Project Echo Requirements Document

Alexander Douglass, Jacob Morris, Ben Reed, Luke Woodbury
December 7th, 2016

Description: Use deep learning (machine learning) to train an artificial neural network (ANN) to identify and classify speakers from recorded audio. We will train this network using existing datasets of recorded speakers including meta-data on their sex and dialect/language. Then, the network should be able to identify these speakers saying things that were not used in the training as well as correctly classifying their gender and dialect.

Requirement Document Questions:

1. What exactly is it you are trying to build?
A program capable of recognizing and classifying people based on recordings of their speech.

2. Have you thought through the features of your project, or are you "winging" it?
We have identified clear goals for our program. See use cases.

3. What risks do you expect to encounter while building this project?
Our biggest risk is not being able to fully optimize and train our ANN due to potentially long training times. Additionally, we cannot be completely certain that the problem is solvable through our proposed method. However, even failure will yield useful information about the capabilities of deep learning in this problem domain.

4. Do you have a reasonable chance to complete this project in time?
Given that our goal is to optimize as much as possible in the allotted time, completion of the project should be achievable. That said, for us to feel that we have truly succeeded in this project will require impressive results. This is a very hard problem that we do not expect to solve completely.

Use Case 1: Evaluate voice recording

Primary Actor: *User*

Scope: System with trained Artificial Neural Network (ANN).

Level: Identification and classification of speaker.

Brief: User supplies sound file of English speaking subject. Software analyzes this file and returns a speaker identification as well as classifications of the speaker (age, dialect, gender, etc.).

Stakeholders: AI Researchers, Linguists, Software Applications.

Postconditions:

> **Minimal Guarantees**: An answer is given.
>
> **Success Guarantees**: Correct answer is given (in the case the speaker is not present in training data, the identification returned will be considered an indication of greatest similarity).

Preconditions: The ANN is trained and the user is able to provide a .wav file containing the subject English speaker's voice.

Triggers: The user decides to use the program.

Basic flow:

1. The user opens the program.
2. The user provides a suitable .wav file.
3. The file is examined and values are returned for speaker identification and classification.

Use Case 2: Train ANN

Primary Actor: *Trainer*

Scope: System with untrained or partially trained ANN and an existing data set of speakers.

Level: Trained ANN capable of identifying and classifying speakers.

Brief: Trainer supplies training set of sound files along with meta data. It will train on each file.

Stakeholders: AI Researchers, Linguists, Software Applications.

Postconditions:

Minimal Guarantees: ANN is trained.

Success Guarantees: ANN is trained and effective.

Preconditions: The ANN has to be set up.

Triggers: The trainer decides to train the ANN.

Basic flow:

1. The trainer opens the program.
2. The trainer provides the training set and meta data to the program.
3. The program processes this information and trains itself.
4. The trainer is notified of the performance of the ANN on a test data set.

Use Case 3: Optimize ANN

Primary Actor: *Optimizer*

Scope: System with untrained or partially trained ANN and an existing data set of speakers.

Level: Optimized ANN capable of identifying and classifying speakers.

Brief: Optimizer supplies training set of sound files along with meta data. It will train on each file. This process will be repeated with variation (either manually or through use of an evolutionary algorithm) to explore space of possible solutions.

Stakeholders: AI Researchers, Linguists, Software Applications.

Postconditions:

> **Minimal Guarantees**: Highest performing ANN configuration found during the search is saved for use in use case #1.
>
> **Success Guarantees**: Highest performing ANN configuration found during the search is significantly better than the average solution.

Preconditions: The ANN and evolutionary algorithm have to be set up.

Triggers: The optimizer decides to optimize the ANN.

Basic flow:

1. The optimizer opens the program.
2. The provides the training set and meta data to the program.
3. The program processes this information and trains itself.
4. The optimizer is notified of the performance of the ANN on a test data set.
5. Do one of the following:
   a) Return to step 1, but using a new ANN configuration.
   b) Return to step 1, but using a randomly mutated ANN configuration (evolution).
   c) End. Keep best performing ANN found as solution.