before the model is completed. In contrast, adding an eighth rule-set to complete a set of pairwise models, would involve creating seven more (k-1) pairwise models that would run at roughly 30 minutes to two hours apiece. Similarly, if one of the existing rule-sets needed to be expanded or otherwise altered, only the pairwise models that included that rule-set, (k-1) models, would be re-created. If the user did not wish to include a rule-set in a comparison, effectively "deleting" the rule-set for the time, no new model would have to be created. The user would simply run the classification without including models that contain the unwanted rule-set. In a model that does a k-way comparison between rule-sets in one model, a new model would have to be created for deletions as well as additions or alterations.

The drawback of solely creating pairwise models is that the extra refinements libsvm-3.16 runs to create demarcations between the k rule-sets (where k is larger than two) after the initial internal pairwise comparison is lost. This means that while the training process may be many times faster and more tractable, it may not be quite as accurate. This is shown in Chapter 5, Language Family Classification Experiment. Currently, it also requires much more input on the part of the user, since each pairwise model is created separately, rather than in an automated process. In the same way, each time a speaker of unknown origin is tested against a set of models, the user must still run each test individually and keep track of the results.

For example, when classifying a speaker of unknown origin against a three-way comparison model of English, German, and Italian as L1s, if 50% of the vectors are classified as Italian, 25% as German, and 25% as English, the origin of the speaker is presumed to be Italian. (As mentioned previously, the vectors all currently have a weight of one, without regard for the distance of the vector from the boundary between classes.)

Italian as an L1 in this case is ascertained by training the SVM on one model, running one test, and having the result output. However, if the same speaker was classified against pairwise comparison models, three models would be trained and two or three tests, depending on the results, would be run. The models created would be files of the support vectors between English and German as L1s, German and Italian, and English and Italian. If any two of these were used for classification and the output was the same, i.e. over 50% of the vectors were classified as Italian after the latter two models were used in classification, the speaker would be classified as Italian without needing to use the third model for classification. Not only does the user have to create the three models individually, he or she must also run the two tests and take note of the results. Until this part of the process is automated, more user error can occur here than in the k-way single model creation and language testing process.

3.2.4.4 Classification using pairwise models. For the experiments in this work, pairwise comparisons were used unless noted otherwise, for the reasons stated previously. This required a standardized method of running the tests and interpreting the results.

The first step to testing a rule-set against a series of pairwise models in an efficient, accurate way is to choose a classification (e.g. Italian, Greek, Germanic language family, Southern American English) and test the rule-set against every pairwise comparison that includes this classification. It doesn't matter which classification is initially chosen. This will be (k-1) tests, or pairwise tests against (k-1) models. The result of this comparison will be the number of vectors that matches against each classification in the model. The training rule-set that matches against the larger number of vectors (or rules) in the testing rule-set determines the classification of that single test. The exact percentages of match are not considered in classification. In the unlikely event that the testing rule-set matches the same number of vectors in both training rule-sets, the outcome of the test is considered to be both rule-sets, for the unknown speaker falls equally into both training rule-sets. The outcome of all (k-1) tests determines whether or not the rule-set needs to be tested against more models. If the test rule-set matches the initial classification chosen in all (k-1) tests, no further tests need to be run, because no other classification can achieve a majority classification over (k-1). (For example, if the initial classification chosen is Italian and the test rule-set was classified as Italian in every pairwise model that included Italian as an option, no further tests need to be run on that rule-set.)

If most, but not all, of the (k-1) tests come out to be the initial classification, the next set of pairwise models to compare the test rule-set against would be all of those containing the classification with which the test rule-set has the second-highest match. To continue with a variation on the previous example, if the test rule-set was classified as Italian in six out of nine tests, German in one out of the nine tests, and Swedish in one out of the nine tests, the next set of tests against the pairwise models could be tests against all of the (k-2) models that contain German as a possible classification, or against those that contain Swedish as a possible classification. For this example, I will choose to run the next set of tests on German. If the test rule-set does not have the majority of the rules classified as German in at least six out of the eight tests, the test rule-set is still classified as Italian, and the seven models remaining that contain Swedish should be run next. If the test rule-set is classified as German in seven out of the eight tests, German is considered the classification of the test rule-set and no further tests should be run. If the test rule-set is classified as German in exactly six out of the eight tests for that particular classification, the new classification ties with Italian for the highest likelihood of it being the speaker's L1 (adding the six new classifications as German from the second set of tests and the single German classification from the first set of tests is equal to the seven classifications as Italian from the first set of tests). Barring one of the tests from the second set being classified as Swedish, no further tests need to be run.

In the event that during the second round of tests the test that compared German and Swedish classified the unknown speaker as Swedish, the third set of comparisons must be run on every model that contains Swedish, unless the other seven tests during the second round classified the speaker as German, because there would still be a possibility of eight or more of the nine possible comparisons including Swedish to classify as Swedish.

The tests should proceed through every pairwise model of the next highest classification until it can be ascertained either that one rule-set holds the majority classifications with the test rule-set, or there is a tie between classifying the unknown speaker into two or more rule-sets. In the case of equal classification into three or more rule-sets, the accent of the speaker is termed "inconclusive."

3.2.4.5 Output, certainty, and vector issues. Using the previously discussed classification method, every vector in an individual unknown speaker's transcript will be classified as a member of one of the classes in a model. No vector is ever classified as belonging to two or more classes. This will lead to output for a speaker in a percentage format, where x% of the vectors will belong to class 1 in a pairwise model and 100-x%

will belong to class 2. While it may be tempting to assign higher certainty to testing output where a speaker has a much higher percentage belonging to one class, this is not always accurate. It is reasonable to assume that a speaker who scores very strongly (over 60% in multiple tests) in the same language repeatedly across pairwise tests that the language is the speaker's L1 with higher certainty than a speaker who scores only slightly over 50% in a language in the same tests. However, that is just a rule of thumb. In reality, the composition of the data that the models for that language were trained on can skew the results, leading to no overall measure of certainty. At this time, a speaker is classified as either a likely L1 speaker of one language, an L1 speaker of a language differing from the L2 but not discernible between two strong candidates, or an L1 speaker differing from the L2 but of inconclusive L1, with no additional certainty measures.

While it is obvious that an unknown speaker may vary in degree of accentedness affecting classification, overall classification is affected more by the composition of the data used to train the model for the language, dialect, or language family. The principal culprits of poor or weak classification due to the model fall into four categories: 1) there are not many speakers contributing to the training rule-set, 2) the speakers contributing to the training rule-set vary in accent strength, 3) the L2 in question does not differ significantly in phonemic inventory from the L1 reference sample, and 4) the vectors in the L2 rule-set being trained have significant overlap with the vectors of another L2 rule-set.

The first issue is the easiest issue to solve, in theory. More speakers in the training rule-set lead to more rules and more incrementation of existing rules, thus, a higher percentage of high probability rules and a higher chance of locating unique rules for classification.

However, the first issue is strongly related to the second issue. Namely, will a higher quantity of speakers actually improve the training model in and of itself, or does the degree of accentedness of speakers used in the training model affect the accuracy of classification more than the number of speakers? It may be that those speakers that display a higher degree of accentedness in the L2 lead to a more robust model for that L1. Conversely, it may be that speakers with a low degree of accentedness lead to rules that are more representative for a larger number of unknown speakers of that L1, and thus better overall classification. If an L1 is only classified by the most marked speakers in an L2, it may be that the training model will not pick up speakers with lesser degrees of accentedness. Since measures of accentedness were not part of this work, the underlying assumption here is that the largest number of speakers usable for a training model would average out the more accented speakers with the less accented speakers, giving a general model of the accent from a particular L1.

The third issue, that of similar existing phonemic inventories, is something that cannot be solved in the current version of the program, or even by using the suggested future modifications. It is likely that when the L2 has a very similar phonemic inventory to the L1, classification will be less accurate than when the inventories hold different phonemes. However, it is not an inherently unsolvable problem. Using the same theoretical underpinnings in Shibboleth, a similar program could create feature sets that are then used to classify unknown speakers against trained models using SVMs. However, the feature sets will no longer be made up of, or at least not solely made up of, broad articulatory features. They would have to contain either more specific articulatory features, such as the diacritics that Shibboleth removes, acoustic features, or both.

The last issue, overlapping vectors between training models, is an issue that is not solved in the current implementation of Shibboleth, but is one that would likely be somewhat ameliorated by the previously mentioned future modifications to classification. Removing the support vectors themselves would almost certainly remove some of the rules that overlap between models and L1s, leaving a higher percentage of vectors that fall squarely into one class or the other. The other suggested modification, calculating distance from the support vector, would also give higher weights to vectors that have a higher likelihood of only belonging to one class and lower weights to vectors that are more likely identical across classes.

CHAPTER 4

LANGUAGE CLASSIFICATION EXPERIMENT

The Shibboleth program was evaluated by means of two experiments: a language classification experiment and a language family classification experiment. The language classification experiment sought to measure whether Shibboleth satisfies the main goal of classifying speakers of several unknown L1s speaking a second language into the correct L1 classes. This classification should be done at a rate better than chance to be considered successful. Previous research has shown that many factors can affect the strength of accentedness in second language speakers to various extents (Flege et al., 1995; Flege et al., 1997; Piske et al., 2001; Flege et al., 1999; Long, 1990). Shibboleth should perform better on speakers with a greater degree of accentedness, by the obvious corollary that speakers with no foreign accent will be indistinguishable from native speakers. Some factors, however, I consider more likely to affect Shibboleth's performance. The factors considered most likely to affect the accuracy of the results are age of onset and years spent living in a primarily English-speaking country. Some factors, such as age and method of instruction, are considered less likely to affect the accuracy of the results. Other important factors on which the Speech Accent Archive does not provide data include frequency of L2 use and self-reported degree of accentedness.

4.1 Participants

The pool of participants for this experiment is all of those from the Speech Accent Archive for whom transcripts were uploaded at the time of the experiment. Within the Archive, there are a vast number of languages represented, but many of the languages contain fewer than 10 speakers. Several have just one or two speakers represented. While the limitations of Shibboleth have yet to be exhaustively determined, having at least 10 speakers in a training set and four to be used in the unknown speaker test set (14 total transcripts) was the absolute minimum for a language to be used in this experiment. Ten speakers was the minimum number of languages I hypothesized Shibboleth would need to create a rule-set for the characterization of a foreign accent or dialect in a language. In the case of pluricentric languages, at least one standard dialect needed to have at least 14 speakers' transcripts available in the archive for the language to be represented in the experiment. This limitation reduced the number of languages from the Archive that were able to be used in this experiment to 18, including the native English speakers. Of these, the six languages I chose to use for L1s were Northern American English, Standard German, Italian, Japanese, Latin American Spanish, and Russian.

Each one of the six groups was divided by a randomizer spreadsheet function into 75% training data and 25% testing data. There were 125 total speakers' transcripts used in this experiment, with 101 of the speakers' transcripts being placed in a training pool and 27 transcripts placed in the testing pool. To ensure an equal number of participants from each L1 were tested, if a language had more than four transcripts in the testing pool, the additional transcripts were removed, leading to 24 transcripts in the testing pool. The transcripts removed were determined by the same randomizer function used to assign transcripts to either the training or the testing pool. The characteristics of each training group are shown in Table 1. The information about years spent in an English-speaking country gathered from the Speech Accent Archive was rounded to the nearest year, or up to one year if the individual had lived in an English-speaking country for one to five months. The individuals self-reported their status as a native speaker. For pluricentric languages, the speakers were placed in a dialect group by what they identified as their hometown or region. Once an individual's transcript was selected to be used in testing Shibboleth, they were placed in the testing group as an L1 speaker of an unknown L1. The characteristics of this group of unknown speakers related to their age of English onset, number of years speaking English, and number of years living in a primarily English-speaking country are shown in the last row of Table 1. Age and sex information for both groups is shown in Table 2.

Table 1

English Background Data for Speakers Used in Language Classification Experiment

	Age of English onset			Number of years spoken			Years in English-speaking country		
Training L1	*M*	*Mdn*	*s*	*M*	*Mdn*	*s*	*M*	*Mdn*	*s*
English	0	0	0	39	39	17	39	39	17
German	11	11	1	19	14	13	4	1	8
Italian	13	14	5	21	17	10	7	4	10
Japanese	12	12	1	17	13	9	3	2	3
Russian	18	15	12	19	16	12	8	6	7
Spanish	15	15	5	15	13	13	5	2	6
Testing L1									
Unknown	10	11	6	19	18	10	8	4	11

Note. The specific dialects of each L1 used are described in the text above.

Table 2

Age and Sex Information for Speakers Used in Language Classification Experiment

	Age					Sex	
L1	*Min*	*Max*	*M*	*Mdn*	*s*	F	M
English	18	67	39	39	17	4	15
German	19	53	30	23	14	6	4
Italian	18	55	35	32	13	3	13
Japanese	20	49	29	26	9	6	4
Russian	23	66	37	34	13	8	11
Spanish	17	80	29	26	14	10	17
Unknown	18	46	29	29	9	13	11

The smallest training groups were German and Japanese, each with 10 speakers. The largest was Latin American Spanish, with 27 speakers. The native English speaking and Russian groups each had 19 members and the Italian training group had 16. Not all of the speakers had lived in an English-speaking country at the time of data collection, but they had all studied or spoken English for at least a year, with the mean number of years past age of onset for the training groups ranging from 15 to 21 for those speaking English as an L2, and 39 for native English speakers.

Fifty of the speakers were female and 75 were male. The youngest speaker was 17 years old and the oldest was 80, both of which were in the training group of Latin American Spanish speakers. The lowest mean age was 29 for both the Spanish and Japanese training groups and highest for the native Northern American English speakers, at 39. Through the randomization, the group of speakers to be tested ended up with a higher percentage of female speakers than most of the training groups, and a relatively young overall age, matching the mean age of the youngest training group with a similar sample standard deviation and younger minimum and maximum ages. These slight variations in demographics from the training groups were not expected to affect the results of the experiment, as age and sex were not expected to have a noticeable correlation with the results of Shibboleth testing.

All participants in this experiment had transcripts collected in the Speech Accent Archive. The Speech Accent Archive has collected data from 1998 to the present by varying methods, including unmonitored recording of participants reading a passage into a tape recorder, participants being approached by friends and classmates to be recorded using tape recorders, smartphones, and other recording devices, and participants online sending in speech recordings from their home, work, and school computers. Though the methods varied, all participants read the same elicitation paragraph written in English, shown in Figure 5, in a relatively quiet background, with the file eventually being converted into the QuickTime (MOV) file type. Each sample averaged 27 seconds long. Using these methods, the Archive has many participants who self-selected for their willingness to have their speech recorded, had access to a computer with an internet connection and audio-recording capability, and/or attended GMU. This experiment assumes that the speakers are reading the written passage using the natural accent they would use in this setting.

The transcribers of the data were graduate students at GMU who were in an advanced phonology class with at least one previous semester of phonological transcription. Each transcription was done by two students and Dr. Weinberger, with disagreements handled by group discussion until a transcription could be agreed upon. Each student did three to four transcriptions over a semester, and, in the latter years of the project, used programs such as Praat (Boersma & Weenink, 2013) for assistance.

4.2 Procedure

Evaluation of Shibboleth in this experiment involved training 15 sets of pairwise models using the transcripts of the speakers assigned to the training groups. These models were trained using the common parameters of cost at 32 and gamma at 0.5. They were each trained on the top 50,000 frequency rules in their rule-sets. The unknown speakers were tested against the pairwise rule-sets using the method mentioned in section 3.2.4.4. For the unknown speakers, all rules in their rule-sets were used for classification, regardless of the frequency of the rule. Data for both the training rule-sets and the testing rule-sets were scaled. For the purposes of this experiment, an unknown speaker could only be correctly or incorrectly identified. If a speaker's classification was tied between an incorrect L1 and the correct L1, it was considered incorrect.

4.3 Results

Shibboleth proved successful at identifying the L1 of 42%, or 10 out of 24 of the unknown speakers, when the system had been trained on six different L1s. Identifying speakers by chance would have had a 17% success rate, or four out of the 24 unknown speakers. This result is significant, $\chi^2(1, N=24) = 10.800, p = .0010$. The assignments for each unknown speaker are listed in Figure 23, with the full results of the pairwise tests shown in Appendix E. The results are graphed by L1 in Figure 24.

Unknown Speaker Number	Correct classification	Actual classification
1	German	Inconclusive
2	German	Italian
3	German	German
4	German	German
5	English	English
6	English	English
7	English	English
8	English	English
9	Italian	Italian
10	Italian	Italian
11	Italian	Russian
12	Italian	Italian
13	Japanese	Inconclusive
14	Japanese	Inconclusive
15	Japanese	Italian
16	Japanese	Inconclusive
17	Spanish	Inconclusive
18	Spanish	Inconclusive
19	Spanish	Inconclusive
20	Spanish	Inconclusive
21	Russian	Russian
22	Russian	Inconclusive
23	Russian	Italian
24	Russian	Italian

Figure 23. Classification results for unknown speakers in language classification experiment.

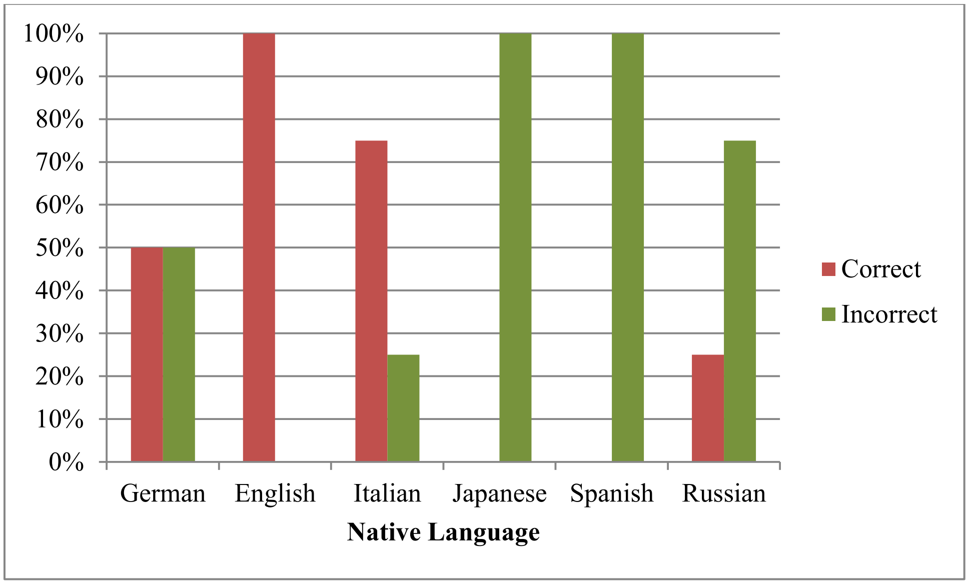


Figure 24. Results for language classification experiment graphed by L1.

It is evident from these results that some factor of the combination of L1 and L2 led to more successful classification for some L1s than others. For example, even though the training rule-set for the Latin American Spanish group was created with more than twice as many speakers as the training rule-set for the German group, the unknown speakers whose first language was German were classified correctly 50% of the time, while not one of the Latin American Spanish speakers was identified. Shibboleth was able to distinguish native and non-native English speakers with 100% accuracy.

When the native English test speakers are removed from the calculations, it becomes clear that every non-native test speaker who was correctly classified had an age of onset later than 11 years, as hypothesized. Contrary to what I hypothesized, greater time spent living in an English-speaking country did not have an effect on classification results. A slightly high percentage of males were identified correctly, but not a high enough percentage to be significant. There was no correlation between number of years English was spoken by the participant and correct classification. Correct and incorrect classification for the unknown speakers by sex, age of onset, number of years that English was spoken, and years spent in a primarily English-speaking country are shown in Figures 25, 26, 27, and 28, respectively.

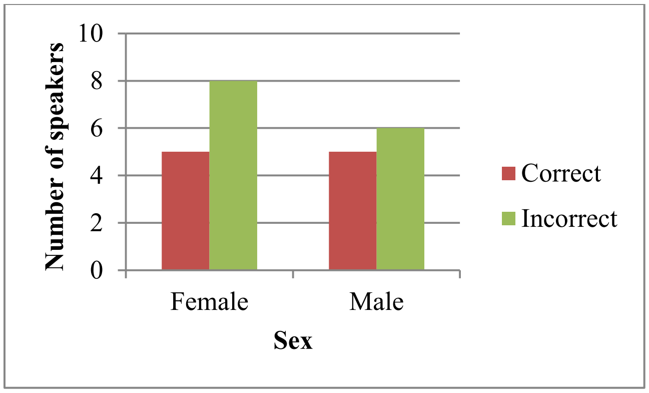


Figure 25. Results of classification in language classification experiment graphed according to sex.

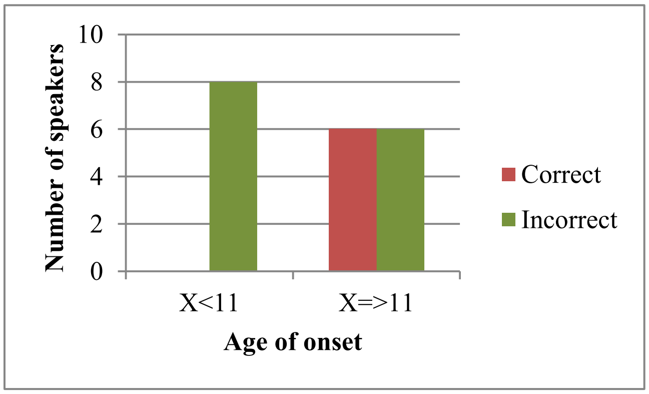


Figure 26. Results of classification in language classification experiment graphed according to age of onset.

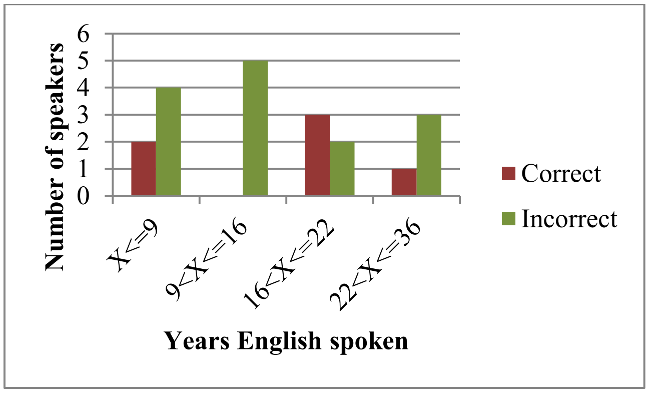


Figure 27. Results of classification in language classification experiment graphed according to number of years English was spoken.

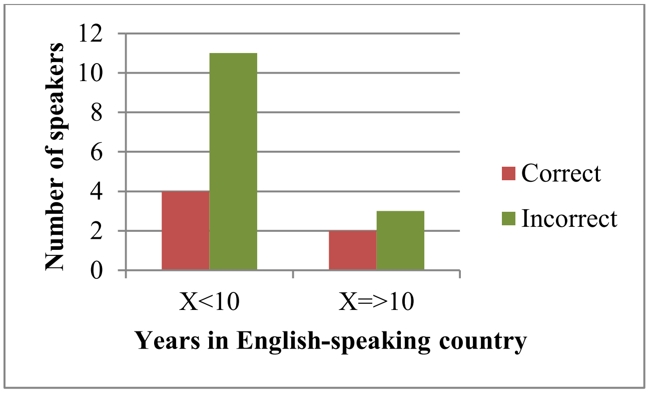


Figure 28. Results of classification in language classification experiment graphed according to years spent in an English-speaking country.

In addition to demographic comparisons, I gathered statistics on the training rule-sets to compare size and weight of the training rule-sets with correct classification of the test speakers. No correlation between number of rules in the transcript used for testing or weight of rules in the training rule-set was found, as shown in Table 3.

Table 3

Statistics from Training and Testing Language Rule-Sets

	German	English	Italian	Japanese	Spanish	Russian
Average weight of top 50K rules	.52	.35	.48	.45	.51	.54
Average number of rules in testing transcript	75,790	35,806	91,009	72,749	80,341	90,206
Number of speakers in training data	10	19	16	10	27	19
Accuracy of testing	50%	100%	75%	0%	0%	25%

The lack of correlation between higher average numbers of rules in the testing transcript and accuracy of testing or higher average weight of the rules in the training data and accuracy of testing may stem from a similar confounding factor. When the accent of an L1 in an L2 does not contain many unique phonetic changes, but, instead, contains many subtle phonetic changes that are common to many L1s, this L1 is very difficult to uniquely identify. Some evidence that this was a problem shows in the precision and recall figures calculated from the individual pairwise tests conducted, as shown in Table 4.

Table 4

Precision and Recall for Pairwise Classifications

L1	Precision	Recall	F1-score	Overall Accuracy
German	.667	.800	.727	.500
English	.741	1.000	.851	1.000
Italian	.594	.950	.731	.750
Japanese	.667	.300	.414	.000
Spanish	.333	.050	.087	.000
Russian	.625	.750	.681	.250

While the F1-score and recall measures trend with the overall accuracy percentages, the trend is not replicated in the precision measurement. Japanese has one of the highest precision values, but one of the lowest accuracy values, and Italian has one of the highest accuracy values, but one of the lowest precision values. This is due to Italian having the highest number of false positives reported and Japanese having almost no false positives found from the pairwise comparisons. A large number of false positives in the pairwise comparison tests are an indicator that an L1 has a large number of rules that are not unique to that L1. This will lead to high recall for that language, but low precision, since the support vector machine will erroneously classify too many native speakers of other L1s into the class.

CHAPTER 5

LANGUAGE FAMILY CLASSIFICATION EXPERIMENT

The language family classification experiment measured Shibboleth's ability to classify speakers of unknown L1 into the correct language family class, even if the L1 is not represented in the training data for the language family. This classification should be done at a rate better than chance for it to be considered successful. As with the language family experiment, I hypothesized that age of onset and years living in a primarily English-speaking country would be factors most likely to affect correct classification. I also hypothesized that languages that had more near relatives in the training data would also positively affect classification.

5.1 Participants

The pool of participants for this experiment is all of the individuals that had a transcript uploaded to the Speech Accent Archive at the time the experiment was conducted. Similar to the language classification experiment, I did not include a language in the training data unless it had at least 10 transcripts in the Archive. I did not include a language family in the training data unless at least three languages from that family had at least 10 transcripts in the Archive. In addition, the language family needed to have at least two languages in the Archive distinct from the three languages used for training transcripts to comprise the testing pool, for a total of five languages. Only three major language families in the Archive met these criteria: Germanic, Romance, and Slavic. The languages used for training the language family classification models were English, German, and Dutch; Italian, French, and Spanish; and Serbian, Polish, and

Russian, respectively. The languages used for the testing pool were Afrikaans, Swedish, Romanian, Portuguese, Bulgarian, and Ukrainian.

No speakers who had previously been used as unknown speakers in the testing pool for the language classification experiment were used in the training data. As with the language classification experiment, 75% of the transcripts for each training language were chosen using a spreadsheet randomizer function. In addition, the transcripts chosen for the training pool consisted of every standard dialect in the pluricentric languages. Unlike the language classification experiment, the testing pool was not 25% of each language chosen for testing. Since some of the languages chosen for testing had such a low number of speakers that 25% would have been an insufficient number for multiple tests, 3 speakers for each one of these languages were chosen by the randomizer function and placed in the testing pool.

This resulted in 193 transcripts being used in this experiment, with 175 speakers' transcripts in the training pool and 18 speakers' transcripts in the testing pool. The characteristics of each training group are shown in Table 5. The information about years spent in an English-speaking country gathered from the Speech Accent Archive was rounded to the nearest year, or up to 1 year if the individual had lived in an English-speaking country for one to five months. The individuals self-reported their status as a native speaker. The characteristics of the group of speakers whose native language family is unknown are shown in the last row of Table 5. Age and sex information for the training and testing groups is shown in Table 6.

Table 5

English Background Data for Speakers Used in Language Family Classification Experiment

	Age of English onset			Number of years spoken			Years in English-speaking country		
Training L1 Family	*M*	*Mdn*	*S*	*M*	*Mdn*	*s*	*M*	*Mdn*	*s*
Germanic	5	0	6	32	27	17	22	19	22
Germanic w/o English	11	11	2	25	23	14	5	1	7
Romance	13	12	7	21	17	15	8	3	11
Slavic	16	14	10	17	15	11	5	2	6
Testing L1									
Unknown	10	10	4	24	21	13	7	5	9

Table 6

Age and Sex Information for Speakers Used in Language Family Classification Experiment

	Age					Sex	
L1 Family	*Min*	*Max*	*M*	*Mdn*	*S*	F	M
Germanic	18	73	37	33	16	24	37
Romance	17	80	35	31	17	25	43
Slavic	20	66	33	30	11	22	24
Unknown	18	56	32	29	12	6	12

The smallest training group was Slavic, with 46 speakers. Germanic and Romance were very close in number, at 61 and 68 speakers, respectively. Not all of the speakers had lived in an English-speaking country at the time of data collection, but they had all studied or spoken English for at least a year, with the mean number of years past age of onset for the training groups ranging from 17 for Slavic to 32 for Germanic. (Without the inclusion of native English speakers in Germanic, the mean number of years past age of onset was 25.)

Seventy-seven of the speakers were female and 116 were male. The youngest speaker was 17 and the oldest was 80, both of which were in the training group for the Romance language family. All of the training groups and the testing group had similar mean ages, from 29 for the testing group to 33 for the Germanic language family training group. All groups had fewer female speakers than male speakers.

All participants in this experiment also had transcripts collected in the Speech Accent Archive. The collection and transcription techniques used for these transcripts were the same as those described in section 4.1.

5.2 Procedure

Evaluation of Shibboleth in this experiment involved two different techniques: training 3 sets of pairwise models and training one three-way model. The former is the Shibboleth training procedure described in section 3.2.4.4. The second technique is the “ideal” scenario mentioned in section 3.2.4.3. In this instance, the number of training groups was low enough that it was feasible to use a three-way model to demonstrate the slight difference in outcome between this model and the normal pairwise models. All models were trained using a cost parameter of 32 and a gamma value of 0.5. The models were trained on the top 50,000 frequency rules in their rule-sets. For the unknown speakers, all rules in their rule-sets were used for classification. Data for the training and testing rule-sets was scaled. Similar to the language classification experiment, an unknown speaker could only be correct or incorrectly identified. Using the three pairwise models, it was not possible to get a tie between two model classifications.

5.3 Results

Using the typical pairwise model method, Shibboleth proved successful at identifying the language family of 33%, or six out of the 18 unknown speakers, when the system had been trained on three different language families. None of the language families contained the actual language of the unknown speaker. This is the same result that identifying speakers by chance would give. Using the alternate three-way comparison, Shibboleth proved successful at identifying the language family of 44%, or eight out of the 18 unknown speakers. Though this is an improvement, it is still not statistically significant $\chi^2(1, N=18) = 1.000$, $p=.3173$. The assignments for each unknown speaker using both methods are shown in Figure 29, with the full results of the pairwise tests shown in Appendix F. The results are graphed by L1 language family in Figure 30.

Unknown Speaker Number	L1	Correct classification	Classification by pairwise model	Classification by three-way model
1	Swedish	Germanic	Inconclusive	Romance
2	Swedish	Germanic	Germanic	Germanic
3	Swedish	Germanic	Germanic	Germanic
4	Afrikaans	Germanic	Inconclusive	Germanic
5	Afrikaans	Germanic	Inconclusive	Germanic
6	Afrikaans	Germanic	Inconclusive	Romance
7	Romanian	Romance	Romance	Romance
8	Romanian	Romance	Romance	Romance
9	Romanian	Romance	Romance	Romance
10	Portuguese	Romance	Romance	Romance
11	Portuguese	Romance	Inconclusive	Germanic
12	Portuguese	Romance	Inconclusive	Germanic
13	Bulgarian	Slavic	Inconclusive	Romance
14	Bulgarian	Slavic	Inconclusive	Romance
15	Bulgarian	Slavic	Inconclusive	Romance
16	Ukrainian	Slavic	Inconclusive	Romance
17	Ukrainian	Slavic	Inconclusive	Romance
18	Ukrainian	Slavic	Inconclusive	Germanic

Figure 29. Classification results for unknown speakers in language family classification experiment.

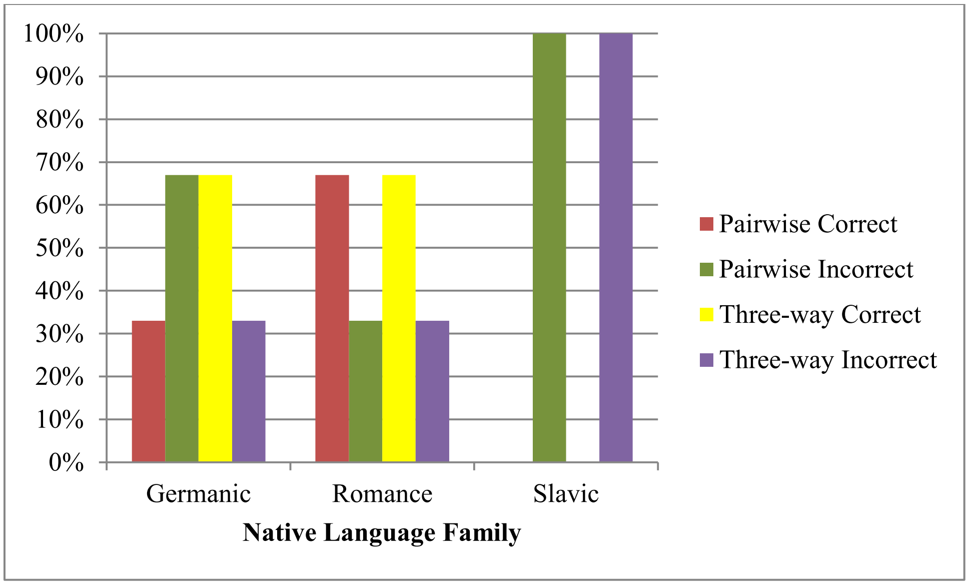


Figure 30. Results for language family classification experiment graphed by L1 language family.

The slight improvement in classification accuracy between pairwise and three-way classification shows in the improved accuracy of the unknown speakers from the Germanic language family. In fact, as evidenced in Appendix F, the two previously misidentified Afrikaans speakers went from failing both pairwise tests to being correctly classified by the three-way classification.

From this, it is clear that Slavic is consistently misidentified, or identified as inconclusive, similar to Spanish and Japanese in the earlier experiment. This was an unexpected result, as it was hypothesized the Russian speakers' transcripts in the training rule-set would aid in identifying the unknown speakers with an L1 of Ukrainian. Conversely, the most easily identified speakers in the testing pool by both methods were native Romanian speakers, the language with arguably the least close relatives in any training group. While Spanish and Portuguese were both in the Iberian branch of the Romance family, Romanian had no such close relative. These results show that having languages from branches of a language family in the training group more closely related to the L1s of unknown speakers is not an indicator of more likely correct language family classification. However, as shown in Figure 31, the majority of the speakers that were classified into the correct language family had an age of onset later than 11 years, as hypothesized and as shown in the language classification experiment.

Correct and incorrect classification for the unknown speakers by age of onset, sex, number of years that English was spoken and years spent in a primarily English-speaking country are shown in Figures 31, 32, 33, and 34, respectively.

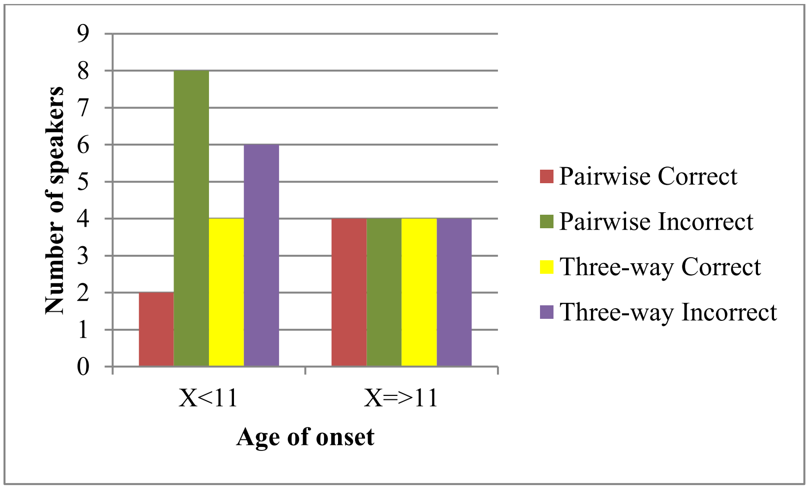


Figure 31. Results of classification in language family classification experiment graphed according to age of onset.

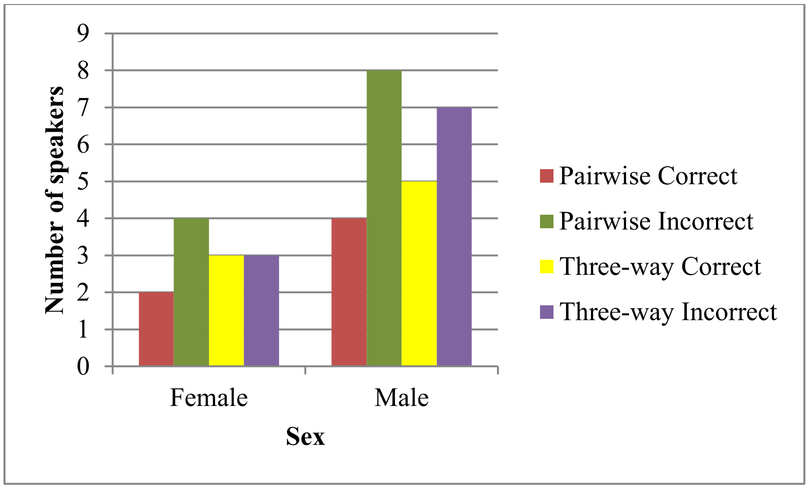


Figure 32. Results of classification in language family classification experiment graphed according to sex.

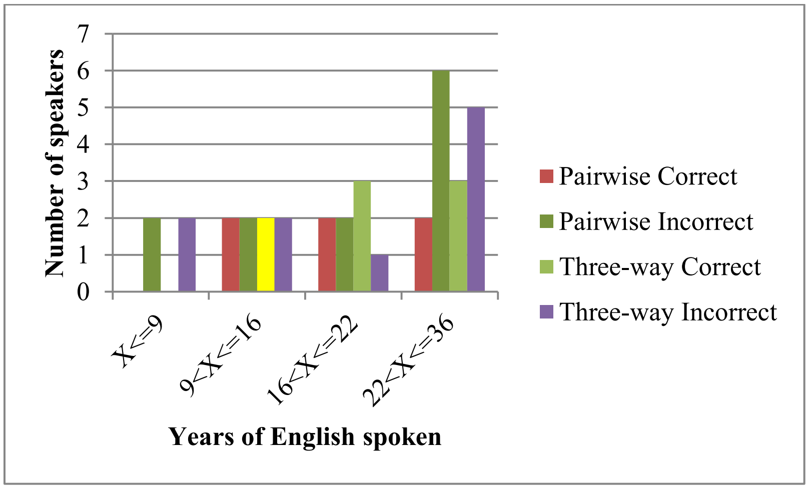


Figure 33. Results of classification in language family classification experiment graphed according to years English was spoken.

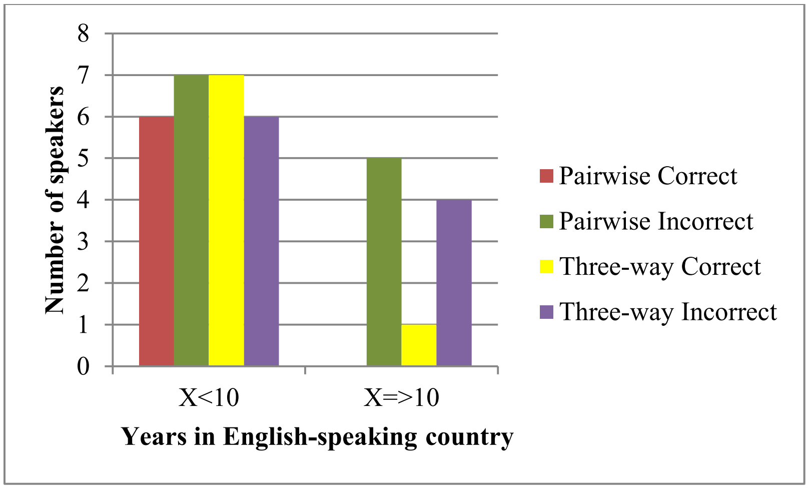


Figure 34. Results of classification in language family classification experiment graphed according to years spent in an English-speaking country.

The statistics that compare the size and weight of the training rule-sets are shown in Table 7. As with the language classification experiment, no correlation was found between the number of rules in the transcript used for testing or weight of rules in the training rule-set.

Table 7

Statistics from Training and Testing Language Family Rule-Sets

	Germanic	Romance	Slavic
Average weight of top 50K rules	.36	.46	.53
Average number of rules in testing transcript	68,898	101,595	89,525
Number of speakers in training data	61	68	46
Accuracy of pairwise testing	33%	67%	0%
Accuracy of three-way testing	67%	67%	0%

In these statistics, a trend can be seen between a higher number of speakers in the training data and higher accuracy of classification. Since there were only three language families represented in this experiment, this apparent trend may also be coincidence, given the difference from the language classification experiment. The language family classification experiment likely also suffered from the issue of having rule-sets with too many common phonetic changes that occurred in the language classification experiment. The precision and recall figures shown in Table 8 support this theory.

Table 8

Precision and Recall for Pairwise and Three-way Classification

Language Family – Pairwise	Precision	Recall	F1-score	Overall Accuracy
Germanic	.571	.333	.421	.333
Romance	.500	.833	.625	.667
Slavic	.556	.417	.476	.000
Language Family – Three-way				
Germanic	.571	.667	.615	.667
Romance	.363	.667	.470	.667
Slavic	.000	.000	.000	.000

The F1-scores and recall measure do not trend with the overall accuracy percentages for the pairwise comparisons, but there is some trending with the overall accuracy for the three-way comparison. Overall, the relatively low precision scores across both the pairwise and the three-way comparisons show the constant misclassification in the language family classification experiment. Slavic was not correctly or incorrectly identified once in the three-way experiment, showing that Slavic is likely a language family that contains many phonetic rules common to other language families. When the three-way model had to create the separation between those rules belonging to other language families, most of the vectors were placed in the classes for the other language families. This led to the 0% accuracy rating in both the pairwise and three-way comparison, even though it is much more clearly reflected in the precision and recall scores for the three-way comparison. Conversely, the Romance family had low precision scores in both comparisons, not due to lack of true positive identification, but because of an overabundance of false positive identification. This latter factor regarding the preponderance of false positives in the Romance data provides compelling evidence that many of the support vectors for Romance were vectors that appeared in multiple language families.

CHAPTER 6

DISCUSSION, CONCLUSIONS, AND FUTURE WORK

6.1 Summary

In this thesis I have described a computer program, Shibboleth, which will automatically identify the native language, dialect, or language family of a speaker from the foreign accent they exhibit in a second language. This description was followed by two experiments I conducted to evaluate the ability of the system to classify speakers accurately. One experiment sought to classify speakers of an unknown L1 into his or her L1 when Shibboleth had been previously trained on rule-sets in the L1. Six L1s were used for this experiment. The second experiment sought to classify speakers of an unknown L1 into his or her native language family, when Shibboleth had been previously trained on rule-sets in the language family, excluding the speaker's specific L1. Nine L1s across three language families were used in the training data for this experiment. For both experiments, I hypothesized that Shibboleth would be able to classify the unknown speakers at a rate better than chance, with the two factors most heavily influencing correct classification being age of onset of the L2 (English), and the number of years the speaker had spent in an English-speaking country.

The results of the first experiment, language classification, demonstrated a 42% correct classification rate for 24 speakers of unknown L1, when Shibboleth had been trained on 101 total language samples at roughly 27 seconds apiece (45 minutes of training data). This result is significantly better than chance, $\chi^2(1, N=24) = 10.800$, p=.0010, and comparable or better than related work in the field (Angkititrakul & Hansen, 2006; Choueiter, Zweig, & Nguyen, 2008). However, classification was uneven across L1s, with 100% correct classification of native speakers of English, and 0% correct classification for Spanish and Japanese speakers. The factor that most heavily influenced correct classification for this experiment did appear to be age of onset, as I hypothesized. However, the number of years a speaker had spent in an English-speaking country had no clear effect on classification.

The second experiment, language family classification, was not as successful, resulting in 33% correct classification for pairwise comparisons and a 44% correct classification rate for a three-way comparison. Neither result is significantly better than chance $\chi^2(1, N=18) = 1.000$, $p=.3173$. For this experiment, Shibboleth was trained using 175 language samples in nine languages across three language families. Each sample was around 27 seconds, for 79 minutes of training data. With the results of the second experiment approximately equal to chance, neither predictive factor could be shown to have a distinct effect on classification.

6.2 Discussion and Challenges

From these experiments, it appears that Shibboleth shows promise for language classification. While the classification rate is not yet high enough to rely on Shibboleth as a source of certain L1 classification from L2 speakers, it is comparable in correct comparison rate to similar software, with the larger pool of L1s causing a greater positive deviation from chance. In addition, the amount of data per speaker in the training group is less than the amount of data required by related tools.

The experiment most closely related work to the Shibboleth work presented here was that done by Angkititrakul and Hansen (2006). In this work, a hidden Markov model classification system using phone-based acoustic models aligned to text classified unknown speakers into one of four L1s at an accuracy level of 47% at the word level, using 107 minutes of training data from a pool of 64 speakers. Similar to Shibboleth, this system performed best when identifying the accent of native American English speakers speaking English. The ability of accent classification systems to correctly classify American English (or whatever the L2 of the experiment happens to be) stems from the lack of rules arising in the interlanguage for native speakers. Adult native American English speakers speaking American English do not have an interlanguage with rules that reflects interference or developmental processes from another language, so there are many fewer rules describing the accent of a native speaker, and these rules are much more uniform than those of L2 speakers.

Another high accuracy system was a 23-way automatic accent classification task conducted by Choueiter, Zweig, and Nguyen (2008), who obtained a 32% accuracy rate using a combination of heteroscedastic linear discriminant analysis and maximum mutual information. However, not only did this task use many times more data than Shibboleth (a corpus consisting of 27.25 hours of data), purely acoustic data was used. No phonetic rules were found using this method.

While Shibboleth proved successful in the language classification experiment for the scope of this experiment, it did not succeed in language family classification. This could stem from several issues. It is possible that the underlying hypothesis that a language family can be determined for a speaker of unknown origin, even if the system has no data on the specific L1, is simply not true. Data on human experiments or other systems that perform this particular task is not readily available. Another option is that specific language families can be identified from related languages, but that this is not an ability that applies to all language families or branches. The last option is that this undertaking is possible, but is currently not being completed successfully by Shibboleth.

Assuming this latter option is true, the difficulty with this type of classification can have many causes. First among them is lack of data. It is possible that to locate interlanguage phonological characteristics for an entire language family comprising dozens of languages requires data from many more languages within the family, and possibly more data per language, than I used to train Shibboleth for the language family classification experiment. Another option is that language family rule-sets have more non-unique rules than individual language rule-sets. This would require a technique giving greater separation between the classes to be used, such as the technique discussed in section 3.2.4.1, deleting the initial set of support vectors and widening the margin from the hyperplanes. A third option is that more features, such as length, stress, or frequency needs to be used in computing the feature vectors. To accomplish this, Shibboleth would require more detailed transcriptions or data directly from acoustic input.

A hybrid solution would be to use Shibboleth to classify languages into language families when there are already rules from the L1 in the language family rule-set. It is possible that with rules from other languages in the language family, fewer speakers for a specific L1 would need to be used. This solution would draw upon Shibboleth's existing capability at classifying speakers into their native language, as well as adding flexibility in the number of speakers for the L1 required in the training rule-set.

A challenge that arose in both the language classification experiment and the language family classification experiment was the preponderance of rules common to multiple classes. When a large number of rules were common to multiple classes, Shibboleth had difficulty classifying these and other similar phonetic rules into a specific class. Since a rule can belong to the interlanguages of native speakers of multiple L1s but can only be classified into one L1 by Shibboleth, it is likely that there is no way to create accurate hyperplanes between these rules. Not only would this lead to common misclassification of these rules, it would lead to misclassification of similar rules. This problem is obvious in the language family pairwise experiment, when every language family had precision scores influenced by several incorrect positive classifications from other language families, as well as in the three-way language family experiment, where the precision scores fell even further. This indicates that when a three-way separation is forced, the clustering of the common rules largely within one class hurts the overall precision.

For the language classification experiment, the precision scores were, on average, higher, but there were still issues with classification due to overlapping rules. For example, the precision of the classification of Russian speakers was average for this experiment and the recall value was .75, or 15 out of 20 of the pairwise tests, but the accuracy was only 25%. Appendix E shows that in the individual Russian pairwise tests, it was an overriding classification from Italian that kept 50% of the Russian speakers from being correctly classified. In fact, of the speakers that were classified incorrectly as speakers of another L1 (rather than being classified as inconclusive), four out of five were incorrectly classified as Italian. This is a strong indication that the Italian rule-set shares many rules with other rule-sets, and that many of these common rules have been classified by Shibboleth as Italian. It is likely that if the rules creating the margin of the hyperplane were removed for this experiment, each language would be left with more unique rules. While this does not necessarily guarantee improved accuracy in classification of unknown speakers, it is likely that the misclassification of speakers across languages as having an L1 of Italian would go down.

6.3 Contribution and Future Work

Shibboleth proved successful in its main aim of identifying speakers of unknown L1s at a rate better than chance. This capability can be refined for higher accuracy and expanded across speakers of L1s not yet tested. In addition, Shibboleth successfully creates rule-sets of possible phonetic rules in the interlanguage of speakers from any L1 speaking English. This rule-set can be used to explore the interlanguages of L1 speakers by observing the frequency of expected interference and revealing unexpected high frequency rules. (Unexpected low frequency rules are likely individual speakers' disfluencies and unlikely to lead to major insights.) High frequency rules that do not stem from transfer can be examined for understanding common developmental processes in learning a specific L2 from an L1.

Future expansions to Shibboleth fall into one of three categories: improving accuracy, improving usability, and improving the subject pool. As discussed, removing the rules creating the support vectors from the margins should create greater separation between the classes. This rule deletion may have to be iterated multiple times, if there are many rules clustered very close to the margin. This should lead to an improvement in accuracy in language classification and perhaps in language family classification.

Since Shibboleth currently operates over transcripts of speaker utterances, the next major improvement will be the ability to operate over a transcript automatically generated from acoustic data. This will speed data collection enormously. Automatically generated transcripts will not be as accurate, but many of the smaller inaccuracies should be balanced out by the larger amount of speaker data this will provide.

This leads to the third improvement, collection of utterances from more NNS of English. These speakers will come from both the testable L1s in the Speech Accent Archive (Weinberger, 2013) as well as L1s and dialects that are currently not testable in the Archive due to lack of data. In addition, more informal speech containing different phonetic combinations will be obtained.

With these improvements, I expect to obtain greater accent classification accuracy for speakers of unknown languages and language families as well as dialect classification capabilities.

REFERENCES

Angkititrakul, P. & Hansen, J. H. L. (2006). Advances in phone-based modeling for automatic accent classification. *IEEE Trans. On Audio, Speech and Language Proc.,* 14, 634-646.

Bloomfield, L. (1984). *Language.* Chicago: University of Chicago Press.

Boersma, P. & Weenink, D. (2013). Praat: doing phonetics by computer [Computer program]. Version 5.3.57, retrieved 27 October 2013 from http://www.praat.org/

Boser, S., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. *In 5th Annual ACM Workshop on COLT*, 144-152. Pittsburgh, PA: ACM Press.

Brockhaus, W. (1995). *Final devoicing in the phonology of German* (Vol. 336 of Linguistische Arbeiten). Tübingen: Max Niemeyer.

Campbell, W. M. (2008). Language recognition with discriminative keyword selection. *ICASSP 2008*, *Las Vegas, NV.*

Campbell, W. M., Campbell, J., Reynolds, D. A., Singer, E., & Torres-Carrasquillo, P. A. (2006). Support vector machines for speaker and language recognition. *Computer Speech and Language*, 20(2-3), 210–229.

Chang, C.-C. & Lin, C.-J. (2011) LIBSVM: a library for support vector machines. *ACM Trans. On Intelligent Systems and Technology*, 2(3).

Choueiter, G., Zweig, G., & Nguyen, P. (2008). An empirical study of automatic accent classification, *ICASSP 2008*, *Las Vegas, NV.*

Chomsky, N. & Halle, M. (1968). *The Sound Pattern of English*. Cambridge: MIT.

Clyne, M. G. (Ed.). (1992). *Pluricentric languages: Differing norms in different nations.* Berlin; New York: Mouton de Gruyter.

Cortes, C. & Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, 20(3), 273-297.

David, H.A. (1988). The Method of Paired Comparisons. New York: Oxford University Press.

Dictionary.com Unabridged. Retrieved June 24, 2013, from Dictionary.com website: http://dictionary.reference.com/browse/dictionary

Eckman, F. (1981). On the naturalness of interlanguage phonological rules. *Language Learning*, 31(1): 195-216.

Flege, J. E. (1980). Phonetic approximation in second language acquisition. *Language Learning*, 30, 117-134.

Flege, J. E., Frieda, E. M., & Nozawa, T. (1997). Amount of native-language (L1) use effects the pronunciation of an L2. *Journal of Phonetics,* 25(2), 169-186.

Flege, J. E., Munro, M. J., & MacKay, I. R. A. (1995). Factors affecting strength of perceived foreign accent in a second language. *The Journal of the Acoustical Society of America,* 97(5), 3125-3134.

Flege, J. E., Yeni-Komshian, G. H., & Liu, S. (1999). Age constraints on second language acquisition. *Journal of Memory and Language,* 41(1), 78-104.

Hecht, B. F., & Mulford, R. (1987). The acquisition of a second language phonology: Interaction of transfer and developmental factors. In Ioup, G. & Weinberger, S. (Eds.), *Interlanguage phonology: The acquisition of a second language sound system.* (pp. 213-228). Cambridge, MA: Newbury House.

Java - Methods. Retrieved November 8, 2013, from tutorialspoint.com website: http://www.tutorialspoint.com/java/java_methods.htm

Kondrak, G. (2000). A new algorithm for the alignment of phonetic sequences. *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, 288-295. Retrieved from http://webdocs.cs.ualberta.ca/~kondrak/papers/naacl00.pdf

Kunath, S. A., & Weinberger, S. H. (2009). STAT: Speech transcription analysis tool. *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies 2009: Demonstrations*, 9-12. Retrieved from http://www.aclweb.org/anthology-new/N/N09/N09-5003.pdf

Labov, W., Ash, S., & Boberg, C. (2006). *The atlas of North American English: Phonetics, phonology and sound change: A multimedia reference tool.* Berlin: Mouton de Gruyter.

Long, M. (1990). Maturational constraints on language development. *Studies in Second Language Acquisition,* 12, 251–285.

Love, N. & Ansaldo, U. (2010). The native speaker and the mother tongue. *Language Sciences*, 32(6), 589-593.

Macken, M. A., & Barton, D. (1977). A longitudinal study of the acquisition of the voicing contrast in American English word initial stops, as measured by VOT. *Papers and Reports on Child Language Development*, 14, 74-121.

Macken, M. A. & Ferguson, C. A. (1981). Phonological universals in language acquisition. In Ioup, G. & Weinberger, S. (Eds.), *Interlanguage phonology: The acquisition of a second language sound system.* (pp. 3-22). Cambridge, MA: Newbury House.

Major, R. C. (2008). *Foreign Accent: The Ontogeny and Phylogeny of Second Language Phonology.* Mahwah, NJ: L. Erlbaum.

Major, R. C. (2013). Foreign accent. In C. A. Chapelle, J. Levis, & M. Munro (Eds.), *The encyclopedia of applied linguistics.* New York: Wiley-Blackwell.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing,* 55(1-2), 169-186.

Piske, T., MacKay, I. R. A., & Flege, J.E. (2001). Factors affecting degree of foreign accent in an L2: A review. *Journal of Phonetics*, 29(2), 191-215.

Reynolds, D. (2008). Gaussian mixture models. *Encyclopedia of Biometric Recognition.* Springer.

Richards, J.C. & Schmidt R. (2002). *Longman Dictionary of Language Teaching and Applied Linguistics*. (5th ed.). London: Longman.

Shen, W., Chen, N. & Reynolds, D. A. (2008). Dialect recognition using adapted phonetic models. *Interspeech 2008*, *Brisbane, Australia.*

Sherrod, P. H. (2013). *Support-vector machines*. Retrieved from http://www.dtreg.com/svm.htm

SIL International. (2011, August 28). Doulos SIL Font Package [Font Package]. Retrieved from http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=doulossilfont

Theodoridis, S. & Koutroumbas, K. (2009). *Pattern recognition* (4^{th} Ed.). Academic Press.

Vapnik, V. & Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automated and Remote Control,* 24, 774-780.

Weinberger, S. (2013). *Speech Accent Archive*. George Mason University. Retrieved from http://accent.gmu.edu

Weinreich, U. (1968). *Languages in Contact. Findings and Problems.* The Hague: Mouton de Gruyter.

APPENDIX A

KEY TERMS

age of onset – the age at which a non-native speaker begins instruction in a second language

cost – a parameter in support vector machines that controls the trade-off between allowing training errors and forcing rigid margins (Sherrod, 2013).

dot product equation – an algebraic equation that takes two equal-length sequences of numbers and returns a single number.

feature vector – see **vector**

gamma – a parameter in support vector machines that defines how far the influence of a single training example reaches, with lower values being farther away (Sherrod, 2013).

gigabyte (GB) - 1024^3 bytes, a unit of digital information storage.

hyperplane – a subspace of one dimension less than its ambient space.

interference – see **negative transfer**

kernel function – functions that provide a way to map data as though it were projected into a higher dimensional space, by operating on it in its original space (Boser, Guyon, & Vapnik, 1992).

k-way comparison – the process of comparing entities in groups of size k to judge which of each entity is preferred (David, 1988).

first language – see **native language**

second language – see **non-native language**

marked – a phenomenon A in some language is more marked relative to some other phenomenon B if, cross-linguistically, the presence of A in a language necessarily implies the presence of B, but the presence of B does not necessarily imply the presence of A (Eckman, 1981).

method – a collection of programming statements that are grouped together to perform an operation (“Java – Methods,” 2013).

monocentric language – a language with one standard dialect or form (Clyne, 1992).

n-dimensional – a space or object that requires coordinates in at least n dimensions to specify any point within it (i.e. a line has a dimension of one, because it requires only one coordinate to specify a point on it).

native language (L1) – the language a person has learned from birth (Bloomfield, 1984).

native speaker (NS) – a person who was born and immersed in a language during youth, in a family where the adults shared a similar language experience as the child (Love & Ansaldo, 2010).

negative transfer – the imposition of the native language on the structure of the target language, impeding acquisition (Macken & Ferguson, 1981).

non-native language (L2) – any language learned after a native language (Richards & Schmidt, 2002).

non-native speaker (NNS) – an individual who is not a native speaker of a language.

pairwise comparison – any process of comparing entities in pairs to judge which of each entity is preferred (David, 1988).

pluricentric language – a language with several standard dialects (Clyne, 1992).

positive transfer – the imposition of the native language on the structure of the target language, enhancing learning (Macken & Ferguson, 1981).

random-access memory (RAM) – a form of computer data storage.

support vector – a linear combination of $N_S \leq N$ feature vectors that are associated with $\lambda_i \neq 0$. Vectors on the margin of a hyperplane (Theodoridis & Koutroumbas, 2009).

support vector machine (SVM) – supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis (Cortes & Vapnik, 1995).

vector – a representation of the measurements used for classification uniquely identifying a single pattern or object (Theodoridis & Koutroumbas, 2009).

APPENDIX B

FEATURE SET COMBINATIONS CREATED BY [ɛ]→[ɪ]

Input 1	Input 2	Input 3		Output 1	Output 2	Output 3
V			→	semi-close	near-front	unrounded
V			→	null	near-front	unrounded
V			→	semi-close	null	unrounded
V			→	semi-close	near-front	null
V			→	semi-close	null	null
V			→	null	near-front	null
V			→	null	null	unrounded
null	front	unrounded	→	semi-close	near-front	unrounded
null	front	unrounded	→	null	near-front	unrounded
null	front	unrounded	→	semi-close	null	unrounded
null	front	unrounded	→	semi-close	near-front	null
null	front	unrounded	→	semi-close	null	null
null	front	unrounded	→	null	near-front	null
null	front	unrounded	→	null	null	unrounded
null	front	unrounded	→	V		
null	front	null	→	semi-close	near-front	unrounded
null	front	null	→	null	near-front	unrounded
null	front	null	→	semi-close	null	unrounded
null	front	null	→	semi-close	near-front	null
null	front	null	→	semi-close	null	null
null	front	null	→	null	near-front	null
null	front	null	→	null	null	unrounded
null	front	null	→	V		
null	null	unrounded	→	semi-close	near-front	unrounded
null	null	unrounded	→	null	near-front	unrounded
null	null	unrounded	→	semi-close	null	unrounded
null	null	unrounded	→	semi-close	near-front	null
null	null	unrounded	→	semi-close	null	null
null	null	unrounded	→	null	near-front	null
null	null	unrounded	→	V		
open-mid	null	unrounded	→	semi-close	near-front	unrounded
open-mid	null	unrounded	→	null	near-front	unrounded
open-mid	null	unrounded	→	semi-close	null	unrounded
open-mid	null	unrounded	→	semi-close	near-front	null
open-mid	null	unrounded	→	semi-close	null	null
open-mid	null	unrounded	→	null	near-front	null
open-mid	null	unrounded	→	null	null	unrounded
open-mid	null	unrounded	→	V		
open-mid	null	null	→	semi-close	near-front	unrounded
open-mid	null	null	→	null	near-front	unrounded
open-mid	null	null	→	semi-close	null	unrounded
open-mid	null	null	→	semi-close	near-front	null
open-mid	null	null	→	semi-close	null	null
open-mid	null	null	→	null	near-front	null
open-mid	null	null	→	null	null	unrounded
open-mid	null	null	→	V		
open-mid	front	unrounded	→	semi-close	near-front	unrounded
open-mid	front	unrounded	→	null	near-front	unrounded
open-mid	front	unrounded	→	semi-close	null	unrounded
open-mid	front	unrounded	→	semi-close	near-front	null
open-mid	front	unrounded	→	semi-close	null	null
open-mid	front	unrounded	→	null	near-front	null
open-mid	front	unrounded	→	null	null	unrounded
open-mid	front	unrounded	→	V		

open-mid	front	null	→	semi-close	near-front	unrounded
open-mid	front	null	→	null	near-front	unrounded
open-mid	front	null	→	semi-close	null	unrounded
open-mid	front	null	→	semi-close	near-front	null
open-mid	front	null	→	semi-close	null	null
open-mid	front	null	→	null	near-front	null
open-mid	front	null	→	null	null	unrounded
open-mid	front	null	→	V		

APPENDIX C

RULE-SET INCLUDING FREQUENCY INFORMATION

[CONSULT ATTACHED FILES]

APPENDIX D

SUPPORT VECTOR MACHINE EQUATIONS

(1) Maximum-margin hyperplane

$$\max \lambda \left(-\frac{1}{2} \sum_{i,j} \lambda_i \lambda_j y_i y_j x_i^T x_j \right)$$

subject to $0 \leq \lambda_i \leq \frac{1}{N}$, i=1, 2,…., N

$$\sum_{i=1}^{N} \lambda_i \, y_i = 0 \, x$$

$$\sum_{i=1}^{N} y_i \geq v$$

Where x_i, $i = 1, 2,…., \text{N}$ are points on the decision hyperplane, v is a constant, T denotes transposition, λ is a Lagrange multiplier, and y is the mapping between the original input space and a higher dimensional space that is defined in the lower dimensional space.

(2) Nonlinear support vector machine classifier

Assign x in $\omega_1(\omega_2) if \; g(x) = \sum_{i=1}^{N_s} \lambda_i y_i K(x_i, \; x) + \; \omega_0 > (<) \; 0$

Where ω_i, $i = 1, 2,…., M$ are classes; x_i, $i = 1, 2,…., N$ are points on the decision hyperplane, N_S is the number of support vectors, λ is a Lagrange multiplier, $K(x_i, \; x)$ is any symmetric, continuous function, and y is the mapping between the original input space and a higher dimensional space that is defined in the lower dimensional space.

(3) Radial basis function

$K(x, z) = \exp(-\gamma \| x_i - x_j \|)^2$, for $\gamma > 0$

Where x_i, $i = 1, 2,…., N$ are points on the decision hyperplane and γ is a free parameter that can be altered for more accurate classification.

APPENDIX E

TEST RESULTS FOR LANGUAGE CLASSIFICATION

[CONSULT ATTACHED FILES]

APPENDIX F

TEST RESULTS FOR LANGUAGE FAMILY CLASSIFICATION

[CONSULT ATTACHED FILES]