LingoPros

Technological Feasibility

October 26, 2017

Josh Shaffer, Luis Montes, Erik Strauss, Matt Quintana

Sponsors: Dr. Okim Kang & Dr. David O. Johnson

Mentor: Ana Paula Chaves Steinmacher

NORTHERN ARIZONA UNIVERSITY

Overview: The technological feasibility document will detail the different technologies available to implement the project, which of them we have chosen to use, and why those technologies chosen will best serve in the completion of the project.

Table of Contents

Introduction

Non-native english speakers often have a hard time articulating their speech not just in what they say, but how they say it. Proper emphasis on words or just general speech tone is something that native english speakers do not have to think about in their everyday speech as it is something that just comes naturally. For example, in a conversation between a non-native English speaker and a native speaker, the non-native speaker may not be aware of how, by not matching the tone or cadence of the native speaker, the conversation can begin to feel awkward and unnatural. Recognizing these differences in tone, pitch, stress and other prosodic features between native and non-native English speakers, is one of the key focuses of Dr. Okim Kang's research at the Applied Linguistics Speech Lab (ALSL) at NAU.

Currently, students working with Dr. Kang at the ALSL have to do all of the speech analysis by hand, listening to audio samples and recording what features they hear or recognize. This process can take hours on end even for a small 10 second audio clip, so as such, an automated process of speech analysis that is easily available to the students would drastically improve their research capabilities by saving them time and energy. Ideally, Dr. Kang wants to have a program that is able to analyze recordings of non-native english speakers and grade their English proficiency based on aspects like tone and emphasis, as mentioned before. Dr. Kang currently has a speech analysis program that can recognize these features, however it is cumbersome to use as it is spread across two different machines and operating systems. Additionally, the model that the current program is based on, a framework based on a distinct form of speech analysis posed by David Brazil called *discourse intonation*,[1] is not recognized as the standard in speech analysis. A set of conventions called Tones and Break Indices (ToBI) is considered to be the standard for annotating speech features, and so Dr. Kang wants to show that the Brazil based program performs just as well or better than automatic analyzers based on ToBI.

We have been brought on to develop a speech analysis program based on the ToBI model where we will use an automatic labelling program called AuToBI to extract suprasegmental features from the speech audio files. We will then develop a machine learning program to score those extracted features. Once the speech analyzer is built, we are then tasked with creating a web application for Dr. Kang's framework (David Brazil Model) so that her program can be executed in one single location that can be easily accessed from anywhere.

In this document, we will start by outlining some technological challenges that we will encounter while developing our the speech analysis program and the web application. These are not necessarily all the challenges we will encounter during our development period, rather they are ones that we anticipate encountering. After that, we will outline some of the options that solve the problems outlined in the previous section. We will then analyze the pros and cons of each solution and determine which one would be the best to use in our solution. In the last section, we will talk about how we will bring our individual solutions to the technological challenges together into one coherent system which will both provide information on the difference between the Brazil and ToBI model, and improve speech analysis research for Dr. Kang and the ALSL.

Technological Challenges

We have narrowed our technological challenges down to four topics:

I. **AuToBI -** We will need to use an automatic speech analysis program to obtain results that are then passed to a machine learning program.

II. **Machine Learning-** We will need to create a machine learning toolchain program that takes the output from AuToBI which will be converted to suprasegmental measures

III. **Website-** We will need a reliable server and elegant website that allows ease of use with regards to our client's program running on a browser for everyone to use.

IV. **Database-** We will need a database backend that allows us to store user data such as, login information, previous results, and uploaded files.

Technology Analysis

I. AuToBI

Issue Introduction:

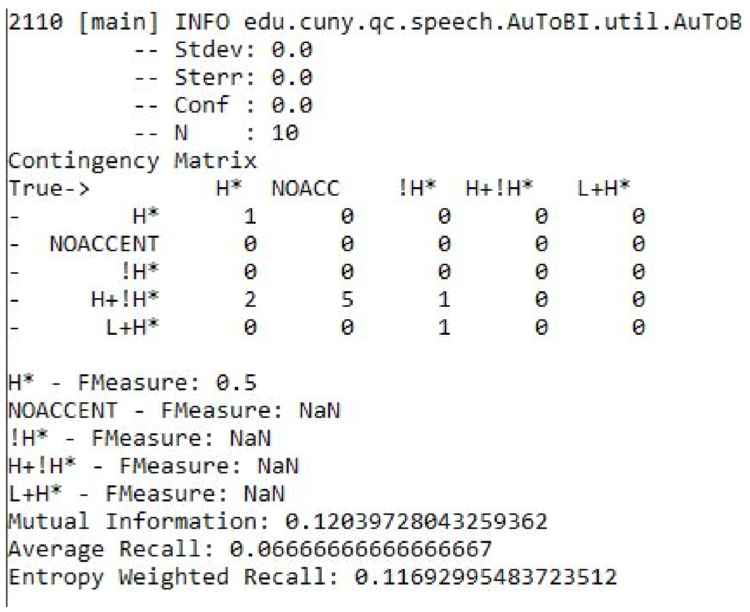
The first part of our project involves using a third-party speech analyzer called AuToBI to extract prosodic features from speech samples. The problems we face however are learning how AuToBI functions and if we will be able to successfully extract its results so that they can be compared to the Brazil framework. Determining whether or not we will be able to use AuToBI to extract the features we want is the basis for the first part of the project, and if it's unable to do so then the first part will not be viable.

Using AuToBI:

Developed by Andrew Rosenberg at Queen's College, AuToBI is a "Java toolkit that hypothesizes pitch accents and phrase boundaries"[1]. It can detect where intervals of silence occur within a speech file and create hypotheses using those measurements. With the use of previously trained analysis models, speech files are passed in and analyzed in order to detect: pitch-accent, phrase boundaries, and/or intonation.

The program is run from the command line where a word segmentation file, audio (.wav) file, hypothesis output file, and model parameters are passed to the program to perform its analysis. The program then returns results back to the command line, writes out its analysis hypotheses to an output file, and records the features used in the analysis along with their values into an Attribute-Relation File Format (.arff) file.

AuToBI output:



```
2110 [main] INFO edu.cuny.qc.speech.AuToBI.util.AuToB
         -- Stdev: 0.0
         -- Sterr: 0.0
         -- Conf : 0.0
         -- N    : 10
Contingency Matrix
True->          H*  NOACC    !H*  H+!H*   L+H*
-         H*      1      0      0      0      0
-   NOACCENT      0      0      0      0      0
-        !H*      0      0      0      0      0
-      H+!H*      2      5      1      0      0
-       L+H*      0      0      1      0      0

H* - FMeasure: 0.5
NOACCENT - FMeasure: NaN
!H* - FMeasure: NaN
H+!H* - FMeasure: NaN
L+H* - FMeasure: NaN
Mutual Information: 0.12039728043259362
Average Recall: 0.06666666666666667
Entropy Weighted Recall: 0.11692995483723512
```

Extracting Results:

The .arff files created by AuToBI contain a list of attributes, also known as features, and their related values from the analysis. AuToBI will also display results from a model's analysis on the command line in the form of a contingency matrix as well as displaying other measures produced by the model. We can then parse the results with either a third party application or by developing our data analyzer, however in both cases, we would have a means of extracting relevant features from AuToBI's analysis.

Conclusion:

The AuToBI toolkit, while daunting at first, is straightforward to use after understanding its different parameters and how it will output results. We can take these results and extract meaningful data from the program's ability to output an .arff file which contains the set of prosodic features used in the analysis. Extracting these prosodic features and passing them to the machine learning portion of the toolchain is a key part of the first part of our project, and AuToBI will provide a viable means of doing so.

II. Machine Learning

Issue Introduction:

The next step of our project involves a separate prosodic labeling model called ToBI which is similar to the David Brazil model our client is using to label prosody. Our client has already created a machine learning program to compare the measures output by their model and wants us to do the same for ToBI. The aforementioned third-party program for this other model is called AuToBI and our job is to create a machine learning tool chain that will take the output from the program and turn it into suprasegmental measures. These measures will then be scored by the machine learning software compared to what humans scored the measures, if the accuracy was not high enough the machine learning will reintroduce the measures with a different subset and try to improve accuracy. Our client wants to be able to compare her model's accuracy with the AuToBI accuracy because she believes her model is the better of the two.

Technological Options:

In order to decide which software we need we must figure out specific functionality will be necessary for our project to function. For example, the software must be able handle the output from AuToBI which is an Attribute-Relation File Format (ARFF) file and process it. The software must also visualize the data so that we can easily display the results for our client. Although there are hundreds of options for machine learning software available we have narrowed our choices down to four possible options. They include: Waikato Environment for Knowledge Analysis (Weka), Amazon Machine Learning(AML), MATLAB, and BigML. Upon researching different options we found these are the top options of software that meet our needs so we must compare them.

1. Weka

Weka is an open-source Java library that is imported to create a powerful machine learning program locally on a computer. Weka comes with a GUI, command line interface, and a Java API, that will allow us to visualize the data from the ARFF output then manipulate the data and its features the way we need to. A downside of this software is that it does not offer cloud based storage, you must install the application and run it locally. A negative review we saw is that the GUI is somewhat cluttered and hard to navigate without experience. A strong point for this option is that the team is most experienced in Java development compared to other languages.

2. AML

Amazon's Web Service version of machine learning which allows people with minimal prior experience to start developing programs on their cloud-based application. It is pretty simple to use and abstracts the algorithms being used so that the user does not have to fully understand them to use the program. It has many upsides compared to the rest as it is very easy to use and can be used directly on a browser, along with a lot of bells and whistles. However, the major downsides to AML is that it costs money, offers no free version, and it does not support ARFF files as input.

3. MATLAB

This is the language our client had used to create their machine learning program and suggested we look into it but also told us to consider other open-source projects. It is similar to Weka in that it is hosted locally and installs specific machine learning libraries. It offers some powerful tools but falls short compared to the quality of the others. The major negative feature of MATLAB is that none of the team members have significant experience programming in it.

4. BigML

BigML is similar to AML in that it uses cloud-based storage and can be ran completely on a browser, all that is necessary to get started is registering an account. At first there is a free version with a small data cap on storage so the software can be tested, but as for cost this software has a PRO version that can be paid for but is free for students and educators. BigML is also very simple to use and within minutes of testing it out with sample data files, the data was already being visualized with interactive models. After testing it out with an ARFF output file it did not work correctly on the web application but they do offer a Java API and it is possible to convert the file to the correct type.

Chosen Approach:

This table rates each of the options on a scale from 1 to 5 of how well it works for the specific feature (One being it does not work at all and five being it works well).

Name \ Feature	**ARFF File**	**Visualization**	**Ease of Use**	**Team Experience**	**Cost**
Weka	**5**	**3**	**3**	**5**	**5**
AML	**3**	**5**	**5**	**1**	**1**
MATLAB	**5**	**3**	**2**	**1**	**5**
BigML	**3**	**5**	**5**	**5**	**5**

Taking into account the above table of differences and the team's experience with Java we have decided to build our program in Java using the BigML Java API. Another reason is that it will help our client to easily visualize the data and understand the results with interactive graphs. They could even access it online using the web application rather than having to show them the results on our local machine. Although we did not choose Weka because it has many useful tools for ARFF files we will still have to use it in order to convert the files to the correct format for BigML.

Conclusion:

With our original problem of needing a machine learning software that is highly visual and can handle ARFF files we believe programming in Java using BigML is our best solution. We plan on creating a Java application that takes in an ARFF file from AuToBI as input, converts it into a Comma-separated values (CSV) file, and then inputs that into our BigML program. Using the inputs, the program will then score the suprasegmental measures and allow us to easily display those scores to our client with interactive models.

III. Website

Issue Introduction:

We have a toolchain plan described in subjects I and II on how we plan to build our speech analysis program, and for non-collaborative desktop use (e.g. a single linguistic scientist needing quick results for prosody measures from a sound file or files) the project phase one goals could be settled… but native-only use requires the users of our program to have particular operating systems and have to install what will likely be a memory intensive and space costly program. Phase II of this project is to convert our entire program to be an online application, usable from anywhere with an Internet connection and at minimum access to a web browser such as Firefox or Chrome. Web applications involve the "**backend**" and "**frontend**" paradigm where the implementation of hosting servers and networking aspects are handled in the backend and not accessed by the user, while the website style and user interfaces are implemented on the frontend which is where the user interacts with the application or website. We need high-speed, low overhead solutions in order to implement both "ends" accurately for the single major Phase II requirement of making our program usable from the internet. The added bonus goal is a LingoPros company goal: making our code bases efficient and maintainable for future Capstone group students or others.

The Back End: The Java application server.

Issue Introduction:

We need to be able to run the Brazil Model on our server and not client side since the program must be built with Eclipse and is operating system specific. Knowing that this MATLab based program can be converted Java classes, we know a Java application server must be set up and started so as to allow the Brazil toolchain to run in the backend.

Technological Options:

Since there are two major "ends" to web applications, it was important we study both sides to cover more technological possibilities our entire system could end up benefiting from. Starting with a backend, we realize we need an application server to run the David Brazil Model analyzer on any input sound file so as to return prosodic results. The David Brazil Model was written in MATLAB, but unfortunately there are no MATLab code specific application server options. However MATLab code can be easily converted into runnable Java classes from the MATLab environment's many converting tools. That said, a Java application server could run the Brazil application whenever a user inputs a sound file and send out prosodic results to the frontend client side, instead of needing our users to install the Brazil model, build it locally, and then run it on their machines. There are five competing mainstream Java application servers to pick from that could be used for the running the Brazil model on the server we use to host the web application.

All five application servers could potentially run the Brazil model for our website so it's highly important to perform our due diligence by weighing relative strengths and weaknesses of each one to see which one would be the best for us to use based on what is possible.

This table rates each of the features on a scale from 1 to 5 of how well the specific feature manifests itself with relation to the application server name (One being it hardly has the feature above it and five being it embodies extremely the above feature.).

Name \ Feature	**Setup Simplicity**	**Various Java IDEs Applicable Tools**	**Ease of Configuration after Setup**	**Team Experience**	**Documentation, Community, Tutorials**
Tomcat	**5**	**5**	**4**	**2**	**5**
Jboss	**3**	**5**	**4**	**1**	**4**
Glassfish	**1**	**3**	**3**	**1**	**4**
Jetty	**5**	**3**	**3**	**1**	**3**
Liberty	**3**	**2**	**5**	**1**	**4**

The results, Tomcat and Jetty have the easiest installation due to simple setup of just an archive extraction on the server after downloading either application server, and additionally both can be started in a single command from the command line [3]. Glassfish has the worst setup due to its vagueness in turning on the actual server after installation has occurred. For configurable options found in a Java IDE for the selected application server, Tomcat wins because of its near omnipresence on all major Java IDEs (IntelliJ, Netbeans, and Eclipse) for quick and widget like server configuration. Liberty serves as the worst in this aspect as it requires plugin installation on even some of the most popular Java environments in order to be configurable. For configuration of the server after startup, Liberty actually has the best of this feature because a developer can change its main .xml file settings without needing to restart or stop the server, meaning in real time changes can be made to the server for small fixes without alway needing to lock out users over a dreaded "server under maintenance" message anytime something needs fixing server-side. The worst configuration after startup developer experiences likely come from Jetty and Glassfish because both of their configuration files' defaults can be exceptionally wordy and complicated to many developers. On Team knowledge of application servers, all the options save for Tomcat and Luis' experience at the USGS with that kind of server, get the lowest score of one since none of the team members have any experience starting or configuring Java application servers other than Tomcat. Finally, extra detailed documentation and tutorials (all generated from the fairly active Java application server communities) are quickest to side with Tomcat due its old age and widest spread of use [4].

Chosen Approach:

We choose to go with Tomcat as our Java Application server for two main reasons. One, its feature proficiency total from the above comparison chart was the highest at 21 points and hence appears to have the best overall implementation/association of desired top row features/attributes. ITS here at NAU can set up the Tomcat server for us with all the defaults Tomcat provides, which saves time on the group setting up for this Phase 2 of the project and more importantly implies that our project can be hosted on NAU's Unix server named Jan.

<u>Demo:</u> Luis will make a small Java adder application that takes a number and return the number plus 2, test Tomcat application deployment by putting the simple adder on our ITS configured Tomcat server so the server will constantly run the adder, and then try to make Ajax get/post client to server requests involving integers from our simple CEFNS team page and aim for getting correct returned values (e.g. send 2 get 4, send 12 get 14, send 102 get 104). That demo

will prove the team can send and receive values to a Java application hosted server side through the NAU provided Tomcat server.

The Front-End I: Using a View Library or Not

Sub-Issue Introduction:

In Web 2.0, the use of view libraries for web applications became commonplace. The view library (or libraries) is a collection of prepackaged code for instantiating typically complicated DOM elements like frequently used panels for an application or widgets. These libraries are well tested and modularize building the the user interface. The question is, are we better off with or without implementing one or none of these view libraries in our application's front-end.

Technological Options:

Various options of view libraries all written for the web are Vue.js, Facebook's (and a growing web application development professional standard) React.js, or Angular.js (Google's competing view library).

Chosen Approach:

We choose to avoid using a view library for now just because of all the overhead required in using any of them (albeit some, like the TypeScript based Vue.js, are less complicated than others, like the wordy JavaScript based React.js) such as: the setting up (individual DOM element build processes on top of our server build processes, *yikes!*), the syntax learning curve (e.g. what complicated DOM components look like, and every one of the three example libraries handles this description differently), and the grammar learning curve (e.g. how to configure a library's DOM insertions). We currently do not foresee needing more than four screens to develop our multi-page application, with those four being a non-logged in homescreen, a login/create account screen, a logged in homescreen, and a results page that comes back from the server through AJAX (so the page will not be entirely rendered again from the server to the client) after a sound file has been input and ran in the server side Brazil Model. Hence we do not need a heavy duty view library for enforcing sometimes arbitrary Google or Facebook determined DOM standards on our application's frontend. It is unlikely we will be rewriting complicated DOM elements in multiple pages due to our small volume of pages that do not behave similarly. Rewriting the same complicated DOM constructions over and over again in different places of a many page web application is almost always a sign that a view library or view controller needs to be implemented in your project, but that situation is unlikely to be ours.

The Front-End II: Using the Best CSS Front-End Framework

Sub-Issue Introduction:

While we did pass a tentative ban on view libraries for our simplistic multi-page web application based on our current knowledge of what the client wants in Phase II, we can still try using plethoras of pre-built CSS/HTML/JS boilerplate templates from Material Design Lite (by Google), Bootstrap (by Twitter) or Foundation (by Zurb). Letting actual artists tell you how to make DOM elements look amazing is to heed advice, and while no one on the team claims to be a front-end artist we should not try to write our own artistic CSS classes.

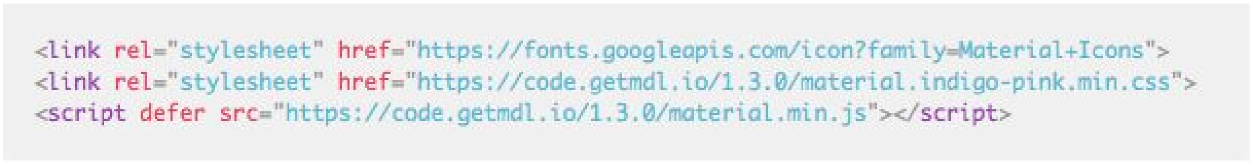
Technological Options:

This table rates each of the features on a scale from 1 to 5 of how well the specific feature manifests itself with relation to the application server name (One being it hardly has the feature above it and five being it embodies extremely the above feature.).

Name \ Feature	**Setup Simplicity**	**Aesthetic Variety**	**Team Experience**	**Documentation, Community, Tutorials**
Material Design Lite (MDL)	**5**	**5**	**3**	**4**
Bootstrap	**4**	**4**	**1**	**5**
Foundation	**2**	**4**	**1**	**3**

Chosen Approach:

We choose to go with Material Design Lite because of the setup being as simple as a script link in our main index.html looking like:



```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
<link rel="stylesheet" href="https://code.getmdl.io/1.3.0/material.indigo-pink.min.css">
<script defer src="https://code.getmdl.io/1.3.0/material.min.js"></script>
```

From MDL's Site: https://getmdl.io/started/index.html

Unlike, Bootstraps' CMS like setup or Foundation's small to large screen development with a GUI to add elements.

We believe MDL will provide the most unique style to our application because of its modular ability to add its CSS classes. We can use just how Google makes their styles their web button from them yet still make our own drop down menu if we wanted, for example. With the other two, it is generally all or nothing in implementing their look for certain DOM elements. Luis has had some experience with MDL from his work so this will not be too new of a front-end framework for the group to try implementing. Finally, MDL may have great documentation found here: https://getmdl.io/started/index.html, but BootStrap being more common is bound to have more tutorials and perhaps better clarified documentation, but we think the pros of modularity of new DOM styles, quick setup, and a mix of our own CSS with MDL will make for an overall better looking front-end than what we could get with Bootstrap or Foundation.

IV. Database

Issue Introduction:

One requested feature for the online web app is the ability for users to create and login to an account. This is not uncommon and it requires a database to be able to create and store user information like the username and password, which allows them to log on to their specific account. These accounts will be tailored to each user and will display account specific information such as the results of past analyses that they have performed. So the main things we will need a database system for would be for users to login to their account and then display information specific to their account.

Technological Options:

When it comes to database systems, there are different approaches that we could take. Of all the database software available, there are three that we believe would be the best to pick from: MySQL Workbench, phpMyAdmin, and MongoDB. Each have different pros and cons that will ultimately determine which one we will use.

1. MySQL Workbench:

MySQL Workbench is probably the software we are all most familiar with. We used it in CS 345 to create complex databases, so we have plenty of experience with it. Additionally, MySQL is very easy and efficient to use. One notable feature of the software is that it allows the user to create a database by designing a diagram, which can then be exported as MySQL code. That in itself could save time and help us visualize our final database as well. MySQL is also better at handling larger and more complex databases than other software, however, the tradeoff is that MySQL is typically more of a resource hog than other database software. In the case of our database not being very complex, it may be disadvantageous to use MySQL as we would not need its full power while still having it use up a substantial amount of resources.

2. phpMyAdmin:

phpMyAdmin is another software that we are all familiar with as we had to use it in previous classes. It can handle a lot of queries very quickly, but unlike MySQL Workbench, it does not have a diagram option for constructing a database, so it is harder to visualize the final product. This means that it may take more time for us to develop a database in phpMyAdmin than it would using MySQL. However, since our database may not be too complex (just handling logins and some previous analyses) it may be a better fit for us.

3. MongoDB:

Lastly is a database system called MongoDB. None of us have as much experience with MongoDB as with phpMyAdmin or MySQL Workbench. MongoDB is a “NoSQL” database, which means related data gets stored in a single document to allow for faster retrieval. While MongoDB may be faster than the other two options, reading up on it suggests that as a database grows, MongoDB may create duplications and inconsistencies which would ultimately hurt the performance of our database system. We have no way to tell how big our database will eventually grow once it is live, so we think that MongoDB might not be a safe pick for our project.

Chosen Approach:

After reviewing the three top choices for our database system, it comes down to either MySQL Workbench or phpMyAdmin. phpMyAdmin is quick with easy queries and less of a resource hog, and MySQL is easier to develop with and can handle more complex databases. In the end, we feel that MySQL will be the better choice due to having more experience with it and its ability to better handle larger databases.

Conclusion:

From here, we will create an example database using MySQL to make sure it can handle the amount of data and perform queries in a timely manner. We might also build one in phpMyAdmin just to compare to see if we really made the right choice or not.

Technology Integration

With regards to the first portion of our project involving ToBI we will be using AuToBI to create the suprasegmental measures that will then be input into our machine learning program using BigML Java API which will score the measures and predict their accuracy compared to human scores. This will allow our client to easily visualize the results and compare them with their model to see which is more accurate.

As for the second part of the project we will be setting up the client's program on an application server called Tomcat so that it can be used remotely on a browser for easy access. We will be saving user data such as, login information, uploaded files, and result history using a MySQL database that is created via MySQL Workbench.

Conclusion

Automatic speech recognition is a topic that has been around for several decades, however it is only recently that the field has advanced to the point where researchers are able to study the deeper subtleties of speech due to advancements in computational capability and machine learning. Linguistics researchers are now taking advantage of these new, better performing tools to carry out their analysis so that they can gain a better understanding of how humans communicate. Dr. Okim Kang is one such researcher who is taking advantage of these new technologies and using them to develop a new framework for automatic speech analysis.

Dr. Kang has asked our team to help develop a speech analyzer based on a classic speech analysis framework so that she may compare the results against her Brazil framework. In addition we have been asked to build a web application that can provide Dr. Kang and her students with a central location to analyze speech samples with the Brazil framework, rather than using the cumbersome and time wasting approach of having the framework spread across several different machines. Our solution will help Dr. Kang and her students at the ALSL conduct better research so that the field of speech analysis as a whole can gain a better understanding of how humans communicate.

This document has outlined which technologies are available, the advantages and disadvantages of each one, and which ones we have chosen as the best option for the development of each part of the project. For the first part of the project, the development of an intelligent system to score a speaker's proficiency in English, we have been tasked with using the AuToBI program, which we have shown to be a viable means of extracting prosodic features. We've explored the different machine learning tools available to analyze those prosodic features, and found that BigML with the use of Weka, will provide us with best means of easily analyzing the data as well as offering the best form of visual representation of the data. For the second part of the project, the development of a web application to house Dr. Kang's framework, we have found that our best options for storing user and analysis data will be to use MySQL as it provides both the capability to handle large databases and a familiar environment to work in. We've researched the numerous web development technologies available and found that in the development of the front end, the Material Design Lite template by Google will provide an easy means of formatting and designing our webpages, while on the backend, a Tomcat server would serve best due to its ability to host Java applications and all communications between the front and back end would be done through Ajax calls.

Tech Challenge	Solution	Confidence in solution
Viability of AuToBI	Provides necessary results.	High
Machine Learning Implementation	BigML Java API with possible use of Weka API	Medium to High
Website Implementation	Frontend: Material Design Lite for HTML/CSS formatting Backend: Tomcat server to host Java App Server-Client communication: Ajax	High
Database Implementation	MySQL	High

Based on our analyses we are confident that our chosen technologies will be able to fully build the desired product. We plan on making sure that BigML correctly works with our output file from AuToBI because we are not sure what suprasegmental measures we have to make from the output. We will figure this out by consulting with our client and getting the input files for their machine learning program and compare them. We believe we will be able to meet or surpass our client's expectations throughout the development of this project, and hopefully provide them with a useful tool for years to come.

Works Cited

[1] Caldwell Richard. 2012. Brazil, David. Blackwell Publishing Ltd.

[2] Andrew Rosenberg. 2010. AuToBI - A tool for Automatic ToBI annotation. Interspeech '10

[3] Shelajev, O. (2017). *The Great Java Application Server Debate with Tomcat, JBoss, GlassFish, Jetty and Liberty Profile.* [online] zeroturnaround.com. Available at: https://zeroturnaround.com/rebellabs/the-great-java-application-server-debate-with-tomcat-jboss-glassfish-jetty-and-liberty-profile/ [Accessed 27 Oct. 2017].

[4] W3techs.com. (2017). *Tomcat vs. Jetty usage statistics, October 2017.* [online] Available at: https://w3techs.com/technologies/comparison/ws-jetty,ws-tomcat [Accessed 27 Oct. 2017].

[5] Busche, L. (2015). *The Rundown: Bootstrap vs. Google MDL vs. Foundation - Treehouse Blog.* [online] Treehouse Blog. Available at: http://blog.teamtreehouse.com/the-rundown-bootstrap-vs-google-mdl-vs-foundation [Accessed 27 Oct. 2017].