Emmersion Learning: Primary Language Detection Project Proposal

Josh Greaves and Andrew Carr

December 19, 2017

1 Summary of Project

1.1 Overview

We will build a system to detect a user's first language based on accent information obtained during a standardized language assessment test.

1.2 Proposed system design

There are two possible systems that we could build. We would try both and see which one works best. The two systems are:

- **Standard Convolutional Neural Network:** This system has the benefit of being simple to implement and maintain allowing for additions in training data as more languages are analyzed. As such, it is quick to build and iterate with. It would be relatively simple to set up and get into production

  However, the main drawback to this system is that it would require fixed length audio sequences to be used at test time. These sequences might be too long, or too short, resulting in unnecessary test time or inaccurate results.

- **Dynamic Recurrent Neural Network:** This system would work in real time as the user is speaking and deliver a result as soon as it is confident. This is beneficial because it could potentially take less time to deliver accurate results. This type of architecture is more complicated to build and maintain, but comes with the benefit of potentially increased accuracy.

1.3 Potential complications

Since this is ground breaking work, there is no guarantee that either of these systems could work, we would have to reconvene to discuss our options if this is the case. However, we are very confident that this problem is tractable and reasonable to solve. As was shown in our proof of concept, with more data, and optimizations to the system, we believe it will work according to your specifications.

2 Deliverable

There are several potential services and products that could be provided, from very simple to quite involved.

- **Trained Model with documentation:** We would provide the fully trained model for you to load and integrate into your production system without any rigid API guidelines. This would be a quick and simple way to test the model in your system without lots of overhead.

  It would require more involved work on your end to integrate the model and being using it with users.

  Documentation would include methods, results, and usability design specifications.

- **C++ or Python package API for use in production systems:** We would provide the model with standard documentation and a useful API for a more traditional 'plug and play' approach. This would result in higher quality and more reusable code, but would take longer to be written and tested.

- **Web app with tutorials and examples of system in use:** We would provide a trained model with documentation, an API, and an example web application that employs the system.

- **Additional options:**
  - Testing Suite

3 Overall Budgets and Costs

The cost of this system could be variable depending on what services and products are desired. A basic outline is as follows:

3.1 Trained Model with documentation

Price for model only:

Documentation	$ 150
Testing	$ 200
Static CNN	$ 500
Dynamic RNN	$ 750
Total	$ 1,600

3.2 C++ or Python API

Price for Python API:

Documentation	$ 150
Testing	$ 200
Python API	$ 200
Static CNN	$ 500
Dynamic RNN	$ 750
Total	$ 1,800

Price for C++ API:

Documentation	$ 150
Testing	$ 200
C++ API	$ 400
Static CNN	$ 500
Dynamic RNN	$ 750
Total	$ 2,000

Price for C++ and Python API:

Documentation	$ 150
Testing	$ 200
Python API	$ 200
C++ API	$ 400
Static CNN	$ 500
Dynamic RNN	$ 750
Total	$ 2,200

3.3 Web App

Cost Analysis for Web App:

Documentation	$ 150
Testing	$ 200
Python API	$ 200
Web App	$ 500
Static CNN	$ 500
Dynamic RNN	$ 750
Total	$ 2,300

3.4 Additional options

Testing Suite: TBD

4 Timeframe

December 22 - Jan 5 Plan project Jan 5 - Jan 25 Write code and Test

5 Payment Schedule

$\frac{1}{4}$ before hand and $\frac{3}{4}$ upon delivery of product

6 Provisions

Free bug fixes for 1 month Standard rates apply after then