Sharing the Load - Offloading Processing and Improving Emotion Classification for the SoftBank Robot "Pepper"

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Shawn Savela

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Arshia Khan

April 2021

Acknowledgements

I would like to acknowledge the following people who have supported me during the process of achieving my master's degree.

My lovely wife Dawn Savela who has spent a lot of time alone while I did research and classwork, my children Kevin Savela and Heather Savela who always keep me on my toes (and Cole Lahti who keeps Heather on her toes), my advisor Dr. Arshia Khan who helped guide me and got me hands-on experience robotics and IoT, Dr. Ted Pedersen who is on my defense committee and shared his joy of NLP and artificial intelligence, Dr. Edward Downs who is also on my defense committee and shared his passion for sensor fusion, my colleagues and friends Todd Juen, Ritu Rayapudi, and John Nelson who helped me with letters of recommendation to get into grad school, my dad Harry Savela (posthumously), mom Paula Savela, brother Kurt Savela, Patricia Rasch, Harold Stevens, Cheryl Edwards, Stacey Stevens, Jeri and Tom Collier, honorary brother David Janzig who had to listen to me describe my research too many times, and June and Keith Watanabe who helped motivate and inspire me to pursue this degree.

Dedication

This paper is dedicated to all people who have discovered the joy and creative potential of software development, and anyone who has ever dreamt of interacting with a robot.

Abstract

Pepper is a humanoid robot created by SoftBank Robotics that was designed and built with the purpose of being used for robot-human interaction. There is an application interface that allows development of custom interactive programs as well as a number of built-in applications that can be extended and used when creating other custom programs for the robot. Among the pre-installed applications are applications that will classify a person's emotion and mood using data from several data points including facial characteristics and vocal pitch and tone.

Due to the Covid-19 pandemic many people have been wearing face masks in both public and private areas. Detecting emotions based on facial recognition and voice tone analysis may not be as accurate when a person is wearing a mask. An alternative method that can be used to classify emotion is to analyze the actual words that are spoken by a person. However, this feature is not currently available on Pepper.

In this study we describe a software solution that will allow Pepper to perform sentiment classification based on spoken words using a neural network. We will describe the testing procedure that was used to interview participants by Pepper and compare the F1 score of each classification method with each other.

Pepper was able to be programmed to use a neural network for emotion classification. A total of 32 participants were interviewed, with the NLP spoken-word analysis classification achieving an averaged F1 score of .2860 score as compared to the built-in software average F1 scores of .2362 from the mood application, .1986 from the vocal tone and pitch application, and .0811 from the facial characteristics application.

Contents

Contents iv

1 Introduction 2

2 Background 5

2.1 Emotion and Mood Classifications 5

2.2 Pepper the Robot 6

2.2.1 Pepper and Emotion Classification 6

2.2.2 Pepper and Speech Recognition 8

2.3 Neural Networks and Sentiment Analysis 8

2.4 Offloading Sentiment Analysis from Pepper 9

2.5 Neural Networks and GPUs 10

2.6 Processing Power 10

2.6.1 Pepper Processing Power 10

2.6.2 NVidia Xavier Processing Power 11

2.7 Related Work 11

2.7.1 Offloading Processing from Robots 11

2.7.2 Emotion Classification via Web Services 12

3 Implementation 14

3.1 Processing Overview 14
3.2 Detecting Sentences with Pepper 15
3.3 Creating a Hub to Process Sentences 17
3.4 Communications 19
3.4.1 Sending Sentences from Pepper 20
3.4.2 Sending Pepper's Built-In Mood, Emotion, and Facial Detection Information to PepperHub 21
3.4.3 PepperHub - Receiving Sentences from Pepper 22
3.4.4 Sending Sentences to the Neural Network and Receiving a Response 24
3.4.5 Receiving and Processing Messages in the Neural Network . . 25
3.5 Data Storage 26
3.6 Creating a Neural Network Language Model 27

4 Data Collection **29**
4.1 Interview Question Collection Technique 29
4.2 Interview Questions 30
4.3 Sentence Dictation Collection Technique 31

5 Results **33**
5.1 Results from Interview Questions 33
5.2 Results from Pre-Classified Scripted Sentences 35

6 Conclusions and Discussion **38**
6.1 Hypothesis Results 38
6.2 Challenges and Improvements 38
6.3 Cloud Based Neural Network for Sentiment Analysis 39

6.4 Ethical Considerations 40
6.5 Future Work 41
6.6 Other Uses of Neural Networks for Robots 41
6.6.1 Sensor Fusion 41
6.6.2 Integration for Multiple Robots 41

References **43**

1 Introduction

Pepper is a humanoid robot created by SoftBank Robotics designed for robot-human interaction and to be a "people-friendly" robot (*Softbank unveils 'human-like' robot Pepper* 2014).

SoftBank provides a programming interface for Pepper to allow development of custom interactive programs and also many applications for developers to use when creating the custom solutions. For example, Pepper has built-in applications that can perform speech recognition, facial detection, and classify a person's emotion and mood. The emotion classification programs use several cues to perform the classification such as how much a person smiles, their facial expressions, the angle of their head, their gaze patterns, the tone of their voice, and their speech semantics. One method that is not currently available with Pepper to classify emotion is the analysis of a person's spoken words.

With the onset of Covid-19, a variant of coronavirus (*Covid Gov* 2021), many people have been wearing face masks either by choice or mandate. When wearing a mask, facial detection is obscured and voice pitch and tone may be muffled or modified, which may reduce Pepper's ability to classify emotion correctly by using the existing methods. Expanding emotional classification by analyzing the spoken words would add to Pepper's abilities and may allow for a more complete picture when performing overall classification.

One way to classify emotions based on spoken words is by using a neural network, a series algorithms and techniques to process data that somewhat mimics the pro-

cesses of the human brain (Goldberg 2017). In the last few years there have been great advances in neural networks due to research by companies such as Google and Facebook. One advanced neural network is BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al. 2019), a transformer based neural network developed by Google engineers which uses an attention (Vaswani et al. 2017) algorithm to determine neural connections. Although it isn't completely understood why it works as well as it does (Kovaleva et al. 2019), (Rogers, Kovaleva, and Rumshisky 2020), it works so well that it is the current method used in many of the Google technologies (Nayak 2019).

There are challenges to using a neural network on Pepper. As with any computer, Pepper has resource limitations. The robot runs an older, subset operating system and has limited storage. The CPU is tasked with all of the on-board functions of Pepper including rolling from point to point, moving the arms and head, sonar sensing of objects, speech interaction and lighting of the eyes and arms. Since Pepper is designed to move freely, it runs on a battery as opposed to being plugged into a power source. The more processing that is performed by the robot, the heavier the load on the computer resources and the more power will be drawn from the robot battery.

In this study we describe a software solution that allows the use of a neural network to perform sentiment classification based on spoken words by offloading the classification processing to a system outside of Pepper that is more suitable for the task. We will describe the neural network that was trained and used, and describe the testing procedure used to capture spoken words and classify the emotion. We will finally compare the F1 score of our neural network classification with each of Pepper's classification methods.

Pepper conducted human interviews within a University of Minnesota's Institutional Review Board (IRB) approved study (STUDY00010490) called "Bold Ideas".

The interviews were conducted using IRB and Covid-19 guidelines. Face masks were worn by all participants in the study.

The research questions for this study are:

1. Can we create a method that will allow Pepper to use an external system to perform emotion classification based on spoken words processed using a neural network?

2. Will the classifications based on spoken words provide more accurate classification than the built-in applications that use facial characteristics and vocal pitch and tone?

The hypotheses for this study are:

- H1: We will be able to create a system to process emotional classification from interactions with Pepper to another system to use a neural network for emotion classification.

- H2: Classification of emotions from the neural network will have a higher F1 score than Pepper's built-in classification based on mood when a participant is wearing a mask.

- H3: Classification of emotions from the neural network will have a higher F1 score than Pepper's built-in classification based on facial features and characteristics when a participant is wearing a mask.

- H4: Classification of emotions from the neural network will have a higher F1 score than Pepper's built-in classification based on vocal pitch and tone when a participant is wearing a mask.

2 Background

2.1 Emotion and Mood Classifications

There is no clear and agreed upon definition of all human emotions. In 2016, psychologist Paul Ekman published a survey he created where he attempted to find any consensus of what could be defined as basic emotions (Ekman 2016). In the survey, of the scientists he surveyed he revealed general agreement of five emotion labels that have been established empirically. The emotions are "anger", "fear", "disgust", "sadness", and "happiness".

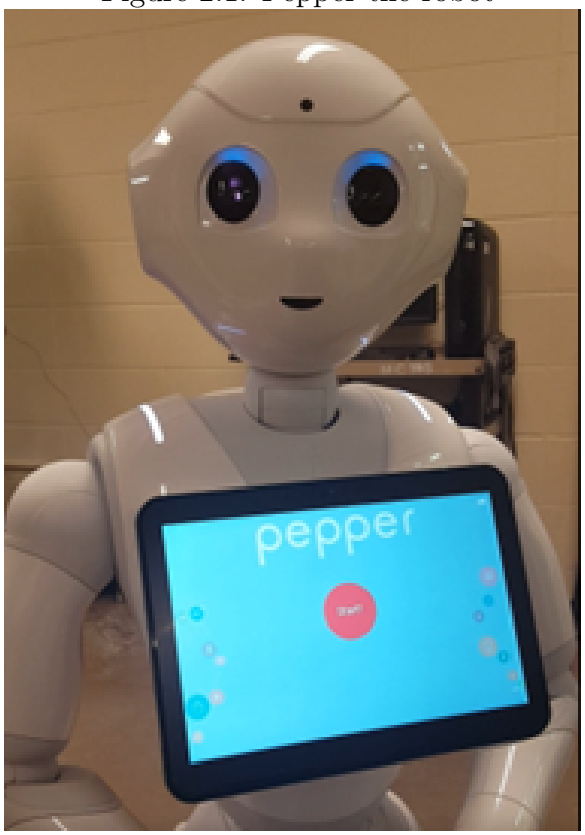
In this study, a number of different detection applications were used to capture emotion. Each classification had a unique set of emotions or moods, but all had an equivalent of "happy" and "angry". Due to this fact, and in order to limit the scope of this effort, we focused on "happy" and "angry" as the emotions we would detect, with any other emotion that was reported classified as "other".

Although related, mood and emotion are not the same. Moods are less intense and last a longer amount of time and emotions are more intense and last a shorter amount of time (Beedie, Terry, and Lane 2005). The NLP application was used to classify emotion, and the mood classifications from Pepper were included in the study for comparison purposes.

2.2 Pepper the Robot

Pepper is a humanoid robot developed and manufactured by SoftBank Robotics (http://softbankrobotics.com). The company's stated mission is to provide robots that are meant to benefit humanity by having a "people-first" design.

Figure 2.1: Pepper the robot



2.2.1 Pepper and Emotion Classification

Pepper has a number of application modules provided by SoftBank to help developers create custom modules. These applications can be accessed via Choregraphe, a client application provided by SoftBank Robotics which provides a graphical interface to a developer to program Pepper and create workflows of interactions (*SoftBank*

Choregraphe 2019). The Choregraphe application provides many built-in code modules that can be modified and extended by a programmer to customize the robot's behavior.

1. Facial Expression: The robot studies the facial expression of the person in front of it and returns on of the following emotions:
   (a) Neutral
   (b) Happy
   (c) Surprised
   (d) Angry
   (e) Sad
2. Voice Emotion: The robot uses the tone and pitch of a person's voice to identify the following emotions:
   (a) Calm
   (b) Anger
   (c) Joy
   (d) Sorrow
   (e) Unknown
3. Mood: This application determines a person's emotional state and mood based on various physical attributes such as how much a person smiles, their facial expressions, the angle of their head, and their gaze patterns.
   (a) Positive
   (b) Neutral

(c) Negative

(d) Unknown

2.2.2 Pepper and Speech Recognition

SoftBank provides software to allow human interaction with the robots, with the design that a developer defines all sentences or sentence fragments that may be spoken to a robot and programs some response to each sentence.

This is not a workable design for capturing sentences for classification since a developer would have no way to anticipate every sentence being spoken by the human. For the sentiment analysis to work, a component must be created for Pepper so sentences can be detected and words extracted from free-flowing conversation.

For this experiment a module was created to capture free-flowing sentences spoken to Pepper. The module is described further in the implementation section.

2.3 Neural Networks and Sentiment Analysis

A neural network is artificial intelligence technology that can be trained for many uses such as language translations, graphical analysis, and emotion classification based on spoken words.

Recently there have been great advancements in neural networks by using an architecture pattern know as "Attention" initially described in the paper "Attention is all you need" (Vaswani et al. 2017).

Language models using Attention have proven to have high accuracy when performing language tasks including translation and sentiment analysis. For example, in a comparison of several modern techniques, the attention-based neural networks performed in the high 90's in accuracy when classifying toxic comments as opposed

to accuracy in the 70's using older neural network technologies (Maslej-Kresnakova et al. 2020).

One advanced attention-based language model is known BERT (Devlin et al. 2019). BERT is a Transformer based neural network from Google that uses an attention algorithm to determine neural connections. The exact reason why it works so well is not fully understood (Kovaleva et al. 2019), (Rogers, Kovaleva, and Rumshisky 2020), but BERT has greatly advanced neural network accuracy in many areas including sentiment analysis.

For this study, the NLP solution was trained with data to detect three emotions

1. Happy
2. Angry
3. Unknown

2.4 Offloading Sentiment Analysis from Pepper

Pepper has a number of limitations for installing or utilizing a neural network. It has limitations from the operating system, disk space, memory, and CPU due to competition with all of the other sub-processes that need to run on the robot. Although Pepper has a GPU processor available, its primary use is for graphics processing and doesn't have the capacity to be programmed for a neural network. It also prohibits installation or is incompatible with multiple software packages.

The neural network processing was offloaded onto the NVidia Xavier machine for the data in this paper. This machine was chosen due to the ability to install the latest neural network applications and libraries, a limitation of the Pepper robot.

The NVidia machine also utilizes a CUDA architecture with GPUs that will allow fast parallel processing for both training and sentence classification.

2.5 Neural Networks and GPUs

Processing neural networks on GPUs is faster than processing on CPUs due to the parallel processing power of their architecture (Nickolls 2007). Graphical Processing Units (GPUs) were originally invented for 3E graphics rendering. Since then they have been advanced with additional programming capabilities. Their design for performing parallel processing have proven to be ideal for use for artificial intelligence processing. GPUs can perform some functions faster by orders of magnitude than CPUs (Lemley, Bazrafkan, and Corcoran 2017).

2.6 Processing Power

2.6.1 Pepper Processing Power

Pepper has a Quad Core Atom E3845 Quad core processor, 4GB of RAM, and 24GB of storage space available to the users. The operating system is NAOqi, a derivative of the Linux operating system.

The robot is designed to move around freely, running on a battery as it's sole power source. All of the on-board functions of Pepper including rolling from point to point, moving the arms and head, sonar sensing of objects, speech interaction and lighting of the eyes and arms drain power from the battery.

Figure 2.2: Nvidia Xavier

2.6.2 NVidia Xavier Processing Power

Xavier has an 8-core ARM v8.2 64-bit CPU and a 512-core Volta GPU with Tensor Cores. The Tensor Cores can be used in learning frameworks like PyTorch (Jeremy Appleyard 2017). The speed and parallel processing of the GPU processors allows for more sophisticated and complex algorithms to run on these systems. An experiment performed on Twitter sentiment using the CUDA kernel on GPUs (Bozkurt et al. 2019) showed significant performance increase using the Nvidia system compared to similar processing on standard systems.

2.7 Related Work

2.7.1 Offloading Processing from Robots

There are currently several different approaches being attempted to offload processing from a robot to a cloud-based solution. A review paper "A Comprehensive Survey of Recent Trends in Cloud Robotics Architectures and Applications" (Saha

and Dasgupta 2018) details several methods of different implementations to offload processing.

An architecture for offloading computational load from robots is described in a high level in the paper "Cloud Robotics: Current Status and Open Issues", (Wan et al. 2016). This architecture focuses primarily on navigation and vision detection, but the overall goal is the same - offload heavy processing from the robot to a cloud based system. In concept, this is the same approach described in this paper for offloading processing tasks.

Finally, the paper "Rapyuta: A Cloud Robotics Platform" (Mohanarajah et al. 2015) introduces a WebSocket based architecture that runs on a Unix system with a unique API Key for communication.

2.7.2 Emotion Classification via Web Services

There are a number of software solutions for classifying emotion based on spoken words.

Google has an experimental web service that will perform text-based Emotional classification via a web service interface called the Natural Language Emotion Classification workshop (https://cloud.google.com/ai-workshop/experiments/language-emotion). An example of a commercial implementation is Receptiviti (receptiviti.com) which uses machine learning to classify emotions using a proprietary corpus of data and provides a REST API endpoint for communication.

Note that cloud-based systems provide challenges of security and privacy. Contrast that with our BERT based system, where all data communications and storage is local to our environment. Another benefit of a local solution is that the training corpus for a local neural network could be tailored to individuals to get a more customized

and personal training and classification results.

3 Implementation

3.1 Processing Overview

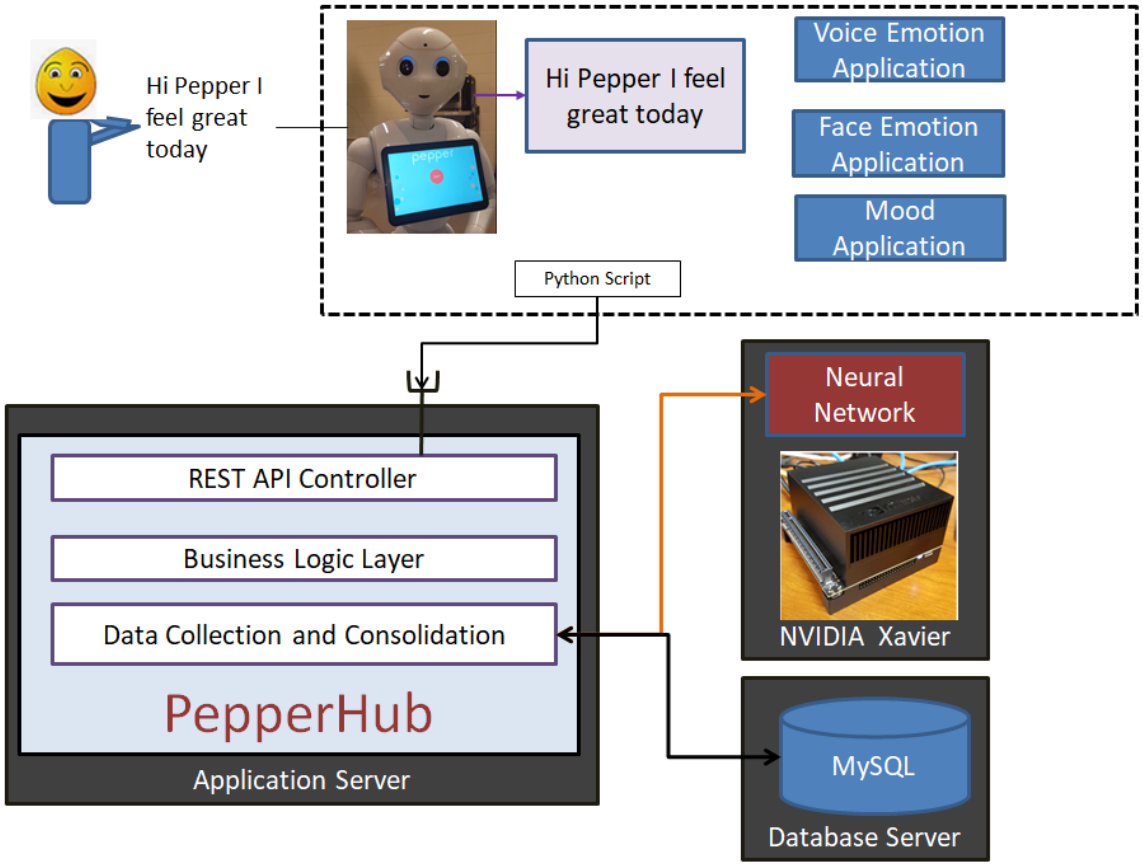
The steps to classify sentiment using a robot are:

- Detect that a sentence has been spoken
- Capture the words that form the spoken sentence
- Clean and normalize the sentence data
- Submit the sentence to a classification system (neural network)
- Retrieve the classification
- Take action on the classification
  - Store the information to a database
  - Respond to the classification (future work)

At a high level, Pepper will capture sentence data using a web-page module created in Choregraphe and store the sentence in local memory. A python module is called to retrieve the sentence data and send it via http to a REST API running on a remote system called "PepperHub".

The PepperHub system will store the sentences in a relational database and in parallel call a BERT-based neural network application asynchronously to retrieve

Figure 3.1: Processing Overview



sentiment classification. When the sentiment classification is returned from the neural network to PepperHub, the classification will be stored in the database and associated to the sentence that was analyzed.

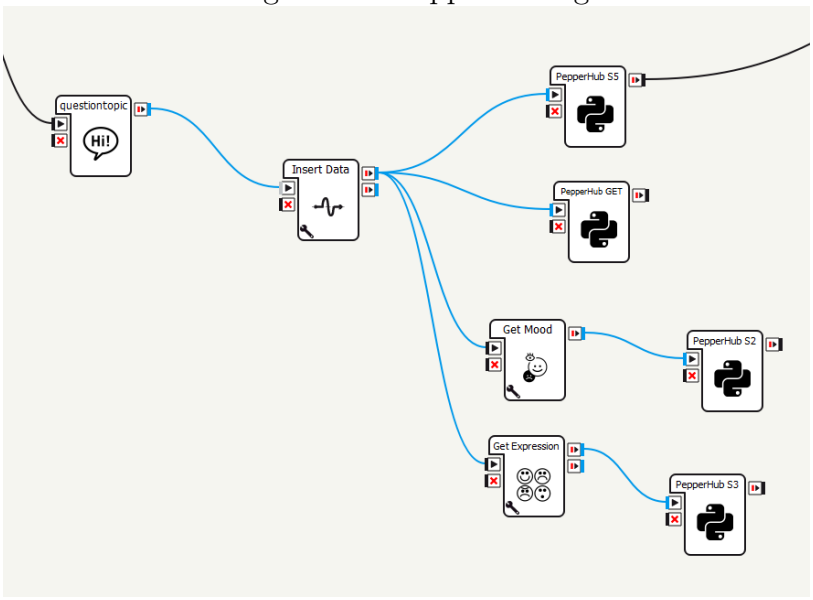
3.2 Detecting Sentences with Pepper

Choregraphe provides an application module called a Dialog (*SoftBank Dialog API* 2021). This module is provided to allow a programmer to define all of the sentence or sentence fragments they may encounter in an interaction. If a defined sentence is detected, the programmer needs to program some response such as further

conversation or some motion or activity.

For this study a new Dialog called "questiontopic" was created to capture sentence data and call the various emotion and mood modules, as well as calling the PepperHub with the sentence data (figure 3.2).

Figure 3.2: Pepper Dialog



The key to this module is that no sentence fragments were provided for the dialog to listen for. Also, no responses are provided (figure 3.3). The outcome of this is that sentences are captured by the Dialog and then sent to the "default" output. The output is stored in memory and a Python module is called to retrieve the data and forward the request the PepperHub REST API.

The original implementation of this algorithm was created by Salisu Wada (Wada 2017), and modified for use in this paper.

Figure 3.3: Question Topic

```
topic: ~questiontopic()
language: enu

u:(_*) $onStopped=$1

# Defining extra concepts out of words or group of words
#concept:(hello) [hello hi hey "good morning" greetings]

# Catching inputs and triggering outputs
#u:(e:onStart) $onStopped=1

# Replying to speech
#u:(~hello) ~hello
```

3.3 Creating a Hub to Process Sentences

An application called PepperHub was created to handle communication for routing and data storage of sentences coming from Pepper and going to and from the sentiment processing nueral network.

The application was written in C# using Microsoft .NET Core 3.1. Microsoft .NET Core allows for the creation of performant applications that can run on multiple platforms including Windows and Linux servers. It is made up of a Controller layer for the REST API set, a business logic layer for any business logic, and a data access layer for storage and other communications. For example, any calls to external sources such as the neural network are performed in the data access layer.

The high-level algorithm for sentence and sentiment classification is

- Receive the sentence from Pepper via an HTTP POST
- Store the sentence data in the database

Figure 3.4: Sentence Processing

```
/// <summary>
/// Save the sentence information, call the sentiment analysis application, and associate the response
/// </summary>
/// <param name="sensorID"></param>The sensor ID that is being captured
/// <param name="hubUserID"></param>The user ID who spoke the sentence
/// <param name="sessionID"></param>The session information for this interaction
/// <param name="sentenceEmotionBO"></param>The emotion data
/// <returns></returns>
2 references | savel008, 99 days ago | 1 author, 1 change
public async Task<string> SaveNLPEmotionFromSentence(int sensorID, int hubUserID, int sessionID, SentenceEmotionBO sentenceEmotionBO) {
    string emotion = "unknown";
    int sentenceID = -1;
    int emotionID = -1;

    // Create an instance of the sentence data business logic object
    SentenceDataBLL sentenceDataBLL = new SentenceDataBLL(_config);
    // Create an instance of the file system data access object
    FileSystemDAO fileSystemDAO = new FileSystemDAO(_config);

    try {
        // Save the sentence to the database
        sentenceID = await sentenceDataBLL.SaveSentence(sensorID, hubUserID, sessionID, -1, sentenceEmotionBO);
    } catch (Exception ex) {
        // Unable to store the data.  Log the error and save the data to a file on the filesystem for later retrieval
        _logger.error($"Unable to save the sentence {sentenceEmotionBO.Sentence} to the database {ex.Message}");
        fileSystemDAO.SaveValueToFile("Sentence", sentenceEmotionBO.Sentence, $"sensorID={sensorID}, hubUserID={hubUserID}" +
            $", sessionID={sessionID}");
        throw new Exception("Unable to save the sentence data.  It was stored to the staging file.");
    }

    // Create an instance of the Emotion DAO
    EmotionDAO emotionDAO = new EmotionDAO(_config);

    // Get the emotion from the Emotion data access object for the given sentence
    emotion = await emotionDAO.GetEmotionFromNLP(sentenceEmotionBO.Sentence);

    try {
        // Save the returned emotion data in the database
        emotionID = await emotionDAO.SaveEmotion(sensorID, hubUserID, sessionID, "NLP", emotion);
    } catch (Exception ex) {
        _logger.error($"Unable to save NLP emotion {emotion} for the sentence {sentenceEmotionBO.Sentence} to the database {ex.Message}");
        fileSystemDAO.SaveValueToFile("Emotion", sentenceEmotionBO.Sentence, $"sensorID={sensorID}, hubUserID={hubUserID}" +
            $"sessionID={sessionID}, type='sentence', emotion='{emotion}'");
        throw new Exception("Unable to save emotion data.  It was stored to the staging file.");
    }

    try {
        // Associate the sentiment with the sentence
        int rowsUpdated = await sentenceDataBLL.UpdateSentenceWithEmotion(sentenceID, emotionID);
    } catch (Exception ex) {
        _logger.error($"Unable to tie the emotion {emotionID} to the sentence {sentenceID}: {ex.Message}");
        throw new Exception("Unable to tie the emotion data to the sentence.");
    }
    return emotion;
}
```

- Call the sentiment processing application (neural network)
- Receive a sentiment response from the sentiment processing application
- Store the sentiment in the database, associating it to the sentence

The business logic method is shown in figure 3.4. The method is called asynchronously, and catches any exception that may be thrown during processing in order to capture any issue that may arise.

The method will attempt to save the sentence to the database. If data can't be stored to the database the sentence is stored to the local file system for later processing.

Once the sentence has been stored, the neural network is called to determine the emotion. The emotion will be saved to the database and tied to the sentence. If this fails, the sentence does not have an associated emotion and a process can be run to associate the sentence to an emotion at a later time.

3.4 Communications

To offload processing from Pepper, for each sentence spoken by the participant the robot will send the following data to PepperHub:

- The sensor being monitored (Pepper)
- The user associated to the data
- Session information (if the dialog session is being tracked)
- The sentence spoken by the person

Communication from Pepper is accomplished by sending HTTP POST messages to the PepperHub REST API through a Python module.

PepperHub will also communicate with the sentiment analysis software. This is accomplished through a TCP/IP socket connection to the Emotion Server process. This communication will only be the sentence itself, and the return value will be the emotion detected by the neural network.

3.4.1 Sending Sentences from Pepper

Choregraphe contains scripting modules that can contain Python code and can be programmed to perform limited Python functions.

In our custom sentence module, the Python code will first verify that a sentence data has been captured. The code then constructs and calls the PepperHub REST API via a HTTP POST call. If the user has been recognized then the user's identity will be sent along with the sentence that was spoken. For this study, participant identities are captured via a number, and the user information is erased after the participant has finished the interview.

Figure 3.5: Pepper Calling PepperHub

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        #put initialization code here
        self.memory = ALProxy("ALMemory")

    def onUnload(self):
        #put clean-up code here
        self.memory = None

    def onInput_onStart(self, keyword):
        # only call if keyword is found
        if keyword:
            self.callXavierWithText(keyword)
        self.onStopped()

    def onInput_onStop(self):
        self.onUnload() #it is recommended to reuse the clean-up as the box is stopped
        self.onStopped() #activate the output of the box

    def callXavierWithText(self, p):
        import urllib2
        url='http://' + PepperServer + '/Emotion/SentenceNLP?sensorID=5'
        try:
            person = self.memory.getData("MyApplication/MyPerson")
        except:
            person = "unknown"

        if person:
            data='{"emotionType": "mood","emotion":"NLP","sentence": "' + p + '","saidBy": "' + person + '"}'
        else:
            data='{"emotionType": "mood","emotion":"NLP","sentence": "' + p + '","saidBy": "unknown"}'

        headers = {'Content-type': 'application/json', 'Accept': '*/*' }
        req = urllib2.Request(url, data, headers)
        response = urllib2.urlopen(req)
```

3.4.2 Sending Pepper's Built-In Mood, Emotion, and Facial Detection Information to PepperHub

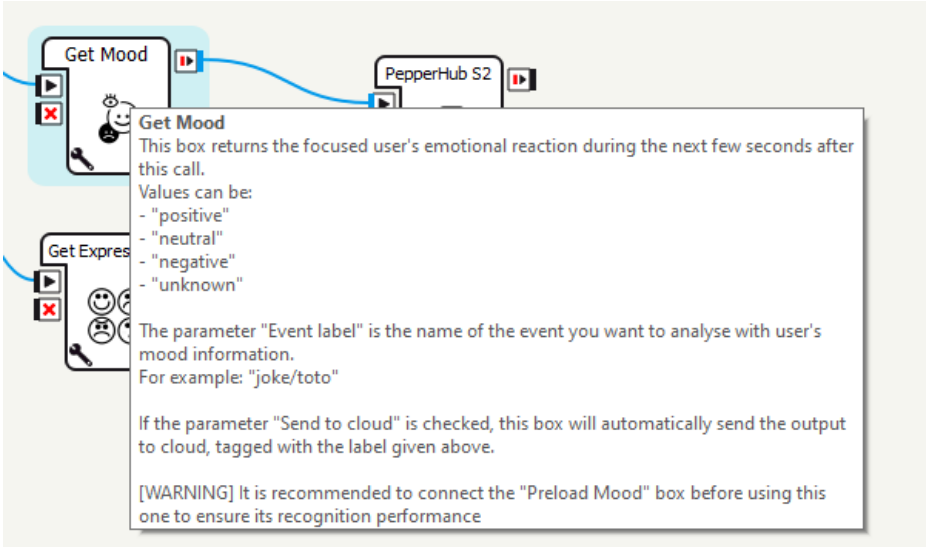
Pepper has modules available in Choregraphe that can be used to detect emotion, mood, and facial detection. Each of these modules will be called to gather data to compare to the classification by the neural network.

Built-In Mood Detection

The "Get Mood" module will detect the mood as positive, negative, neutral, and unknown (Fig 3.6).

The mood that is detected by the user sentence is captured and sent to PepperHub using an HTTP POST method with a sensor ID unique to this module.

Figure 3.6: Built-In Mood Detection

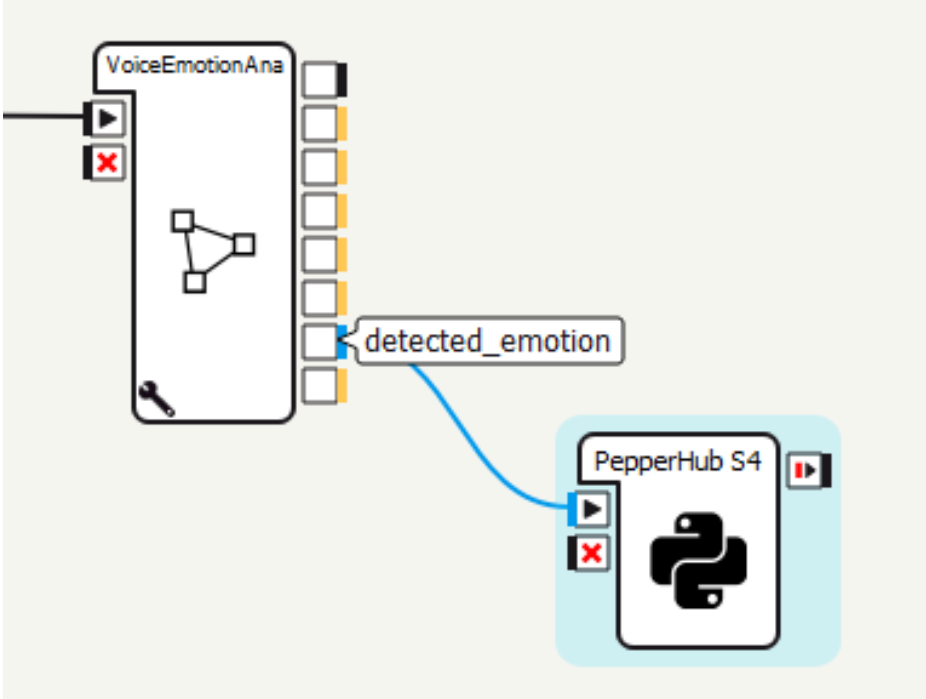


Built-In Emotion Detection

The "Voice Emotion" module will detect a person's emotion of calm, angry, happy, and sad (Fig 3.7).

The emotion that is detected by the user sentence is captured and sent to PepperHub using an HTTP POST method with a sensor ID unique to this module.

Figure 3.7: Built-In Emotion Detection



Built-In Facial Expression

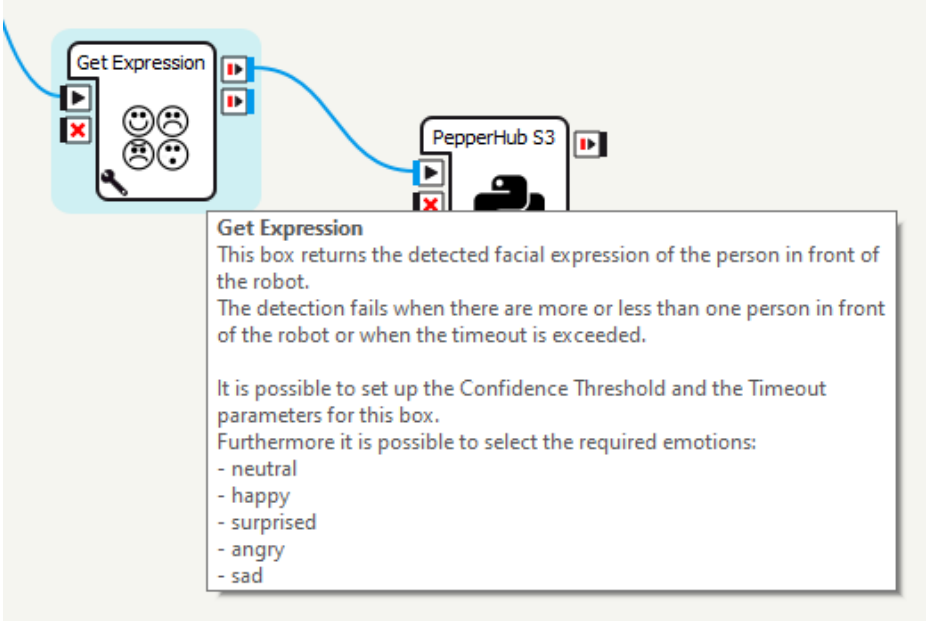
The Built-in "Facial Expression" module captures the expression of the user that Pepper is looking at. The emotion it will output is neutral, happy, surprised, angry, and sad. The confidence threshold is set at .35 for the module (Fig 3.8).

The emotion data that is detected will be captured and sent to PepperHub using an HTTP POST method with a sensor ID unique to this module.

3.4.3 PepperHub - Receiving Sentences from Pepper

The C# module for PepperHub has been created as a standard .NET Core Web Service.

Figure 3.8: Built-In Facial Expression



The Controller class contains a POST method called by Pepper to input the sentences (fig. 3.9). Sensor, user, and session information are all sent in as parameters with the input message sent in as part of the JSON message payload.

Figure 3.9: Get Sentence Controller POST Method

```
[HttpPost("SentenceNLP")]
[Consumes(MediaTypeNames.Application.Json)]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
0 references | savel008, 98 days ago | 1 author, 1 change
public async Task<IActionResult> SaveNLPEmotionAsync(int sensorID, int hubUserID, int sessionID, [FromBody] SentenceEmotionBO sentenceEmotionBO) {
    try {
        EmotionBLL bll = new EmotionBLL(_config);
        string emotion = await bll.SaveNLPEmotionFromSentence(sensorID, hubUserID, sessionID, sentenceEmotionBO);
        return CreatedAtAction("SaveNLPEmotion", emotion);
    } catch (Exception ex) {
        return StatusCode(500, $"Error {ex.Message}");
    }
}
```

3.4.4 Sending Sentences to the Neural Network and Receiving a Response

PepperHub has a Data Access method which calls the Neural Network by opening up a TCP/IP connection to the neural network EmotionServer process.

After the sentence data has been sent, PepperHub waits for a response from the EmotionServer before closing the TCP/IP port. The response is encoded as an ASCII string and returned to the calling method.

Figure 3.10: PepperHub Calling the Neural Network

```
public Task<string> GetEmotionFromNLP(string statement) {
    Socket socket = null;
    if (_disableNLPLookup) {
        _logger.warn("NLP Lookup disabled");
        return Task.FromResult("disabled");
    }
    try {
        // Connect to socket
        byte[] bytes = new byte[3028];
        socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        _logger.debug($"Establishing Connection to {_nlpHost}");
        socket.Connect(_nlpHost, _nlpPort);
        _logger.debug("Connection established");

        _logger.debug($"Socket connected to {socket.RemoteEndPoint.ToString()}");

        // Encode the data string into a byte array.
        byte[] msg = Encoding.ASCII.GetBytes(statement);

        // Send the data through the socket.
        int bytesSent = socket.Send(msg);

        // Receive the response from the remote device.
        int bytesRec = socket.Receive(bytes);

        _logger.debug($"Found emotion {Encoding.ASCII.GetString(bytes, 0, bytesRec)}");
        string returnVal = Encoding.ASCII.GetString(bytes, 0, bytesRec);

        return Task.FromResult(returnVal);
    } catch (Exception ex) {
        _logger.error($"Error connecting to NLP socket {ex.Message}");
        throw;
    } finally {
        try {
            // Release the socket.
            socket.Shutdown(SocketShutdown.Both);
            socket.Close();
        } catch (Exception ex) {
            _logger.warn($"Unable to disconnect from socket {ex.Message}");
        }
    }
}
```

3.4.5 Receiving and Processing Messages in the Neural Network

The BERT-Based neural network is running as a Python service and exposes a TCP/IP Port for communication.

Figure 3.11: Emotion Server TCP/IP Interface

```python
# PepperHub Neural Network
import socketserver
import emotionclass
from collections import defaultdict
from datetime import datetime
import math
import time

PORT = [redacted]
HOST = 'localhost'

# Based on code from
# https://docs.python.org/3.4/library/socketserver.html

class EchoHandler(socketserver.BaseRequestHandler):
    ec = emotionclass.EmotionClass()
    def handle(self):
        # The TCP socket connected to the client
        self.data = self.request.recv(1024).strip()

        # Call the get_emotion method to get the emotion
        emotion = self.ec.get_emotion(self.data.decode("utf-8"))

        # Send the response back to the caller
        self.request.sendall(emotion.encode("utf-8"))

if __name__ == "__main__":
    # Create the server
    print("starting...")
    server = socketserver.TCPServer((HOST, PORT), EchoHandler)

    # Activate the server, keep it running
    server.serve_forever()

```

The Emotion Server Connection (Figure 3.11)

- Opens a TCP/IP port and waits for a connection
- Receives a message string

- Calls the Neural Network method to classify the sentence
- Sends back the classification on the same TCP/IP port

3.5 Data Storage

PepperHub uses a Data Access layer for all database communications. The sentence and emotion data is stored in a MySQL database, an Open Source relational database.

When the sentence data is sent to PepperHub, the information is saved to the sentence_data table (Figure 3.12).

Figure 3.12: Saving Sentence Data

```
const string _insert = "INSERT INTO sentence_data (sensor_id, hub_user_id, session_id, emotion_data_id, said_by, sentence, create_dt)";
2 references | savel008, 98 days ago | 1 author, 1 change
public Task<int> SaveSentence(int sensorID, int hubUserID, int sessionID, int emotionDataID, SentenceBO sentenceBO) { // string sentence) {
    int returnVal = -1;
    string sql = _insert + " values (@SensorID, @HubUserID, @SessionID, @EmotionDataID, @SaidBy, @Sentence, @CreateDt)";
    sql += "; SELECT LAST_INSERT_ID()";
    using (MySqlConnection conn = GetConnection()) {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@SensorID", sensorID);
        cmd.Parameters.AddWithValue("@HubUserID", hubUserID);
        cmd.Parameters.AddWithValue("@SessionID", sessionID);
        cmd.Parameters.AddWithValue("@EmotionDataID", emotionDataID);
        cmd.Parameters.AddWithValue("@SaidBy", sentenceBO.SaidBy ?? "");
        cmd.Parameters.AddWithValue("@Sentence", sentenceBO.Sentence);
        cmd.Parameters.AddWithValue("@CreateDt", DateTime.Now);
        using (MySqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
                returnVal = Convert.ToInt32(reader[0]);
            }
        }
    }
    return Task.FromResult(returnVal);
}
```

The sentiment associated to the sentence is saved in the emotion_data table once it is returned from the EmotionServer (Figure 3.13).

Finally, the sentence and emotion are tied together by adding the Emotion index to the sentence_data table (Figure 3.14).

Figure 3.13: Saving Emotion Data

```
const string _insert = "INSERT INTO emotion_data (sensor_id, hub_user_id, session_id, emotion_type, emotion, create_dt)";
2 references | savel008, 122 days ago | 1 author, 1 change
public Task<int> SaveEmotion(int sensorID, int hubUserID, int sessionID, string emotionType, string emotion) {
    int returnVal = -1;
    string sql = _insert + " values (@SensorID, @HubUserID, @SessionID, @EmotionType, @Emotion, @CreateDt)";
    sql += "; SELECT LAST_INSERT_ID()";
    using (MySqlConnection conn = GetConnection()) {
        conn.Open();
        MySqlCommand cmd = new MySqlCommand(sql, conn);
        cmd.Parameters.AddWithValue("@SensorID", sensorID);
        cmd.Parameters.AddWithValue("@HubUserID", hubUserID);
        cmd.Parameters.AddWithValue("@SessionID", sessionID);
        cmd.Parameters.AddWithValue("@EmotionType", emotionType);
        cmd.Parameters.AddWithValue("@Emotion", emotion);
        cmd.Parameters.AddWithValue("@CreateDt", DateTime.Now);
        using (MySqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
                returnVal = Convert.ToInt32(reader[0]);
            }
        }
    }
    return Task.FromResult(returnVal);
}
```

Figure 3.14: Associate Sentence and Emotion Data

```
const string _updateEmotion = "UPDATE sentence_data set emotion_data_id = @EmotionDataID" +
    " WHERE sentence_data_id = @SentenceDataID";
1 reference | savel008, 122 days ago | 1 author, 1 change
public Task<int> UpdateSentenceWithEmotion(int sentenceID, int emotionDataID) {
    int returnVal = -1;
    string sql = _updateEmotion;
    try {
        using (MySqlConnection conn = GetConnection()) {
            conn.Open();
            MySqlCommand cmd = new MySqlCommand(sql, conn);
            cmd.Parameters.AddWithValue("@SentenceDataID", sentenceID);
            cmd.Parameters.AddWithValue("@EmotionDataID", emotionDataID);
            using (MySqlDataReader reader = cmd.ExecuteReader()) {
                while (reader.Read()) {
                    returnVal = Convert.ToInt32(reader[0]);
                }
            }
        }
        return Task.FromResult(returnVal);
    } catch (Exception ex) {
        _logger.error($"Unable to update sentence and emotion data {ex.Message}");
        throw;
    }
}
```

3.6 Creating a Neural Network Language Model

The Neural Network implementation was based on a model created by Venelin Valkov (Valkov 2019). The application is using a pre-trained BERT model 'bert-base-cased', which can be found at the HuggingFace site (*HuggingFace Transformer Models* 2019) and was trained with a combination of data. The first data source was data intended for the SemEval 2019 Task 3 problem - "EmoContext Contextual Emotion Detection in Text", (Chatterjee et al. 2019) downloaded from the LinkedIn

group named"EmoContext" (*LinkedIn Emotions* 2019). The training data consisted of over 30K lines of text with hand-labeled sentences classified as "happy", "angry", "sad", and "others".

This training data was combined with an additional 218K rows of data from the Unified Emotion DataSets project (Bostan and Klinger 2018). The training set was run for six epochs with a maximum sentence length of 160 characters.

4 Data Collection

Three separate techniques were used to capture data for analysis.

- A series of interview questions meant to evoke happy or angry responses were asked of participants by Pepper
- A series of pre-classified sentences were spoken to Pepper with the participant wearing a mask
- A series of pre-classified sentences were spoken to Pepper with the participant not wearing a mask

4.1 Interview Question Collection Technique

To test the the sentence sentiment classification system a number of participants were interviewed with questions designed to evoke an emotional response when interacting directly with the Pepper robot.

The interview was performed with the participant sitting in a chair facing Pepper, where Pepper would ask questions and listen and record the responses. Any interaction Pepper had with the human were scripted before the interview and sent to Pepper by a human. All questions asked and statements spoken to the person were not programmed into Pepper. Due to Covid guidelines, all participants of the study wore face masks covering their noses and mouths during the interviews.

The participant was asked a number of questions that were classified as questions that should provide a "Happy" or "Angry/Agitated" response.

As the interviewee answered questions, the sentences spoken were captured and sent to PepperHub for processing. In addition, the built-in Pepper emotion, mood, and facial detection applications were invoked and the output data was also sent to PepperHub for data storage.

4.2 Interview Questions

The following list of questions were asked of each participant. They are broken up to first evoke happiness, then agitation, then back to happiness.

Questions to evoke happy responses

- Please think about your favorite animal. Please describe the animal and what it does to make you happy
- I am feeling a little sad and down. Could you say something to help lift my spirits?
- If you had a full day off and could do anything, describe your day.

Questions asked to evoke agitated responses

- Please think of something that someone has done recently that annoys you. Now please act like I am the person who annoyed you and tell me what I did to annoy you.
- Since I annoyed you please tell me to stop. Be very direct and stern with the words you use

- Now imagine you are standing in a line for an hour and I just cut in front of you. Tell me how you feel about it.
- And finally tell me what I should do since I cut in front of you

Final questions to evoke happy responses

- What was your favorite childhood song? What do you think of when you hear it?
- What is your favorite game? Can you describe it?
- What is your favorite dessert? How do you feel when you eat it?

4.3 Sentence Dictation Collection Technique

To get a more reproducible set of data, we generated a list of 15 sentences classified as "happy", and 15 sentences classified as "angry". The classifications were based on human interpretation of the sentences.

The sentences used for the dictation section are in Figure 4.1 and 4.2.

Figure 4.1: Happy Sentences

#	Classification	Statement
1	Positive	I'm feeling happy
2	Positive	What a treat
3	Positive	Isn't this lovely
4	Positive	This is fun
5	Positive	This is fun isn't it
6	Positive	I'm so content right now
7	Positive	Let's keep doing this
8	Positive	This is so fun
9	Positive	I love this place
10	Positive	This is so good
11	Positive	I can do this all day
12	Positive	Let's do this again
13	Positive	Hooray this is great
14	Positive	I'm so happy to see you
15	Positive	I like you

Figure 4.2: Angry Sentences

#	Classification	Statement
1	Angry	I don't like you
2	Angry	Leave me alone
3	Angry	Keep away from me
4	Angry	Don't touch me
5	Angry	Stay away
6	Angry	You're mean
7	Angry	I'm not happy with you
8	Angry	I'm angry right now
9	Angry	Stop it
10	Angry	Why are you so mean
11	Angry	Don't be like that jerk
12	Angry	You're stupid
13	Angry	Keep your distance
14	Angry	This sucks
15	Angry	This is a waste of time

5 Results

5.1 Results from Interview Questions

In total 32 participants were interviewed with data from 28 out of the 32 participants was used for analysis and four eliminated due to failures when capturing the data. Each sentence was classified using the built-in software mood (labeled "mood" in the data results table), emotion (labeled "voice" in the data results table), and facial recognition software (labeled "face" in the data results table) and the PepperHub neural network. These classifications were compared the predicted responses classified as either happy or angry based on the type of question being asked. If a classification was returned as "other", the response was counted an incorrect response.

Figure 5.1: Data Results

Mood	Predicted Class			
Actual Class	happy	angry	other	total
happy	33	15	148	196
angry	12	14	86	112
other	0	0	0	0
	45	29	234	308

Face	Predicted Class			
Actual Class	happy	angry	other	total
happy	18	9	169	196
angry	8	0	104	112
other	0	0	0	0
	26	9	273	308

Voice	Predicted Class			
Actual Class	happy	angry	other	total
happy	11	17	168	196
angry	1	22	89	112
other	0	0	0	0
	12	39	257	308

NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	23	9	164	196
angry	3	27	82	112
other	0	0	0	0
	26	36	246	308

Direct NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	55	14	92	161
angry	1	47	44	92
other	0	0	0	0
	56	61	136	253

Figure 5.2: Precision, Accuracy, and F1

Confusion Matrix - Interview Questions

Mood	Happy	Angry
TP	33	14
TN	100	181
FP	12	15
FN	163	98

Mood	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	196	45	43.18%	0.7333	0.1684	0.2739
angry	112	29	63.31%	0.4828	0.1250	0.1986
Avg			53.25%	0.6080	0.1467	0.2362

Voice	Happy	Angry
TP	11	22
TN	111	179
FP	1	17
FN	185	90

Voice	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	196	12	39.61%	0.9167	0.0561	0.1058
angry	112	39	65.26%	0.5641	0.1964	0.2914
Avg			52.44%	0.7404	0.1263	0.1986

Face	Happy	Angry
TP	18	0
TN	104	187
FP	8	9
FN	178	112

Face	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	196	26	39.61%	0.6923	0.0918	0.1622
angry	112	9	60.71%	0.0000	0.0000	0.0000
Avg			50.16%	0.3462	0.0459	0.0811

NLP	Happy	Angry
TP	23	27
TN	109	187
FP	3	9
FN	173	85

NLP	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	196	26	42.86%	0.8846	0.1173	0.2072
angry	112	36	69.48%	0.7500	0.2411	0.3649
Avg			56.17%	0.8173	0.1792	0.2860

Direct NLP	Happy	Angry
TP	55	47
TN	91	147
FP	1	14
FN	106	45

Direct NLP	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	161	56	57.71%	0.9821	0.3416	0.5069
angry	92	61	76.68%	0.7705	0.5109	0.6144
Avg			67.19%	0.8763	0.4262	0.5606

The NLP solution (labeled "NLP" in the data results table) with sentences captured by the robot had an F1 score of .2860, which was better than all of the other built-in detection methods. The next closest F1 score was .2362 from the mood analysis software, followed by an F1 score of .1986 from the voice analysis software and finally a .0811 score for the facial features software (Figure 5.1).

Since the robot did not capture every word spoken by the participants, a final comparison was done by feeding sentences from transcripts of the spoken sentences to the NLP application. This was done to show how well the NLP could perform if the data capture mechanism was flawless. When using the transcripts, the F1 score rose to .5606. This data is labeled as "Direct NLP" in Figure 5.2.

5.2 Results from Pre-Classified Scripted Sentences

There were two tests done with scripted pre-classified sentences (figures 4.1 and 4.2). One test was done when wearing a mask, the other was done without wearing a mask.

Figure 5.3: Masked Interview Data Results

Mood	Predicted Class			
Actual Class	happy	angry	other	total
happy	0	0	15	15
angry	0	0	15	15
other	0	0	0	0
	0	0	30	30

Face	Predicted Class			
Actual Class	happy	angry	other	total
happy	0	0	15	15
angry	0	0	15	15
other	0	0	0	0
	0	0	30	30

Voice	Predicted Class			
Actual Class	happy	angry	other	total
happy	0	1	14	15
angry	0	0	15	15
other	0	0	0	0
	0	1	29	30

NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	6	0	9	15
angry	0	9	6	15
other	0	0	0	0
	6	9	15	30

Direct NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	6	0	9	15
angry	0	10	5	15
other	0	0	0	0
	6	10	14	30

When wearing a mask, the mood application only returned "unknown" and the facial response application did not return any results. The voice response mis-classified one "happy" classification as "angry", but otherwise registered as "sorrow". The sentences with human classification were not run against the NLP program before running the tests to make it a true test for classification. Even so the NLP solution did a much better job achieving an F1 score of .6607. There was one classification made differently using the NLP software through Pepper as opposed to the sentence running directly to the NLP software, so the Direct NLP classification was slightly higher at .6857 (figures 5.3 and 5.4).

When unmasked using the same pre-classified sentences, all of the built-in Pepper

Figure 5.4: Precision, Accuracy, and F1 - Masked Interview

Confusion Matrix - Scripted Masked Statements

Mood	Happy	Angry
TP	0	0
TN	15	15
FP	0	0
FN	15	15

Mood Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	0	50.00%	0.0000	0.0000	0.0000
angry	15	0	50.00%	0.0000	0.0000	0.0000
Avg			50.00%	0.0000	0.0000	0.0000

Voice	Happy	Angry
TP	0	0
TN	15	14
FP	0	1
FN	15	15

Voice Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	0	50.00%	0.0000	0.0000	0.0000
angry	15	1	46.67%	0.0000	0.0000	0.0000
Avg			48.33%	0.0000	0.0000	0.0000

Face	Happy	Angry
TP	0	0
TN	15	15
FP	0	0
FN	15	15

Face Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	0	50.00%	0.0000	0.0000	0.0000
angry	15	0	50.00%	0.0000	0.0000	0.0000
Avg			50.00%	0.0000	0.0000	0.0000

NLP	Happy	Angry
TP	6	9
TN	15	15
FP	0	0
FN	9	6

NLP Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	6	70.00%	1.0000	0.4000	0.5714
angry	15	9	80.00%	1.0000	0.6000	0.7500
Avg			75.00%	1.0000	0.5000	0.6607

Direct NLP	Happy	Angry
TP	6	10
TN	15	15
FP	0	0
FN	9	5

Direct NLP Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	6	70.00%	1.0000	0.4000	0.5714
angry	15	10	83.33%	1.0000	0.6667	0.8000
Avg			76.67%	1.0000	0.5333	0.6857

applications performed better than they had with a masked subject except for the facial recognition application. Without a mask, the mood application achieved the highest F1 score with .7890, followed by the NLP F1 score of .6857 and finally the voice analysis application at .0588 and the facial features coming in a 0.1000 (figures 5.5 and 5.6).

Figure 5.5: Unmasked Interview Data Results

Mood	Predicted Class			
Actual Class	happy	angry	other	total
happy	8	4	3	15
angry	0	15	0	15
other	0	0	0	0
	8	19	3	30

Face	Predicted Class			
Actual Class	happy	angry	other	total
happy	0	3	12	15
angry	0	2	13	15
other	0	0	0	0
	0	5	25	30

Voice	Predicted Class			
Actual Class	happy	angry	other	total
happy	1	0	14	15
angry	1	0	14	15
other	0	0	0	0
	2	0	28	30

NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	6	0	9	15
angry	0	10	5	15
other	0	0	0	0
	6	10	14	30

Direct NLP	Predicted Class			
Actual Class	happy	angry	other	total
happy	6	0	9	15
angry	0	10	5	15
other	0	0	0	0
	6	10	14	30

Figure 5.6: Precision, Accuracy, and F1 - Unasked Interview

Confusion Matrix - Scripted Unmasked Statements

Mood	Happy	Angry
TP	8	15
TN	15	11
FP	0	4
FN	7	0

Mood						
Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	8	76.67%	1.0000	0.5333	0.6957
angry	15	19	86.67%	0.7895	1.0000	0.8824
Avg			81.67%	0.8947	0.7667	0.7890

Voice	Happy	Angry
TP	1	0
TN	14	15
FP	1	0
FN	14	15

Voice						
Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	2	50.00%	0.5000	0.0667	0.1176
angry	15	0	50.00%	0.0000	0.0000	0.0000
Avg			50.00%	0.2500	0.0333	0.0588

Face	Happy	Angry
TP	0	2
TN	15	12
FP	0	3
FN	15	13

Face						
Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	0	50.00%	0.0000	0.0000	0.0000
angry	15	5	46.67%	0.4000	0.1333	0.2000
Avg			48.33%	0.2000	0.0667	0.1000

NLP	Happy	Angry
TP	6	10
TN	15	15
FP	0	0
FN	9	5

NLP						
Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	6	70.00%	1.0000	0.4000	0.5714
angry	15	10	83.33%	1.0000	0.6667	0.8000
Avg			76.67%	1.0000	0.5333	0.6857

Direct NLP	Happy	Angry
TP	6	10
TN	15	15
FP	0	0
FN	9	5

Direct NLP						
Class	n (truth)	n (class)	accuracy	precision	recall	F1 Score
happy	15	6	70.00%	1.0000	0.4000	0.5714
angry	15	10	83.33%	1.0000	0.6667	0.8000
Avg			76.67%	1.0000	0.5333	0.6857

6 Conclusions and Discussion

The architecture that was designed and created for this paper successfully achieved the goal of offloading sentiment classification from Pepper to an external system. The BERT-based NLP solution achieved a higher F1 score than all of the classification applications provided with Pepper.

6.1 Hypothesis Results

Offloading sentiment analysis using a neural network has proven to be feasible (H1). We were able to send sentence information to a neural network from Pepper and retrieve sentiment classification with no processing delays.

The neural network was able to perform sentiment classification with greater F1 score than all of Pepper's built-in classification systems of mood (H2), facial features (H3) and vocal pitch and tone (H4).

The classification worked as expected and augmented the emotion classification capability of Pepper. The solution was performant and can be easily expanded for future use.

6.2 Challenges and Improvements

Although offloading the sentiment classification was successful, there are some areas of the study that have raised opportunities for improvement.

Capturing spoken sentences was a challenge for the robot. This may be due to people wearing masks, where voices were softer and somewhat muffled. Being able to properly capture the spoken words isn't a problem for Pepper's built-in emotion classification systems since they don't analyze the words, but for the neural network the hit-or-miss nature of the word capture reduced the overall accuracy of the sentiment classifications.

Another challenge was the robot's nature of being distracted when attempting to carry on a conversation. When the robot interacts with a person the conversation is interrupt driven. The robot needs an enhanced communication application written so it can filter out some of the environment noise it encounters and focus on the person it is have a conversation with.

6.3 Cloud Based Neural Network for Sentiment Analysis

Google is running a beta test on a cloud-based sentiment analysis system. The site accepts a sentence as input, analyzes it for sentiment classification, and returns the result. This is site that was used for data verification for this paper.

Using the Google detection method would have its advantages. The company has vast resources at its disposal in hardware, software talent, and training corpus. The BERT-based neural network used in this paper was developed by Google engineers.

That said, the use of a neural network that is locally trained and deployed also has several advantages. The data is contained to the local network, so confidentiality can be more easily maintained. Depending on the use, the training data could be tailored to an individual. Finally, the system currently relies on a LAN as opposed

to a WAN, which means that the communication might be less prone to availability issues and more likely to have faster speed and fewer latency issues.

6.4 Ethical Considerations

This project touches on some ethical issues around data security, confidentiality, and the appropriate use of the information being gathered.

When storing any personal information, there must be a determination which data should be stored if any. If data is stored, a privacy question can be raised if the data should be tied to the individuals providing the data.

The solution presented in this paper stores the question and answer data but with no personal information tying the data to who was saying the sentences. Also, the questions being asked in the interview were created to avoid the use of any personal information in the answers.

Since this project was built with internal processing and data storage, exposure of the data was minimal compared to if we had stored the information on a cloud based system and used an external sentiment classification system such as the solution provided by Google.

The PepperHub application is designed so that it could take action after determining the emotional state of the person it is talking to. Taking some sort of action on the data, even with good intentions, raises another ethical question. If we determine that the person interacting with Pepper is becoming angry or agitated, we could program pepper to attempt to calm the person down or notify a nurse if it's in a medical facility. This could be considered a violation of patient privacy. It could also cause more harm than good if, instead of lessening the emotions, the intervention techniques escalates the emotion.

6.5 Future Work

6.6 Other Uses of Neural Networks for Robots

There are other uses for a neural network that could possibly further enhance human-robot interaction. For example, using a neural network in a chat-based program for the robot could greatly enhance the conversation ability of the robot or be used to translate languages. A neural network could also be used to enhance the visual recognition abilities of the robot, such as identifying hand gestures for sign language.

6.6.1 Sensor Fusion

The technique to offload sentiment analysis by itself is a good first step, but many other metrics can also be gathered by PepperHub and placed into a centralized database.

As part of the Bold Ideas grant, there are multiple sensors sending data to the PepperHub application including pulse, Electrodermal activity (EDA), and Blood Pressure. With all of these sensor metrics available including the sentiment detection, the PepperHub application could be programmed to combine the information to detect emotions and mood. It could also use this information when interacting with a person to better adjust to how the person is reacting and perhaps deescalate or notify a medical professional about a potential emotional episode.

6.6.2 Integration for Multiple Robots

Since PepperHub is a distributed solution multiple robots could use this platform concurrently. PepperHub could coordinate multiple robots in one location, determin-

ing the emotional classification of each individual as well as analyzing the data to determine an comprehensive classification of the group.

If multiple connections start to overwhelm the server resources, PepperHub can be scaled horizontally to handle the increased load. This is something that is not currently possible for the Pepper robot.

Finally, any robot that is able to perform and capture speech recognition and send the information to a web service could use the PepperHub system. This opens up the possibility of robots with much less capabilities than Pepper to be used to detect human emotions.

References

Beedie, Christopher, Peter Terry, and Andrew Lane (2005). "Distinctions between emotion and mood". In: *Cognition and Emotion* 19.6, pp. 847–878. DOI: 10.1080/02699930541000057. eprint: https://doi.org/10.1080/02699930541000057. URL: https://doi.org/10.1080/02699930541000057 (cit. on p. 5).

Bostan, Laura Ana Maria and Roman Klinger (2018). "An Analysis of Annotated Corpora for Emotion Classification in Text". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, pp. 2104–2119. URL: http://aclweb.org/anthology/C18-1179 (cit. on p. 28).

Bozkurt, Ferhat et al. (2019). "High Performance Twitter Sentiment Analysis Using CUDA Based Distance Kernel on GPUs". eng. In: *Tehnički Vjesnik* 26.5, pp. 1218–1227. ISSN: 1330-3651. URL: https://doaj.org/article/e4cfe5e51bfb45998d97b6784ba66f4
(cit. on p. 11).

Chatterjee, Ankush et al. (June 2019). "SemEval-2019 Task 3: EmoContext Contextual Emotion Detection in Text". In: *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, pp. 39–48. DOI: 10.18653/v1/S19-2005. URL: https://www.aclweb.org/anthology/S19-2005 (cit. on p. 27).

Covid Gov (2021). URL: https://coronavirus.dc.gov/page/what-covid-19 (cit. on p. 2).

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL] (cit. on pp. 3, 9).

Ekman, Paul (2016). "What Scientists Who Study Emotion Agree About". eng. In: *Perspectives on psychological science* 11.1, pp. 31–34. ISSN: 1745-6916 (cit. on p. 5).

Goldberg, Yoav (2017). *Neural network methods for natural language processing*. eng. San Rafael, California]: Morgan and Claypool. ISBN: 9781627052986 (cit. on p. 3).

HuggingFace Transformer Models (2019). URL: https://huggingface.co/transformers/model%20doc/bert.html (cit. on p. 27).

Jeremy Appleyard, Scott Yokim (2017). *CUDA Programming*. URL: https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/ (cit. on p. 11).

Kovaleva, Olga et al. (Nov. 2019). "Revealing the Dark Secrets of BERT". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, pp. 4365–4374. URL: https://www.aclweb.org/anthology/D19-1445 (cit. on pp. 3, 9).

Lemley, J., S. Bazrafkan, and P. Corcoran (2017). "Deep Learning for Consumer Devices and Services: Pushing the limits for machine learning, artificial intelligence, and computer vision." In: *IEEE Consumer Electronics Magazine* 6.2, pp. 48–56. DOI: 10.1109/MCE.2016.2640698 (cit. on p. 10).

LinkedIn Emotions (2019). URL: https://www.linkedin.com/groups/12133338/ (cit. on p. 28).

Maslej-Kresnakova, Viera et al. (2020). "Comparison of Deep Learning Models and Various Text Pre-Processing Techniques for the Toxic Comments Classification". eng. In: *Applied sciences* 10.23, p. 8631. ISSN: 2076-3417 (cit. on p. 9).

Mohanarajah, Gajamohan et al. (2015). "Rapyuta: A Cloud Robotics Platform". eng. In: *IEEE transactions on automation science and engineering* 12.2, pp. 481–493. ISSN: 1545-5955 (cit. on p. 12).

Nayak, Pandu (2019). *Understanding Bert*. URL: https://blog.google/products/search/search-language-understanding-bert (cit. on p. 3).

Nickolls, John (2007). "GPU parallel computing architecture and CUDA programming model". eng. In: *2007 IEEE Hot Chips 19 Symposium (HCS)*. IEEE, pp. 1–12. ISBN: 1467388696 (cit. on p. 10).

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky (2020). *A Primer in BERTology: What we know about how BERT works*. arXiv: 2002.12327 [cs.CL] (cit. on pp. 3, 9).

Saha, Olimpiya and Prithviraj Dasgupta (2018). "A comprehensive survey of recent trends in cloud robotics architectures and applications". eng. In: *Robotics (Basel)* 7.3, p. 47. ISSN: 2218-6581 (cit. on p. 11).

SoftBank Choregraphe (2019). URL: https://developer.softbankrobotics.com/nao6/naoqi-developer-guide/choregraphe-suite (cit. on p. 6).

SoftBank Dialog API (2021). URL: https://developer.softbankrobotics.com/pepper-naoqi-25/naoqi-developer-guide/getting-started/hello-world/hello-world-3-using-dialog-topic (cit. on p. 15).

Softbank unveils 'human-like' robot Pepper (2014). URL: https://www.bbc.com/news/technology-27709828 (cit. on p. 2).

Valkov, Venelin (2019). URL: https://www.curiousily.com/posts/sentimentanalysis-with-bert-and-hugging-face-usingpytorch-and-python (cit. on p. 27).

Vaswani, Ashish et al. (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf (cit. on pp. 3, 8).

Wada, Salisu (2017). *DetectingSpeech*. URL: https://medium.com/@salisuwy/displaying-users-speech-on-pepper-humanoid-robot-tablet-3556580e4d01 (cit. on p. 16).

Wan, J. et al. (2016). "Cloud robotics: Current status and open issues". In: *IEEE Access* 4, pp. 2797–2807. DOI: 10.1109/ACCESS.2016.2574979 (cit. on p. 12).