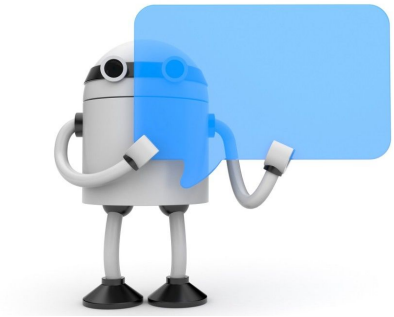
PENNSTATE

Department of Computer Science

PSU RERC on AAC 3: Active Listening Chatbot
Final Report

May 1, 2017



Michael Judge
Vishnupriya Bakthisaran
Daniel Levine

No Intellectual Property Rights Agreement Applies
No Non-Disclosure Agreement Applies

Executive Summary

Parents play a critical role in the development of communication skills in their child who uses Augmentative and Alternative communication (AAC); however, they frequently report their frustration with their relationships with communication professionals (McNaughton 2015). Active listening skills greatly bridge the gap between the family of the child using AAC and the professional, providing better outcomes. Thus, teaching active listening skills to pre-service SLPs is an important component of their learning program. Role playing is an important activity in an active listening program. In role play, one participant will play the role of the SLP and the other participant will play the role of the parent of the child using AAC. This project will examine the development of a chatbot to play out the various scenarios that are typically used during role-play in an Active Listening program. This chatbot will be used to determine the usefulness of chatbot technology in the context of an active listening program for pre-service SLPs in future studies. The two main objectives of the project were the creation of an administrator and user site. The administrator site has an interface enabling the admin to add scenarios for the bot to learn and then role-play with users. A scenario contains information such as the name of the person, description, and example responses for the bot to use. This scenario object gives the bot a personality. The user site supports interaction with a chat widget, and detailed score reports for completed conversations. A user can chat with any of the bots playing out the scenarios in the database. The site is supported with the Flask, and MongoDB python modules. The bot is a retrieval-based agent that uses a Naive Bayes classifier to converse with the user. We have met the need of building a system that supports a user and administrator site. Future work could improve upon the classifier, and improve the aesthetics of the website.

Table of Contents

1.0 Introduction

Parents play a critical role in the development of communication skills in their child who uses Augmentative and Alternative communication (AAC); however, they frequently report their frustration with their relationships with communication professionals. A recent study examined the effect of instruction in an active listening strategy on the communication skills of pre-service Speech-Language Pathologists (SLPs). Participants and parents of children who use AAC described their post-instruction interactions more positively than the pre-instruction interactions (McNaughton 2015). This indicates that active listening is an important skill for SLPs to learn in order to effectively communicate with parents of children who use AAC.

Role playing is an important activity in programs used to teach active listening skills. In role play, one participant will play the role of the pre-service professional and the other participant will play the role of the parent of the child with a disability. The participant playing the role of the parent needs to learn and/or use a script in order to converse with the pre-service professional. This project will examine the development of a chatbot to play the role of the parent of a child with a disability.

1.1 Problem Statement

Professionals who work with families need to demonstrate strong communication skills. An important skill that can be used to communicate effectively is active listening. These skills can be practiced in role plays, but it would be useful to determine the effectiveness of chatbot technology used in an active listening program. A chatbot would eliminate the time needed for a student to learn a script to converse with a partner. Ideally the Chatbot could handle a 3-5 minute conversation in which a parent has a concern, and the teacher practices (1) greeting the parent, (2) asking questions to find out more about the concern (3) summarizing the parent's concern (4) suggesting a first step (McNaughton).

1.2 Objectives

The chatbot will be a retrieval-based agent rather than a generative-based agent. This means, that it will be able to respond to what the user is asking with the most likely response in its knowledge base, but will not make up responses on its own. Thus, the responses must be pre-scripted. The bot must retrieve the correct response to user queries with high likelihood while also letting the user know when it is not sure what they are asking. The chatbot will make its decisions on what to say by using machine learning and natural language processing techniques. The system must support a database in order to allow the admin to add multiple scenarios that the bot can enact. Administrative use must be simple and therefore must not include any programming. The aim of this project is to have developed an administrative and user website. The administrative site will allow the admin to add/edit/remove scenarios that the bot can enact. These scenarios will contain example dialog, name of the person that the bot will be, and a brief description of the scenario to be displayed to the user. The user site will display a chat interface that the user will use to interact with the bot. Useful information such as a picture of the person they are interacting with, and a brief description of the scenario will be displayed as well. Lastly, the project must be well documented using good coding practices to ensure that the project can be improved upon in the future.

2.0 Project Management

2.1 Communication and Coordination with Sponsor

For this project, we had weekly meetings with our sponsor to ensure that communication of the requirements, and implementation was smooth. We also agreed to streamline our project process documentation by using the Trello platform. In our first meeting with our sponsor we came to a common agreement to use Agile-oriented productivity techniques such as scrum to accommodate flexibility of the project requirements. Our sponsor desires the end product to be scalable, so that it is flexible to changes in requirements. Hence, we decided to adopt the Agile methodology as our project implementation method to ensure that the end product is scalable and can evolve to fit future requirements.

2.2 Project Timeline

January 19th: we met with our sponsor for the first time and discussed future visions for the project. Discussed the real world application and got more information on the context for our chatbot. We began looking into background readings and considered various approaches.

January 26th: Established weekly meeting times with our sponsor. Completed base background readings and began to investigate the tools used in previously implemented chatbots.

February 2nd: Finished the background readings covering existing research on chatbot technologies. Researched the difference between deep learning and database chatbot technologies. Began work on various prototype chatbots.

February 9th: We have a base Naive Bayes program running that we were able to show the sponsor. We have decided to continue with this approach for the remainder of the project. More concrete goals, such as expectations for interactions and hosting issues, have been created.

February 16th: Started investigation of how to connect the system with Moodle. Developed a clear scope for the project.

February 23rd: Chatbot prototype has basic prompts. Currently works with one scenario. Moodle account access created in order to test the moodle integration when the project gets to this stage.

March 16th: Divided work between group members. Deciding upon Moodle integration with application. The layout of the project poster is being planned out.

March 23rd: Vishnu is working on putting the poster together. Dan is working on the front end. Mike is working on the backend. We are working on preparing a pilot for April 21-24.

April 6th: Worked on mockup admin front end. Database is setup. We are using MongoDB. Admin login page finished. Backend finished.

April 13th: Worked on linking backend and frontend. Compiled a list of dependencies for the project.

April 20th: Application is fully linked and fully functional. User interface is polished off. Preparing to install code on web server.

April 27th: In process of installing on web server. Poster presentation complete.

2.3 Deliverables

At the conclusion of the project, the aim is to have produced an administrative and a user site. The administrative site will make the process of adding/editing/removing Scenario objects to the knowledge base simple. The user site will allow users to communicate with the chatbot acting out various scenarios.

2.4 Budget

This project did not require the purchase of any materials.

3.0 Detailed Design

3.1 Component and Component Selection Process

For this project we chose a variety of components. The selection for these components was based on which tool was best suited for our needs. First off we decided to write the backend in python. Python is a very flexible and easy to use programming language with rich libraries. Whomever wants to pick up this project in the future should have an easy time getting up to speed with the coding. For the web server we used the Flask framework. Flask is a python module that enabled us to link the server code with the html and css files used to design the website. We needed to decide upon a database for the project. We came to the conclusion that MongoDB would be the best option as it integrates with our project the best. Lastly, we decided to write our own classifier rather than using services such as Watson or Wit.ai because neither offered the integration we needed for this project. Both had a bit of a learning curve to use, and if the service is discontinued then so is the project as all data would live on their servers.

3.2 Object Oriented Design

The project must be able to support multiple scenarios that the bot can roleplay. Scenarios contain information about a given conversation that a professional might have with a family. This information includes the name of the person the professional would speak to, a description of the person, and a Dialog object for example. The Dialog object contains a list of Response_Node objects, and can perform useful operations on this list. Response_Node objects contain a response that the bot can say, and a list of questions that yield this response. The questions are utilized in the Naive Bayes classifier. The list of Response_Node contained in neighbors are responses that are likely to come after this response. This list is used in the classifier to enable basic context aware conversations. Points are awarded for a user evoking this response in a conversation. A Bot object uses a Scenario and a smoothing constant to form its knowledge base. The Bot_Manager contains a dictionary that maps a scenario_id to a Bot. This class is used to avoid having to create a new bot every time a user initiates a chat with a particular Scenario. The Scenario_DB contains useful operations to interact with the database. The UML diagrams are displayed below.

Response_Node
-response: String -questions: List of Strings -neighbors: List of Response_Node -points: Int
+Response_Node(String, List, List, Int) +getters and setters +add_question(String) : void +remove_question(Int) : void +set_question(Int, String): void +get_question(Int): String +add_neighbor(Response_Node) : void +remove_neighbor(Int) : void +get_neighbor(Int): Response_Node +set_neighbor(Int, Response_Node): void

Dialog
-responses: List of Response_Node -total_points: sum of points in responses
+Dialog(List) +getters and setters +set_total_points(): void +add_response(Response_Node) : void +remove_response(Int) : void +get_response(Int) : Response_Node +set_response(Int, Response_Node): void +get_length(): Int

Scenario
-name: String -description: String -image: String -dialog: Dialog -video_link: String
+Scenario(String, String, String, Dialog, String) +getters and setters

Bot
-kb: list -transition_prob: dictionary
+Bot(Scenario, Float) +reply(Int, String) : List

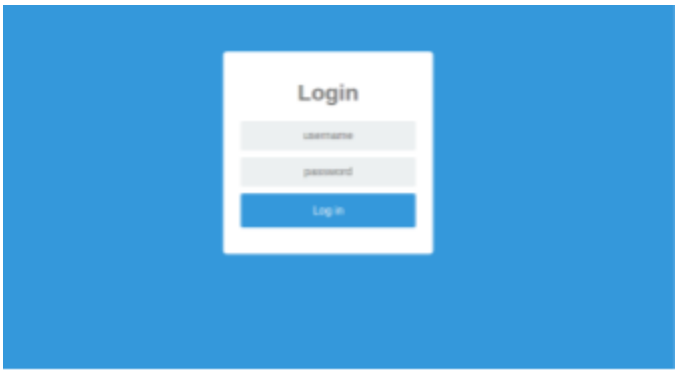
Bot_Manager
-bot_map: dictionary
+Bot_Manager() +get_bot(String) : Bot +load_bots() : void

Scenario_DB
+Scenario_DB() +get_scenarios() : List +add_scenario(Scenario) : String +update_scenario(String, Scenario) : boolean +delete_scenario(String) : boolean +get_scenario(String) : Scenario +wipe_db() : void +export_raw() : List +import_raw(List) : void

3.3 GUI Design

On the user facing site, we have developed a page that display a typical chat widget along with useful information about the person that the bot is role playing. When a user is finished with the chat, they can click the end chat button. Clicking this will take them to a results page which summarizes the conversation with a final score report. There is a button on this page that when clicked will download the report as a pdf.

We also developed an administrator site. First, if the administrator isn't logged in they will be greeted with a login page where they will need to provide a correct username and password to get into the admin homepage. The administrator homepage provides a variety of options for the admin to use to modify the database. If they click the New Scenario button or click on one of the links next to an already created Scenario, they will be taken to the edit Scenario page. On this page the admin can edit all of the fields that belong to a Scenario object. They can use the dropdown menu to select a response in the Dialog object contained in the Scenario. Selecting a Response will redirect the admin to the edit Response page. Here, the admin can edit all of the fields in a Response_Node object. Any additions/edits/deletions performed on the previous mentioned admin pages are immediately reflected in the database. Screenshots are displayed below.



Admin Login Screen

- Joseph: The parent of a child using AAC View
- Mike Judge: A senior at Penn State. View

New Scenario

Refresh Bots

Choose File No file chosen Import Database

Export Database

Wipe Database

Logout

Admin Homepage

Home

Image:

Choose File No file chosen Upload Image

name: Joseph

Description:

The parent of a child using AAC

Youtube Video URL: https://www.youtube.com/embed/MfI010cq_Kvk

Total Points Possible: 5

responses:

Select a Response Add Response

Save

Import Scenario

Delete

Edit Scenario Page

Go Back

Response: I have spoken to many people about this problem Points: 1

Questions:

Model Question: Have you spoken with anyone else about this? Delete Question

Alternate Question: Have you consulted anyone else about this problem? Delete Question

Alternate Question: Have you talked to anyone yet? Delete Question

Add Question

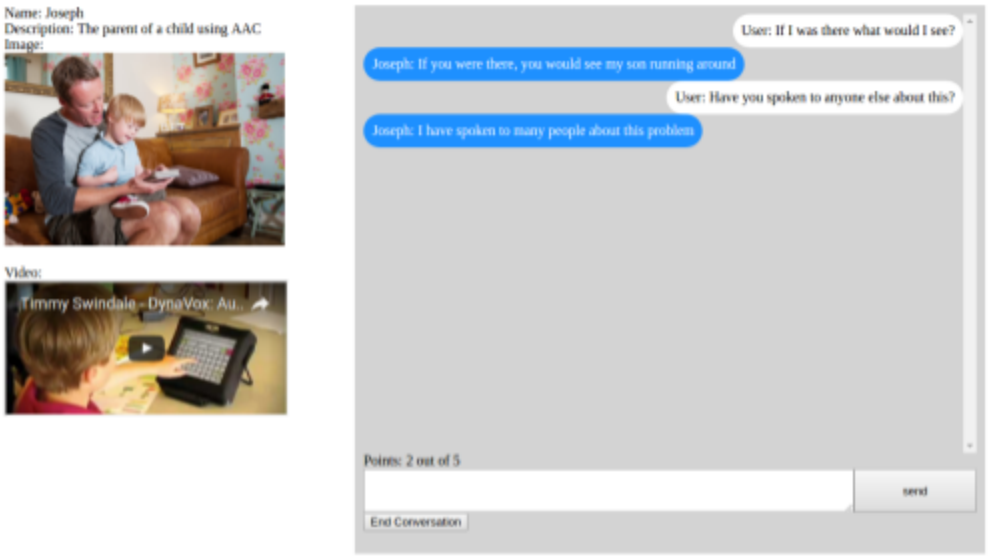
Neighbors:

Add a Neighbor: Select a Neighbor

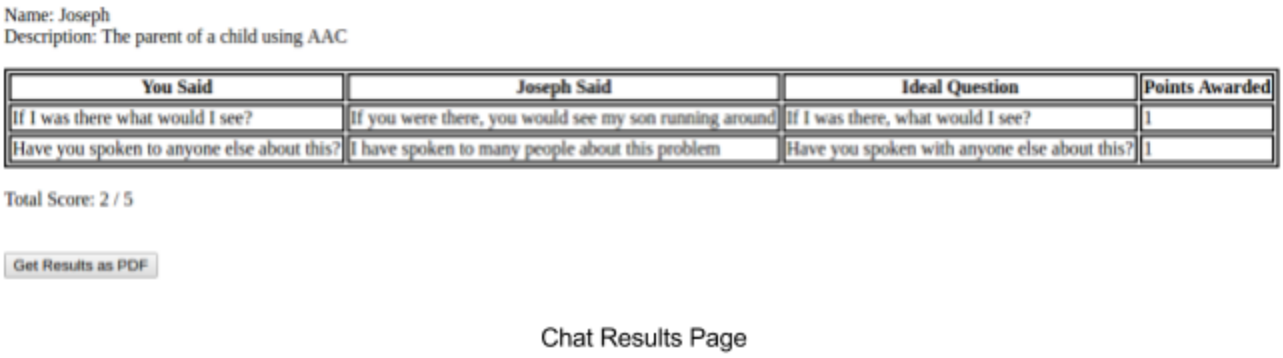
Save

Delete

Edit Response Page



Chat Page



Chat Results Page

3.4 Database Design

This project utilizes MongoDB to store the Scenario objects. MongoDB is a document-oriented database. Scenario objects are stored in the database by serializing all the fields in a Scenario object, and the objects it contains into a dictionary. When the dictionary is added to the database, a unique id is added to the dictionary. This scenario id is used throughout the system to access the Scenario to edit it or remove it. The Scenario_DB class contains all of the methods to support the serialization and deserialization operations. The client using a Scenario_DB object only needs to provide a scenario_id and they will be given a Scenario.

3.5 Use Case Definitions

There are two main inputs that can come into the system. These are the user input, and the administrator input. All website interaction is handled through the Flask module. Each website entry point has its own function defined in app.py. The input to these functions are parameterized in the site URL. The functions handle the input and return an html view to the web browser.

In the user input case, user input comes into the system in the form of a question string. After the user clicks submit, the server retrieves the input from the text box on the page. The server uses the site address to retrieve the scenario_id. The scenario_id is used to get the Bot that plays the Scenario from the Bot_Manager. Next the user input, and the previous response id (which is stored on a user session variable) are passed to the Bot object. The Bot object calculates and sends a list of the most likely responses from greatest to least along with corresponding response ids for each. The system updates the session variables for this user, and lastly refreshes the page with the most likely response appended to the conversation dialog.

In the administrator case, the admin has the ability to add/edit/remove scenarios. The Flask module directs all admin actions on the website to their respective functions defined in app.py. Depending on the action, the system may need to retrieve/add/edit/delete Scenarios in the database. The Scenario_DB object supports all the actions mentioned. The admin website is updated to display all information that is currently contained in the database.

3.6 Interface Design

Below is a table describing the interface of the project. The services above rely on the services below them.

User/Admin Website (HTML and CSS)
Flask Module
app.py (URL endpoints get routed here by Flask)
Bot_Manager
Scenario_DB
Bot
Scenario
Dialog
Response_Node

3.7 Test Procedure

Mostly this project was tested through unit-testing. As we developed the program, we tested a variety of inputs, and states that the system can enter for correctness and speed. Our sponsor tested our website by using it in our meetings and we adjusted the site interface and functionality based on his input. A user friendly interface was a priority for this project. We did not get the chance to test out the website on a large scale, but we hope to have the website hosted on the server by the end of finals week. The next group that works on this project will need to perform load testing amongst others when the website goes live.

4.0 Final Discussion

4.1 Construction Process

This project is hosted on github at https://github.com/MikeJudge/Chatbot. Running the program is as simple as cloning the repository and installing the required dependencies. It is advised that you download these libraries in the following order in order to not run into any errors:

- flask
- nltk
- pymongo
- cffi
- cairocffi
- flask_weasyprint

Then, run the main source file “app.py” and it will run the application on your system. For more details on the installation procedure consult the github page linked above.

4.2 Test Results and Discussion

We unfortunately did not have enough time to run a pilot with a group of students to test our program. We did however have weekly meetings with our sponsor whom was satisfied with where we took the project this semester. Every major function that was required was implemented along with some useful ones we came up with such as the import and export database function for example. The system runs well on our own machines. We hope to have the service available on a Penn State server owned by our sponsor by the end of finals week. At this time there are no observable bugs in our code. One area that could use improvement is the interface. It could probably look better. It’s intuitive, but not the greatest looking website to use. This is definitely an area for the next group to explore. Overall we are happy with where we took the project. We started from nothing, and got this project to a point where the major components are wired up and fully functional. Improvements can be made, but the overall layout of the system will more than likely remain the same. The modularity and readability of the code will ensure that future groups will be able to understand the work we performed quickly. For more details on the work that was performed go to section 3. In the future our sponsor plans to test the application on students that plan on becoming pre-service professionals. As a professor he will have a good testing size of students to make sure the application works correctly.

5.0 Conclusions and Recommendations

At the start of this project, the problem was very broad. We were assigned to develop a chatbot to train pre-service professionals in the context of an Active Listening program. Active Listening programs involve role-playing as a component of the instruction. These role-plays typically last 3-5 minutes, and the partner playing the role of the patient uses a script to answer questions. With this information the obvious choice was to develop a retrieval-based agent rather than a generative-based agent. As data for these kinds of conversations is limited we decided to use a Naive Bayes classifier as the retrieval algorithm as it can still perform well given a limited amount of data. Our sponsor liked the idea and the accuracy of its responses in our early terminal program, so we kept with this classifier. We explored other services already created on the market such as Watson and Wit.ai, but none of them offered the fine grained control and ease of use that our sponsor desired. In a typical Active Listening program there are many different role-plays that the students practice. Thus our system needed to support multiple scenarios with each chatbot using a classifier specific to a given scenario. A large portion of the development time went into developing an interface to enable an administrator to add/edit/remove scenarios. To support this we needed a database system and a module to support a web server. For the database portion we picked MongoDB. We use Flask for the web server. Both of these modules have excellent APIs and were easy to pick up. They perform their needed functions well. Once we developed a website and linked up the backend and fronted, we had our sponsor test it. Our sponsor was happy with the ease of use, and all of the functionality that our system offered. There were improvements that were asked to be made such as the naming of the fields and functions displayed to the admin. Also some adjustments to the score results page of the user site needed to be made. Adding a picture and Youtube video to the chat site was a request as well. In the last stretch of the semester we implemented all of the above mentioned tasks, and ironed out all bugs we ran into.

Overall we were all happy with how the project turned out. In the appendix there is a self assessment of the project. We met our sponsor's initial need of the chatbot. We also implemented an easy to use website to enable the chatbot to play out an infinite number of different scenarios. The code is documented well, modular, and easy to read. It was a lot of work, but we all learned a lot from the project. There are always improvements that can be made to any system, but here are a few that we think would be good places to start. Improvements to the classifier. The classifier could be improved with more sophisticated preprocessing and tokenization methods. Also using a two level classifier could improve performance as well. The first classification would classify the input into a section of a conversation (greeting, body, conclusion), and the second layer would work as currently implemented on the subset of responses in the narrowed down class. The interface could also be improved upon. A more modern design could be used to improve the look and feel of the administrator site.

References

Chatbot. Digital image. N.p., n.d. Web. 15 Feb. 2017. <https://onlinelearninginsights.files.wordpress.com/2016/06/chatbot_dm.jpg>.

McNaughton, D., Thistle, J. (2015). Teaching Active Listening Skills to Pre-Service Speech-Language Pathologists: A First Step in Supporting Collaboration With Parents of Young Children Who Require AAC. *Language, Speech, and Hearing Services in Schools, 46*, 44-55.

Appendix

Meeting the Sponsor's Needs: 9

We were one week short of getting the system launched on the server. We may or may not get it running this last week of class. Other than this small shortcoming, we completed everything else our sponsor asked of us. Our sponsor is very impressed with our work and is excited about where this project will go in the future.

Global and Societal Needs: 10

This project will be used by our sponsor in his own research. The research will test the effectiveness of using a chatbot in an Active Listening program. The hope is that our chatbot will improve the outcomes of these Active Listening programs. Active Listening is a very important skills for professionals to learn to improve the outcomes of people with disabilities. The completion of this project is a very important step needed to complete this study.

Daniel Levine

Software Engineer

dul186@psu.edu
301 S. Pugh St.
State College, PA
412-992-7381

EXPERIENCE

Penn State: ARL *Undergraduate Research Assistant*
State College, PA

Performed comprehensive research and received training in advanced research methods. Evaluated, benchmarked, and developed applications using the latest open source packages in streaming and analyzing big data for a cloud environment.

Blackhawk High School *IT Personnel*
Beaver Falls, PA

Provided support to the school's network systems and facilitated issues as a team that would arise with the faculty's systems (hardware and software).

PROJECTS

Virtual Reality Research

Collaborated with Penn State faculty and peers to develop an immersive virtual reality (IVR) system with the purpose to provide online education a full classroom experience.

Anxiety and Depression Anticipation App

Working with a team of people to create an app to detect and combat depression and anxiety in students. This app will be used for studies in the psychology field at Penn State.

Browser Extensions

Designing multiple browser extensions. The first of which is a browser action that gives greater access to user applications. The second is a page action that takes dates/times and makes suggestions to add them to the user's personal Calendar.

EXTRA CURRICULAR

THON Volunteer

Gave my time and effort to help the Panhellenic Dance Organization known as THON. Spent many hours collecting money, preparing, and working volunteer shifts at the Bryce Jordan Center to help fight pediatric cancer.

EDUCATION

Pennsylvania State University

State College, PA
B.S. Computer Science
Graduation: May 2018
GPA: 3.75

LANGUAGES

Java
Javascript
C/C++/C#
HTML/CSS
Swift
Objective-C
Python

MICHAEL R JUDGE

10 Maple Street
Wapwallopen, PA 18660
(570) 417-8893
https://github.com/MikeJudge
mij5171@gmail.com

Education

Bachelor of Science in Computer Science
Minor in Mathematics
The Pennsylvania State University, University Park, PA
Expected Graduation: May 2017
Cumulative GPA: 3.88/4.00

Relevant Coursework:

Object Oriented Programming (Java)	Linear Algebra	Discrete Mathematics
Systems Programming (C)	Probability Theory	Operating Systems
Data Structures and Algorithms	Number Theory	Mathematical Statistics
Graph Theory	Artificial Intelligence (Python)	Theory of Computation

Currently studying University of Washington's Machine Learning specialization on Coursera.

Experience

2048 (Independent Project)
- Developed and published a clone of the popular 2048 game to the Google Play Store (https://play.google.com/store/apps/details?id=thebiggestgame.judge.a2048)
- Designed the User Interface in XML, and wrote java code to match the mechanics of the game.
- Tested on many virtual and physical devices by utilizing Android Studio to ensure compatibility with many device configurations.

Spam Filtering System (Course Project)
- Developed artificial intelligence using Python that is able to correctly detect spam versus not spam on 99.5% of test data.
- Tested and analyzed various features to improve accuracy such as bigrams, email header info, and word types etc.
- Performed calculations in log-space to avoid underflow.
- Utilizes Naive Bayes to make predictions.

PSU Blue Screen App (Independent Project)
- Created an Android application using Java which allows users to view campus announcements, and class cancellations.
- Parses HTML code of the Penn State Blue Screen website to populate the information in the application.
- Required independently learning how Android applications are developed.

Tagline Device Driver (Course Project)
- Built a systems program in C to provide an interface to read/write taglines stored in a RAID array.
- Implemented a block allocation and management algorithm to store/read tagline blocks.
- Incorporated the following features: block cache, disk recovery, and network connectivity.
- Used GDB to debug the program.

Awards

Member, Tau Beta Pi | Member, Eta Kappa Nu | Member, Phi Kappa Phi

Activities

Calculus II Math Tutor, The Pennsylvania State University
Volunteer, American Red Cross
Volunteer, Shaver's Creek Fall Festival
Competitor, Computer Science Olympiad (2013, 2015) - Awarded 1st place in a programming contest

Vishnupriya Bakthisaran

vishnubakthisaran@gmail.com
Mobile: (269) 861-3072

Current Address
242 Simmons Hall
State College, PA 16802.

Permanent Address
1440 Waterbury Rd.
Breinigsville, PA 18031

EDUCATION

The Pennsylvania State University– University Park, Park PA — December 2017
B.S. Computer Science

SKILLS

- Programming Experience – C++, Java, C, MATLAB, HTML, Excel.
- Languages – Spanish (basic proficiency), Tamil (professional working proficiency), Hindi (Basic Proficiency)

RELEVANT PROJECTS

File Sharing TCP/IP Server and Client — May 2015

- Worked as a part of a team to implement a file sharing server in C using TCP/IP protocol with a UNIX based platform where I designed a file-sharing client that would communicate with server.

Single Cycle Processor — December 2015

- Implemented a single cycle processor in Verilog that provided functionality to instigate simple instructions such as branch statements, arithmetical and logical operations.

Hotel Reservation System — December 2014

- Constructed a hotel reservation system GUI in Java that would create hotel reservations for a customer.
- Created a project planning schedule by using Agile based methods and documented sprints to ensure that the project was completed smoothly on time.
- Applied testing procedures in Java via JUnit to ensure high quality of program to the clients.

Turing Machine — May 2013

- Worked as a part of a team to create a 4-tuple Turing machine in C++ where I created test cases to verify the quality and the effectiveness of the program.
- Prepared a professional project report that addressed all aspects of the project and successfully produced quality work within a limited time constraint.

ADDITIONAL EXPERIENCE

Pollock Dining Commons — August 2016-December 2016
Food-Service Worker

- Provided prompt quality customer service to a diverse clientele.
- Assisted in setting up food displays and service stations to increase team efficiency.

ACTIVITIES & AWARDS

- Member, Lion Ambassadors (Fall 2013- Spring 2014)
- Member, Association of Women in Computing (Fall 2015- Spring 2016)