Summer 2024

Context Detection With Word Embedding and Emotionally Relevant Keyword Search for Smart Home Environment

Brent Anderson
Missouri State University, Anderson97@live.missouristate.edu

CONTEXT DETECTION WITH WORD EMBEDDING AND EMOTIONALLY RELEVANT KEYWORD SEARCH FOR SMART HOME ENVIRONMENT

A Master's Thesis

Presented to

The Graduate College of

Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science, Computer Science

By

Brent Anderson

August 2024

CONTEXT DETECTION WITH WORD EMBEDDING AND EMOTIONALLY RELEVANT KEYWORD SEARCH FOR SMART HOME ENVIRONMENT

Computer Science

Missouri State University, August 2024

Master of Science

Brent Anderson

ABSTRACT

Voice-enabled virtual assistants have gained widespread popularity and are increasingly common in smart homes. To enhance customization and personalization in user experiences with these assistants, implementing a context detection feature is beneficial. This feature enables the virtual assistant to gather more information from the audio data of short voice conversations with users, helping it maintain awareness of the conversation and respond more aptly. In this thesis, I propose a novel context detection approach for virtual assistants in smart homes, named WERKS, which leverages user emotions. WERKS stands for word embedding with emotionally relevant keyword search. The WERKS approach incorporates emotion detection, keyword search, and word embedding from voice commands and short conversations to achieve effective context detection. This method comprises emotion detection, basic context detection, word embedding with emotionally relevant keyword search, and ensemble defined context classification layers. Evaluation of the WERKS approach on datasets has demonstrated that it can significantly improve context detection accuracy.

KEYWORDS: audio, bigram, emotion detection, neural network, noun phrase, synset

CONTEXT DETECTION WITH WORD EMBEDDING AND EMOTIONALLY RELEVANT KEYWORD SEARCH FOR SMART HOME ENVIRONMENT

By

Brent Anderson

A Master's Thesis
Submitted to the Graduate College
Of Missouri State University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science, Computer Science

August 2024

Approved:

Razib Iqbal, Ph.D., Thesis Committee Chair

Yassine Belkhouche, Ph.D., Committee Member

Lloyd Smith, Ph.D., Committee Member

Julie Masterson, Ph.D., Dean of the Graduate College

ACKNOWLEDGEMENTS

I dedicate this thesis to my wife, Sarah Anderson.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

In this introduction, the research motivation, problem, questions, and contribution will be addressed to describe the problems this thesis will explore and pursue solutions for. The research motivation will describe the need for including as much feature data in context detection as possible. The research problem will illustrate poor user experience with virtual assistants and propose a potential solution through increased context detection accuracy. The questions this thesis will pursue answers to will then be discussed, along with my existing contribution to this topic. Finally, an outline will be provided to describe the remainder of the sections this thesis includes.

Research Motivation

The Internet of Things (IoT) field has been a growing field for years now [1]. IoT is also seeing a continued increase in the types and number of devices [2]. One area of research within the IoT field focuses on the development of virtual assistants, particularly in terms of their integration with IoT devices, their application in home automation systems, and their contribution to enhancing user experiences. The Multimedia Internet of Things (M-IoT) is the interconnection of multimedia devices through the internet and communication networks. Multimedia devices, specifically in the smart home can include things such as security cameras, lights, and thermostats [3]. Virtual assistants are technologies powered by artificial intelligence, exemplified by services such as amazon's Alexa, google assistant, and apple's Siri. In human to device interaction, an end user (person that uses a given system and or product) in a smart home can interact with these services, often through speech by issuing a command to the device. For device-to-device interaction, these assistants are integrated into an end user's

home environment, such as a house or apartment, so they can interact with smart appliances, entertainment and security systems, and a variety of household systems like HVAC and lighting.

Research has also been conducted in the field of IoT regarding the emotional state and or states over time for a user within a smart home, often referred to as emotion detection [4]. Utilizing the user's emotion, such as ‘happy’ or ‘sad,’ in a smart home environment can lead to multiple inferences being made to assist the user. These inferences can include actions like changing the lighting in a room or adjusting a thermostat, resulting in suggestions or actions the virtual assistant can take if it is connected to other devices. The things a virtual assistant can act on, like the lighting and thermostat adjustments just described, can be referred to as smart home actions. Now, to obtain these emotional states of the user such as ‘happy’ or ‘sad’, the process of emotion detection must first occur. For a virtual assistant, emotion detection typically consists of extracting feature data such as Mel spectrogram (MEL) and Mel-frequency cepstral coefficients (MFCC) from the user’s audio data to be classified as one of many pre-defined emotions.

Another avenue of research in the IoT field is context detection [5]. Context can be defined as all the necessary information required to accurately assess a setting of a statement, event, or idea. A few approaches to context detection utilize sensors, audio data, and sometimes even text to determine context. Context detection has a wide range of these different approaches since 1998 [6]. Obtaining the context of a setting within a smart home environment can also be utilized along with the emotion of the user to provide benefit to the user experience. For example, given a context of ‘music’ used in conjunction with a sensor could inform a virtual assistant to play music at the user's current location. This same smart

home action could be obtained from the emotional state of the user. If the user was currently ‘sad’, then the virtual assistant might suggest that it play some music for the user to assist them in feeling ‘happy’.

To enhance emotion detection systems, the integration of context detection has recently been investigated. This approach combines the context of user interactions with data from various sources, including audio, text, and imagery [7]. Despite this initiative, a gap remains in how these systems incorporate emotional nuances within the context detection itself. Therefore, by integrating emotion recognition directly into context analysis, such systems could significantly improve user experience, offering more intuitive and responsive interactions [8].

In pursuit of addressing the gap left by the approaches that investigate the integration of context detection into emotion detection systems, I have discovered a new insight which has been derived from these approaches. This insight has provided a motivation to approach these systems in reverse, such that the emotion detection methods provide features for the purpose of context detection.

The purpose of my approach to context detection in which emotion detection is utilized is to provide as much detail as possible in the context detection process. This detail is in the form of data for my approach to consume such as data derived from audio and text. Through the incorporation of emotional feature data into the contextual feature data, newly formed contexts can be found to assist in providing a better user experience. This discovery has the potential to improve the smart home environment through user interaction with virtual assistants by providing actions better suited to the needs of the user.

Research Problem

For the virtual assistant in a smart home, the importance of providing the user with the best interactions and smart home actions cannot be overstated, as smart homes have been perceived as having a limited appeal to users along with a perceived failure to meet the needs of the user [9]. If a virtual assistant in a smart home environment does not consistently provide the user with positive interaction that directly results in the correct smart home actions, then the user may not enjoy interacting with the virtual assistant and may therefore also be less likely to be influenced [10]. It is also possible to include context detection as part of the smart home action prediction processes to improve user experience. To achieve the goal of providing the user with the most consistent virtual assistant interactions possible, I propose a system to obtain the highest possible prediction accuracy for these smart home actions via the inclusion of user emotion in the context detection process.

In pursuit of building a system to obtain the highest possible prediction accuracy, I have searched for existing context detection systems that incorporate emotion for the purpose of context detection. In my search, research shows some systems do exist such as in [7] and [11] to incorporate context for the purpose of emotion detection, but the same cannot be said about incorporating emotion into systems for the purpose of context detection. To illustrate the lack of research to incorporate emotion into context detection, it should be stated that throughout the research performed for this thesis, my proposed approach for the inclusion of emotion for the purpose of context detection has yet to be investigated; therefore, based on the literature review in the context detection field, this approach is novel. To obtain a quantifiable result, I pursue an increase of the context prediction accuracy for new and existing

context detection systems. For the smart home, this goal ultimately seeks to improve the user experience by assisting with the accurate prediction of smart home actions during user interactions.

Research Questions

This thesis primarily investigates the following research questions:

1. What quantifiable benefits can the incorporation of user emotion and emotion detection features offer to the field of context detection research, and how can these contributions be systematically measured and evaluated?
2. Can feature data extracted during emotion detection be effectively repurposed for context detection purposes?
3. Is it feasible to directly integrate user emotion into the context detection process, thereby enhancing the accuracy and effectiveness of contextual analyses?

Based on the above research questions and preliminary investigations, I hypothesized that extracting natural language feature data from user audio inputs and utilizing this data to generate unique emotionally relevant context feature data will significantly enhance the accuracy of classifying various predefined contexts within a context detection system for smart homes.

Research Contribution

To prove my hypothesis and address the problem of a better user experience, I propose a novel word embedding with emotionally relevant keyword search (WERKS) approach to improve context detection accuracy through the inclusion of emotion. The WERKS approach to context detection incorporates a combination of emotion detection, keyword search, and word

embedding for context detection from voice commands and short conversations with virtual assistants. The addition of the context detection feature in voice conversations with virtual assistants can offer a more personalized experience in smart homes by maintaining awareness of the ongoing conversation and responding appropriately. The WERKS approach will first detect the emotional state of the user as either angry, fearful, happy, normal, or sad from short voice interactions, and then parse this same audio data into a noun phrase bigram context, which serves as the initial context detection method. The detected emotion and context are then passed to the word embedding with an emotionally relevant keyword search layer to obtain the final context. The tree-based pipeline optimization tool (TPOT) classifier was applied over the publicly available RAVDESS data set and a custom data set to obtain the experimental results, which demonstrated a 15 and 12 percent increase in prediction accuracy of the defined contexts based on the smart home user's verbal interactions and emotional relevancy.

This thesis work has resulted in the following peer-reviewed publication:

B. Anderson and R. Iqbal, "Word Embedding with Emotionally Relevant Keyword Search for Context Detection from Smart Home Voice Commands," in IEEE 21st Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2024, pp. 594-595, doi: 10.1109/CCNC51664.2024.10454678.

Thesis Outline

This thesis is organized as follows: In the Literature Review section, the reader will become familiarized with works that are related to this one and other similar topics of research such as neural networks and topic modeling. The Proposed Approach section will provide a detailed overview and individual component examination of the WERKS approach and overall

architecture within the system. I then describe the supporting features of the system in the Implementation section. The results and process to obtain these results will be presented in the Experimental Setup and Evaluation section. Finally, the Conclusion section will offer an overall summary of this thesis, along with my concluding remarks.

LITERATURE REVIEW

In this section, existing literature will be reviewed to provide insight into the Proposed Approach section of this thesis. The topics that will be explored will include context detection, text classification via a neural network, and finally, topic modeling short corpus.

Context Detection

Context detection has been an active area of research in recent decades [6]. In contrast, context detection in the smart home environment could be considered a newer research field. This research field involves the pursuit of technologies that make a home "smart" through the detection and response of various contextual factors. Various methodologies have been proposed for context detection in general and context detection in the smart home environment. An approach for the latter is presented in [5] where fuzzy logic and sensors are utilized for the data gathering medium to obtain contextual data. Another sensor-centric approach to context detection is presented in [12] in which malicious behavior in the smart home is detected for context-aware security.

As for context detection outside of the smart home environment, there have also been many proposed approaches to this topic. One such proposal can be found in [13] where an approach for context detection is proposed in which two contexts are obtained from audio data in movies as either 'Gunplay' or 'Car Racing'. Another approach presented in [14] shows how the authors utilize a hierarchical approach in which statistical characteristics for given audio events are modeled in a time series for context detection. Audio data is also the main medium in [15] in which contexts were timed and had respective accuracies calculated allowing for

comparison between them. Thus, audio data appears to be a valid data medium for context detection.

Context detection is also applicable to text such that larger prose is condensed and processed to text representative of the context for the given prose. This process is also applicable to shorter texts. One such context detection methodology regarding text is the bigram, in which text is processed into a two-word phrase representative of the respective context of a given prose. Bigrams have been used in other systems as well for purposes such as word error detection and correction in [16] and sarcasm detection in [17]

Finally, another research topic of note regarding the smart home environment that has been implemented in multiple systems is that of emotion recognition. Emotion recognition can also be combined with other technologies to create a complete system, such as context detection. Reference [18] provides a system that implements emotion within the smart home environment which combines data processing, computer vision, learning, and environmental sensing into a self-adapting system. Another system that has been proposed combining multiple systems is described in [4] where dialogue systems and ubiquitous computing are utilized to provide a robust approach for incorporating emotion within the smart home environment. Thus, when emotion is a part of the smart home environment it serves a vital role to bridge the gap in interaction between the smart home and the user for the context detection process.

Text Classification via Neural Network

An artificial neural network (ANN) is an algorithmic model of a collection of nodes/neurons in multiple layers to resemble the neuron structure of a biological brain. Simple

artificial neural networks can solve linear problems, whereas if more layers are added to the ANN then it becomes deep and allows for solving of non-linear problems. This is referred to as a deep neural network (DNN); it should be noted there are additional layers that exist within the hidden layers of the network. DNN's require a lot of computational resources when they get large enough. Thus, even though ANN's have been around since 1944, complex DNN's have only been gaining more and more widespread use in recent years as graphical processing units have advanced to match the computational requirements [19]. A DNN is capable of consuming data for the purpose of assigning this given input to a class. This process is known as classification and has many applications such as advertisement, security, and medicine.

One type of DNN is a convolutional neural network (CNN). Reference [20] suggests general machine learning models are not quite fit for the purpose of text classification with large amounts of data. Instead, a CNN is suggested for this purpose. It is also suggested that in addition to handling larger amounts of data better than machine learning models, deep learning models such as the CNN also provide better performance. Before being passed to the CNN model, the system described in [20] shows the utilization of a word embedding layer. The CNN model is described as having the following layers: Input, convolution, pooling, embedding, dense, dropout, and finally a sigmoid activation function for output. Thus, CNN's appear to be an integral approach to the task of classification.

CNN's can also be utilized to classify the topic of a given text input. Reference [21] discusses the development process of natural language processing (NLP) and explores the following related architectures: word2vec, long short-term memory (LSTM), and text CNN. To compare the LSTM and CNN the loss, accuracy, and mean absolute error calculations were

used. The Accuracy of text CNN was concluded to be better than LSTM, but there were also attention versions of these models included in the comparison. The text CNN with self-attention performed far better than its text CNN counterpart without self-attention. The dataset used to obtain the experimental results was a small and simple text dataset of 461 sentences split into classifications of either computer, or traffic. Thus, it appears adding an attention component to a DNN can enhance performance and produce better results.

Recent research has also been conducted to produce novel approaches to text classification with DNN's as in [22]. In this research, the authors propose an approach to the dynamic graph convolutional neural network (DGCNN) model for text classification. This novel approach is such that the DGCNN takes on a dual-channel forward/backward bidirectional threshold recurrent unit (D-GRU) to extract as much text context feature information as possible. The proposed DGCNN model was shown to outperform a region with CNN (RCNN) model by obtaining a higher accuracy score, obtaining more context feature information, and saving on training time. Thus, neural networks appear to be a valid approach for the task of text classification.

Topic Modeling Short Corpus

Natural language processing (NLP) was originally created in the 1940's after World War II when people realized the importance of language translation [23]. There are many different subsets of NLP. Semantic analysis is the process of analyzing the singular words of a given prose regarding its grammatical structure to obtain meaning from the text. This type of analysis can use context clues around a word to determine the meaning of the word, and therefore the entire text. Sentiment analysis is the process of analyzing text word by word, assigning

numerical values to each word, and taking the summation of these values. This is done to determine if the text is overall positive, negative, or neutral. Thematic analysis is the process of performing qualitative analysis on text to determine patterns of meaning.

Topic modeling is yet another subset of NLP and typically requires the use of larger corpora to provide acceptable accuracy. The subset of NLP, topic modeling, was originally a method introduced as more and more information was placed in a digital format rather than on traditional pieces of paper. As this digital format of information grew larger, it was quickly realized it would be beneficial to devise a way to consume the digital data down to a smaller dataset to allow for a faster and easier way for people to understand the data [24]. One of the first topic models was described in [25] where latent semantic indexing (LSI) makes use of early topic modeling theorems based on spectral analysis of a term-document matrix. These theorems were designed to analyze large corpora.

Research has also been conducted recently to improve upon the topic modeling approach through the utilization of word embedding. A word embedding can be defined as a representation of a word in vector space. The inclusion of word embedding provides an increase to topic coherence. Authors investigate in [26] if making use of trained word embedding vectors over larger corpora such as Wikipedia could improve upon the lack of data obtained from smaller corpora. This methodology improved upon the topic coherence through the utilization of word embedding, but not topic diversity. When word embedding is utilized in the context detection process it could be noted that this enables the similarity measurement, therefore allowing the comparison between the words. Thus, word embedding is a methodology that may be utilized for improving upon topic modeling short corpus.

Reference [27] provides a larger overview of the current methods in use today and explores topic modelling (semantics) as well as the models/algorithms used to obtain the topics. Standard models are the vector space model (VSM), latent semantic analysis (LSA), probabilistic latent semantic analysis (PLSA), Latent Dirichlet Allocation (LDA), and Multinomial Mixture (MM). Clustering models are TermCut, WordCom, Gibbs Sampling Dirichlet Multinomial Mixture (GSDMM), Generalized Polya Urn -Dirichlet Multinomial Mixture (GPU-DMM), Biterm Topic Model (BTM), and Pitman-Yor process mixture model (PYPM). Self-aggregating models are self-aggregation-based topic modeling (SATM) and pseudo-document-based topic modeling (PTM). Deep learning models investigated were the recurrent neural network + biterm Topic Model (BTM) and long short-term memory (LSTM) topic matrix factorization (LTMF). According to [27], the deep learning topic modeling methods such as CNN and RNN face problems such as a failure to retain coherence given small window size and an inability to retain contextual information given an increased sentence length, respectively. This speaks to the difficulty faced when attempting to obtain topics from short text. When text classification via neural network, topic modeling short corpus, and context detection technologies are utilized in tandem, they can provide a powerful combination for data inference.

Therefore, based on the above literature, the trend with context detection methodologies appears to be headed in multiple directions. For example, in [20], [21], and [22] audio data was used as the main data medium for context detection, whereas in [5] and [19] sensors were used. However, these approaches to context detection are limited to audio and sensor-based feature data that do not account for any direct interactivity with a user. As

context detection is incorporated into the smart home for use with virtual assistants, it should be done with as much supporting data as possible. When inconsistent interaction with a virtual assistant occurs for a user in a smart home, they are less likely to become influenced by the virtual assistant [9]. This has led me to my research topic which focuses on context detection that incorporates emotion feature data, which contrasts with the previously described systems that solely utilize audio and or sensor-based feature data. My research works toward a better user experience in a smart home setting. This hypothesis is evaluated with voice-based context detection to help adapt system behavior based on the user emotion and identified context to make smart homes “smarter”.

PROPOSED APPROACH

In this section, I present my proposed approach for context detection. The proposal was presented as a poster and accepted for the IEEE Consumer Communications & Networking Conference in August 2023. An overview of the proposed approach is illustrated in Figure 1 which consists of four major steps: 1. Emotion detection, 2. Basic context detection, 3. WERKS, and 4. Ensemble defined context classification. In step 1, the user's audio data input has its MEL and MFCC features extracted and classified as an emotion to address research question 2, found in the research questions section above. The user's audio data input is also utilized in step 2, where audio data processing and noun phrase detection occur to output a noun phrase bigram context. In step 3, the output from steps 1 and 2 are utilized in my novel WERKS approach, addressing research question 1, which also utilizes my emotion word bank synsets to output a WERKS context. Finally, in step 4 there are three defined context classifiers that take the user's audio data, noun-phrase bigram (from step 2), and WERKS context (from step 3) as input. This input was obtained from the previous three steps and is utilized to predict three defined contexts, which are taken as input into a final defined context classifier as an ensemble learning methodology to predict the final defined context. The subsections that follow will provide further detail as to the logic of how these four steps function.

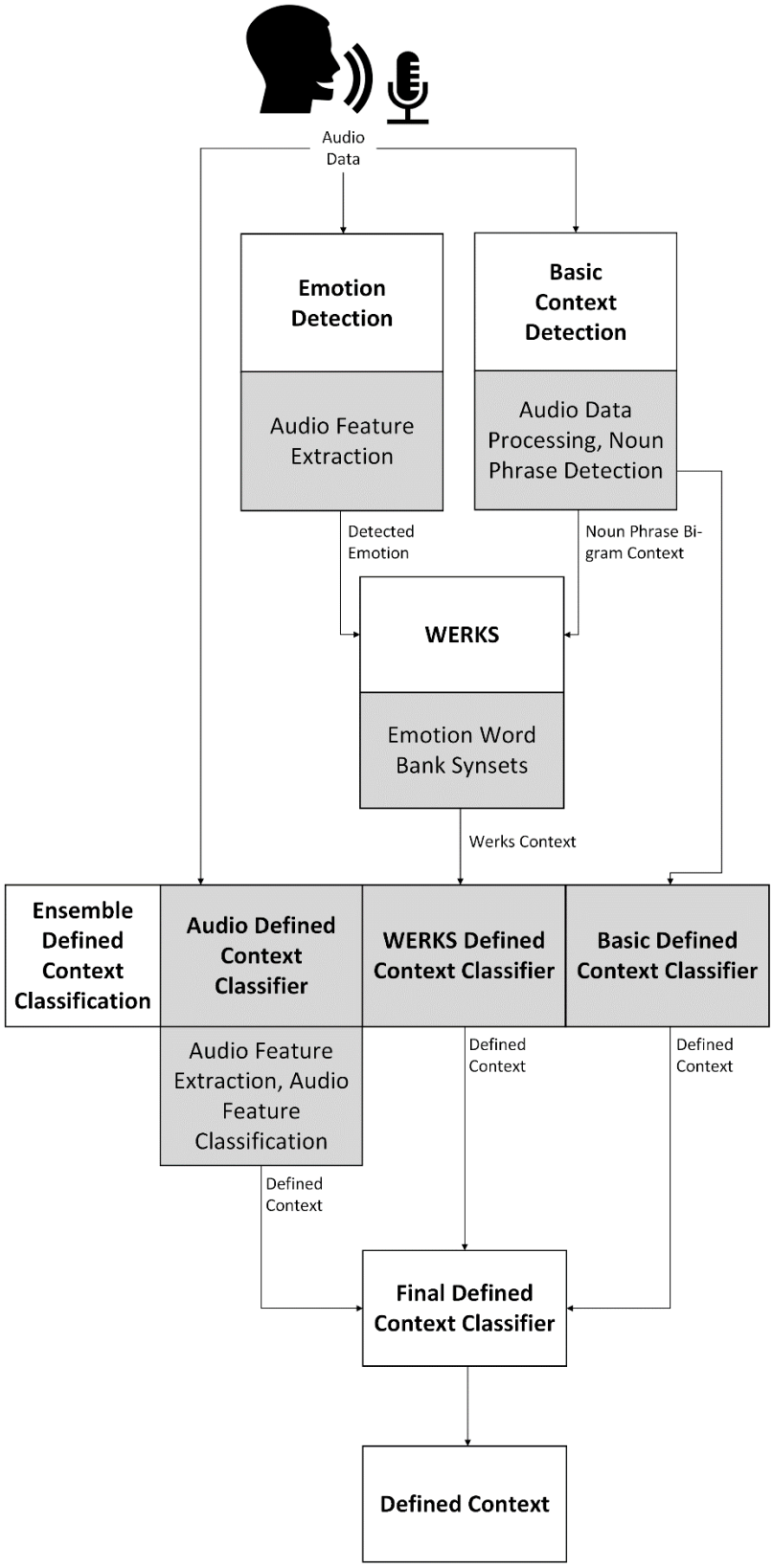

Figure 1: Overview of the proposed WERKS approach.

Emotion Detection

In this step, the voice input is passed to the emotion detection layer. Five emotions have been considered in this layer - angry, fearful, happy, normal, and sad. These emotions were chosen based on the available data sets and they also best fit smart home application contexts [28].

For emotion detection, the MFCC and the MEL features were extracted to be utilized by the emotion classifier to predict the user's emotional state. Before the emotion classifier can accurately predict the user's emotion, it is first trained on the three data sets which are described in full detail in the Data Sets section. The RAVDESS [29] and CREMA [30] data sets were chosen as they are comprised of audio files spoken in various emotional states, such as happy and sad; however, these data sets contain more emotions than under consideration within this thesis. As a result, the RAVDESS and CREMA data was filtered to only audios with emotions classified as angry, sad, normal, happy, and fearful. The SHVC data set only contains these five emotions, and therefore, no filtering was required in the classifier training process.

After the emotion detection classifier was trained, it produces emotion predictions from speech analysis with the MFCC and MEL features. Each time the user interacts with the smart assistant, the emotional state is predicted by the emotion classifier. Based on the predicted emotion it is utilized to determine the appropriate emotion word bank synset, described in the Emotion Word Bank Synsets section.

To provide a visual representation of the differences between the chosen emotions, Figure 2, Figure 3, Figure 4, Figure 5, and Figure 6 present a spectrogram for the angry, fearful, happy, normal, and sad emotions, respectively. These spectrograms are an overlapped visual

representation of Fast Fourier transformations (FFTs) displayed in order of occurrence. Each of these spectrograms are from the same “go home” command, spoken by the same actor with the only difference being the spoken emotion.

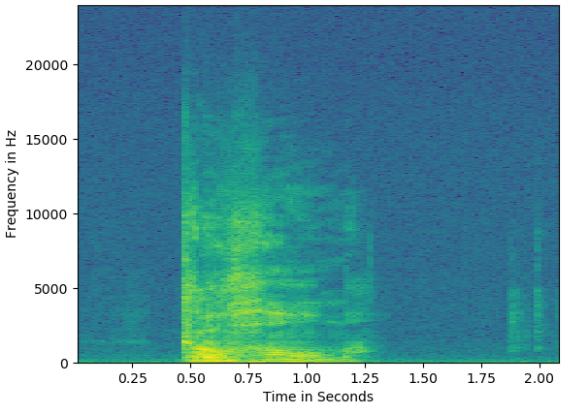

Figure 2: Spectrogram for command “go home” spoken with angry emotion.

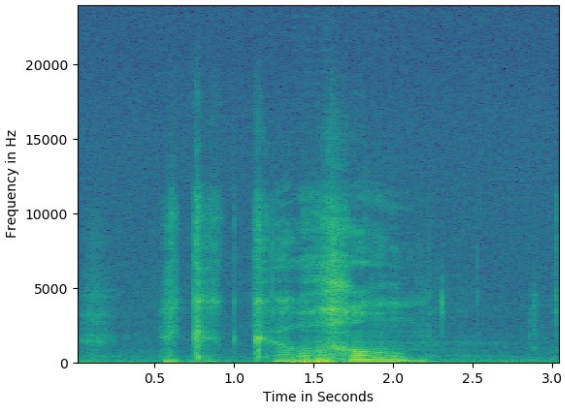

Figure 3: Spectrogram for command “go home” spoken with fearful emotion.

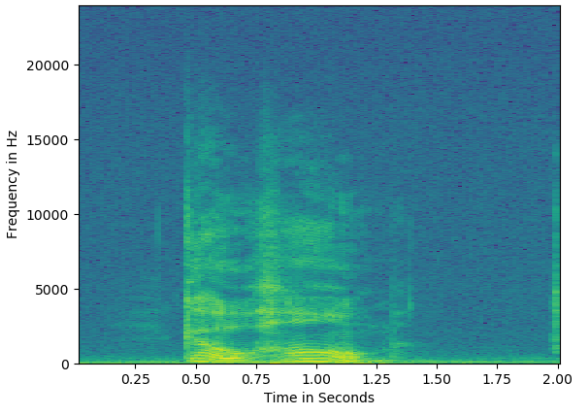

Figure 4: Spectrogram for command “go home” spoken with happy emotion.

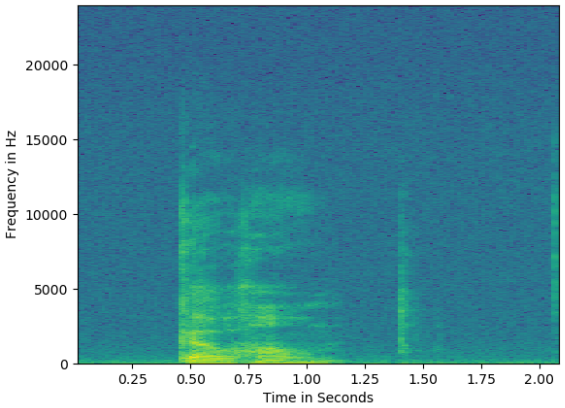

Figure 5: Spectrogram for command “go home” spoken with normal emotion.

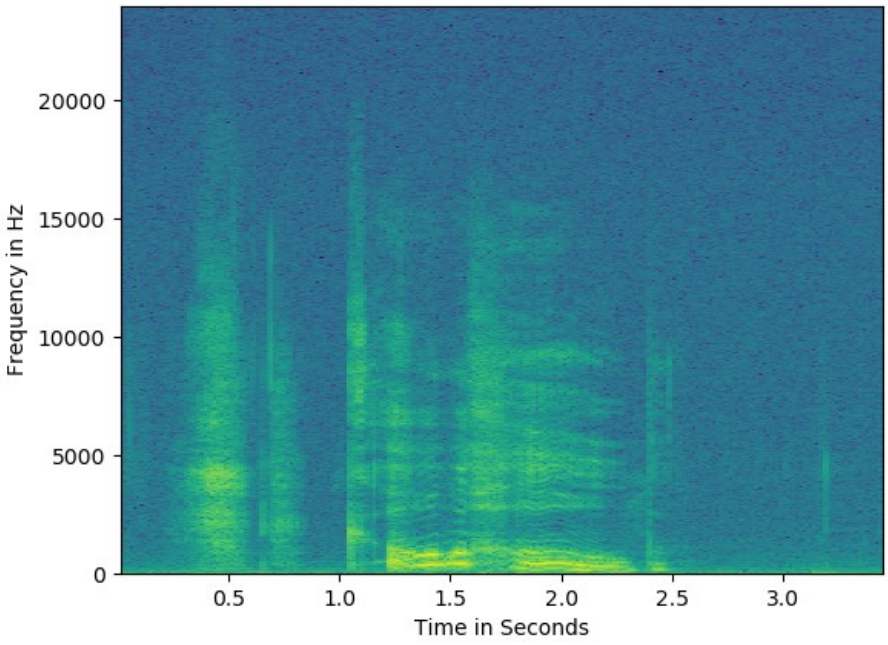

Figure 6: Spectrogram for command “go home” spoken with sad emotion.

Basic Context Detection

For basic context detection, the audio data is converted into text, which then goes through an audio data processing step for the removal of unnecessary text features to produce a “cleaned-up” text. The cleaned-up text is utilized in the noun phrase detection step to identify noun phrases for the next and final step, context detection. In the context detection step, the goal is to identify a noun phrase bigram to serve as the basic bigram context for the system.

Audio Data Processing. The google speech API was utilized to convert the audio data to text. Using regular expressions, characters unrelated to the context were removed such as punctuation symbols, signs, and brackets. Stop words, such as “is”, “at”, “the”, “won”, and “as” are also removed using python's natural language toolkit. The full list of stop words can be

found in Appendix A. These two steps provide a cleaner and more relevant text for basic context detection.

Noun Phrase Detection. In Algorithm 1, the noun phrase detection step logic has been outlined. First, all the bigrams from a given user input are obtained from the cleaned text as described in the Audio Data Processing section above. To achieve this, each word is iterated over for the user text input to obtain the lateral two-word combinations. The next step is to iterate over each of the obtained bigrams and check each of them for a noun phrase. If the bigram functions as a noun, this bigram is considered a noun phrase, and it is allowed to continue in the next processing step. In the case in which no noun phrases are detected, then the list of bigrams is continued into the next processing step instead.

For example, if an audio data from a user is input into the system as “Who’s president of the USA now?”, then the system’s noun phrase detection would perform the following steps: First, the user audio data is “cleaned” to remove stop words and characters unrelated to context. This results in the user input being reduced to “whos president USA”. Next, the reduced user input is iterated over to obtain all possible bigrams. This results in a list of bigrams containing “whos president” and “president USA”, I will refer to this list as the bigram list. Finally, the bigram list (containing “whos president” and “president USA”) is also iterated over, however, this time each bigram is being checked as a noun phrase. Each bigram that is identified as a noun phrase is added to another list, I will refer to this list as the noun phrase bigram list. Thus, during iteration over the bigram list, the bigram “whos president” functions as a noun phrase and is added to the noun phrase bigram list. The bigram “president USA” does not function as a noun phrase and is not added to the noun phrase bigram list. Therefore, the

final noun phrase bigram list contains the bigram “whos president”, as it was the only bigram that is also a noun phrase bigram.

Algorithm 1 Noun Phrase Detection

```
Require: T = Text of user audio data.
Require: S = [A list of stop words.]
  words = regex.remove(regex, T)
  for word in words do
    if word in S then
      words.remove(word)
    end if
  end for
  bigrams = [An empty list]
  for i in len(words) do
    bigrams.append(words[i++] words[i])
  end for
  nounPhrases = [An empty list]
  for bigram in bigrams do
    if bigram is nounPhrase then
      nounPhrases.append(bigram)
    end if
  end for
```

Context Detection. To start context detection, each obtained bigram is converted into a token. For this thesis, a “token” is utilized as a numeric representation of any given bigram. Each token is stored in a matrix of these respective tokens for the given bigram. This allows the implemented logic to find the frequency at which each noun phrase bigram occurs in the final step of this layer. In the final step, the frequency of occurrence for each of the noun phrases has been cumulated. This iterating process is accomplished by going through the current noun phrases and keeping count of each individual noun phrase. For example, if the noun phrase ‘painfully cold’ is encountered twice during iteration, then the respective frequency of this noun phrase would be equivalent to 2. Each of the noun phrases is compared by frequency to find the noun phrase that occurs the most, i.e., has the highest frequency. This most frequently

occurring noun phrase is considered to be the basic bigram context for the proposed approach; however, if no noun phrase occurs more frequently than the rest, the first from the bigram list is chosen to serve as the basic bigram context instead.

Algorithm 2 Context Detection

Require: $nounPhrases = [\texttt{A list of noun phrases.}]$
  $C = CountVectorizor()$
  $matrix = C.fit\_transform(nounPhrases)$
  $totals = matrix.sum(axis = 0)$
  if `totals.hasMax()` **then**
    $context = bigrams[totals.max()]$
  else
    $context = bigrams[0]$
  end if

In Table 1, examples are given for the respective bigram and noun phrase columns obtained from three input sentences after completion of the noun phrase detection step; these tables provide examples for input audio data as “Who's the president of USA now", “How's traffic today in Springfield”, and “What's the Italian of good morning” respectively. Bigrams which cannot be used as a noun phrase are represented as not applicable (N/A).

Table 1. Bigrams and the respective noun phrase representations for input sentences.

Sentence	Bigram	Noun Phrase
Who's the president of USA now	whos president	whos president
	president usa	president usa
How's traffic today in Springfield	hows traffic	hows traffic
	traffic today	N/A
	today springfield	N/A
What's the Italian of good morning	whats Italian	whats Italian
	italian good	italian good
	good morning	good morning

WERKS

In this section, I provide a comprehensive overview of the emotion word bank synsets utilized within the WERKS layer of the system. Additionally, I detail the word embedding and the emotionally relevant keyword search processes that produce the WERKS contexts.

Emotion Word Bank Synsets. Each emotion under consideration (happy, angry, sad, fearful, and normal) has its own associated word bank where synonyms for the respective emotion are kept. I have created these word banks utilizing the Merriam-Webster thesaurus such that each available synonym found in [31], [32], [33], and [34] was placed into a text file as 'happySynset.txt', 'angrySynset.txt', 'sadSynset.txt', and 'fearfulSynset.txt', respectively. The

word banks are utilized with word embedding in the WERKS layer, as described in the immediately following section, to determine the new WERKS context.

The four word banks that I have created are representative of words that are synonyms for each of the considered emotions (happy, angry, sad, fearful, and normal) except for the normal emotion. Each of these word banks is known as a synset. The 'normal' emotion does not have a word bank synset because the basic bigram context is not altered when the user emotion is detected as 'normal', which is described in detail in the Basic Context Detection section (found directly above this section). Therefore, the word banks for each of the respective emotions under consideration in this thesis are happy, angry, sad, and fearful.

The word bank synsets are a set of 163, 108, 140, and 150 synonyms for the emotions happy, angry, sad, and fearful, respectively. Examples of the words in the happy synset are words such as acceptable, capable, and cheerful. For the sad synset, examples are words such as sorry, poor, and pathetic. The angry synset holds words like vengeful, irate, and embittered. Finally, the fearful synset holds words such as terrified, dismayed, and nervous.

In the system, each word bank is represented as a text file such that each of the synsets are stored as 'emotionSynset.txt' where 'emotion' is respective of the considered emotions. For example, a file name for one considered emotion, 'sad', is named 'sadSynset.txt'. The proposed system makes use of the detected emotional state of the user (from the considered emotions) to choose between the respective word banks. If the system detected the user emotion as 'angry', the system would choose to apply word embedding and obtain a vector space for the 'angrySynset.txt' word bank as described in the WERKS section.

Now, to provide further insight into the interaction that occurs between the emotion word bank synsets and the WERKS context methodology, I will provide a brief example of a WERKS context the system could produce in the WERKS layer as seen in Figure 1. If the system detected the user emotion and basic bigram context as ‘angry’ and ‘good morning’, respectively, the WERKS layer would perform word embedding on the ‘angrySynset.txt’ file (the angry emotion word bank synset). One of the words that the angry synset contains is the word ‘unfriendly’, thus one WERKS context the system could produce would be ‘unfriendly morning’, as the word ‘unfriendly’ provides further emotional context than the basic bigram context. Further, if the emotion detected from the user natural language input is as ‘normal’, then the system does not change the obtained context; hence, no normal word bank is necessary.

I have provided the emotion word bank synsets in their entirety in Appendix B which displays the happy, angry, sad, and fearful synsets respectively.

WERKS Context. After basic context detection and emotion detection is completed as described in section Basic Context Detection and Emotion Detection respectively, the detected basic bigram context and detected emotion are passed to the WERKS layer as seen in Figure 1. In WERKS, the first step is to create a vector space related to the synsets described in the Emotion Word Bank Synsets section. Next, the system obtains a keyword from the basic bigram context to be used for searching the emotionally relevant vector space for its most similar word. The word found through this process is then utilized to update the basic bigram context, thus resulting in the WERKS context.

Algorithm 3 Word Embedding with Emotionally Relevant Keyword Search

Require: $E =$ `The user emotion`
Require: $C =$ `The basic bi-gram context`
  $emotions = [$`A list of emotions`$]$ ▷ The emotions are angry, fearful, happy, and sad.
  $werksContext = C$
  $biGramFirstWord = C[0]$
  if `E in emotions` **then**
    $synset = [$`A list of synonyms for E`$]$
    $wordVector = WordVector(synset)$
    $mostSimiliar = MostSimiliar(biGramFirstWord)$
    $werksContext[0] = mostSimiliar$
  end if

Algorithm 3 outlines the pseudocode describing the implementation of the WERKS logic. The WERKS logic does not alter the basic bigram context if the user emotion is normal, thus the considered emotions within the logic are exclusively the following: Angry, fearful, happy, and sad. It starts by taking the detected emotion as a parameter and matching it with the appropriate word bank synset. For example, if the emotion detected was ‘sad’ then the system would utilize the sad synset (of 140 words in length) to perform word embedding to represent each word in a vector space. I use word embedding in this thesis as a relational tool such that the closer each word is to another word in the vector space, the closer these words are in meaning [35]. With each word in the created vector space being a synonym for the emotion, in this example ‘sad’, I consider each word in the vector space to hold emotional relevancy for the given emotion. The method in which I obtain a similar word for a given keyword in vector space is such that I find the word closest to the given keyword in the vector space. Once this word embedding step is complete and I have the vector space set up and ready for searching, I then move on to getting an appropriate keyword to search this vector space with.

The next step is to identify and isolate an appropriate keyword to utilize in searching the vector space with. To accomplish this step, I make use of the basic bigram context. First, I check if this basic bigram context is a noun phrase to ensure I isolate the first word in the noun phrase as the respective keyword. Otherwise, when the basic bigram context is not a noun phrase I pick the first word of the bigram as the keyword. After I have identified and isolated the keyword from the basic bigram context, I must search the vector space for the most similar word.

To search the vector space for the most similar word of a given keyword I find the closest vector to the keyword's vector. As this vector is a representation of a word, this closest vector is considered the most similar word within the given vector space. If the detected emotion and basic bigram context of the system were ‘sad’ and ‘painfully cold’ respectively, then the WERKS layer would proceed with these parameters in the following steps. First, the ‘sad’ emotion word bank synset would be word embedded into a vector space. This vector space would then be searched for a most similar word given the keyword ‘painfully’ where the most similar word found would be ‘woefully’. The basic bigram context would then have its non-noun word replaced to become ‘woefully cold’; this is now what is considered the WERKS context by this thesis. I consider the WERKS context to provide an emotionally relevant context to be a more accurate representation of the user context from the natural language input. This context attempts to improve the user experience with the virtual assistants in the smart home environment.

Ensemble Defined Context Classification

Ensemble learning is a methodology for machine learning that is utilized to achieve state-of-the-art performance through the combination of predictions from multiple base models [36]. The disadvantages to ensemble learning are such that a system must be built larger, to accommodate multiple base models and an ensemble learner, as well as an increase to training and prediction times due to the multiple base models; however, I utilize the ensemble learning methodology to enhance my approach to context detection in pursuit of building a system to obtain the highest possible prediction accuracy. In the first two layers of my proposed approach, two separate models are used for producing separate contexts as described in the above sections, Basic Context Detection and WERKS. Utilizing these separate contexts as feature data, I incorporate ensemble learning into my proposed approach for the purpose of context detection. In the system, I refer to this ensemble learning approach as ensemble defined context classification. To achieve ensemble defined context classification, I created the basic, WERKS, and audio-defined context classifiers (detailed in the Classifier Implementation section) to serve as base models which consume basic contexts, WERKS contexts, and MEL and MFCC feature data, respectively. These three base models are used to produce a defined context, per model, that are ultimately consumed by a final defined context classifier. This final classifier is the ensemble learner of the proposed system.

To create the base models of the ensemble defined context classification layer, I used the TPOT classification technique. This classification technique is known as an automated machine learning (AutoML) process. There are many classification techniques, for example there is the decision tree, adaptive boosting (AdaBoost), support vector machine (SVM), K-

nearest neighbor (KNN), etc. Normally, a system designer would need to tediously choose the perfect parameters and preprocess data to be consumed on a per classifier basis. AutoML tools such as TPOT have been created to streamline and address the need for reducing this time intensive process [37]. I have chosen to utilize TPOT, as this AutoML process consists of machine learning pipelines that consist of a smaller number of operators during training while also providing high performance [38].

The proposed approach utilizes the three base models (basic, WERKS, and audio-defined context classifiers) in the ensemble-defined context classification step. The final classifier takes predictions from these three as feature data to predict the final defined context. The three classifiers in the ensemble classification layer of the proposed approach are the basic, WERKS, and audio defined context classifiers which consist of a gradient boosting classifier, multiple layer perceptron (MLP), and extreme gradient boosting (XGB) classifier, respectively. These classifiers were chosen by the TPOT to provide the highest accuracy possible for the data sets described in the Data Sets section. The accuracy obtained for each of these classifiers can be found in the Evaluation section.

In lieu of training and testing a multitude of classifiers to determine the best performer over the considered data sets (found in the Data Sets section), I utilized TPOT to perform this (training/testing) task by consuming the respective feature data per classifier (basic, WERKS, and audio-defined context classifiers), as this was the only variable in the training process. The TPOT utilizes the scikit-learn API, thus providing high performance and reproducibility [38]. The classifiers that TPOT chose for the basic, WERKS, and audio-defined context classifiers (gradient boosting classifier, MLP, and XGB classifier, respectively) were kept for the following reasons:

The basic-defined context classifier is a gradient boosting classifier which is known to perform better on weak areas of previous models during training and will typically provide less biased predictions than would be obtained from a single model [39]. The WERKS-defined context classifier is a MLP which is an ANN that contains multiple hidden networks, allowing it to solve nonlinear problems such as context detection (multiple-class classification) [40]. The audio-defined context classifier is an XGB classifier which is known to train quickly while also providing high accuracy, even with missing data and nonlinear problems [41]. The final-defined context classifier is also an XGB classifier like the audio-defined context classifier; thus, it provides the same advantages for the system.

The audio defined context classifier makes use of two steps to achieve defined context prediction. The first step is feature extraction which ensures the audio features are obtained from the user audio data input. The same audio feature data obtained from the emotion detection layer described in the Emotion Detection section serves as the feature data for the audio defined context classifier; this is another XGB classifier to predict what defined context a given user audio data input should be categorized.

To achieve the prediction accuracy described in the Evaluation section, the final defined context classifier makes use of the SHVC, RAVDESS, and CREMA data sets which are detailed in the Data Sets section. Each audio data input from these data sets is utilized as input for the audio defined context classifier and processed down to the MFCC and MEL features. These are separated into an 80% training and 20% testing split. After the audio-defined context classifier is trained on this split training data, the test data is used in the accuracy measurement described in the Accuracy section to obtain an 86% defined context prediction accuracy.

The final defined context classifier is also an XGB classifier. It has been trained utilizing the defined context predictions from the basic, WERKS, and audio-defined context classifiers. To obtain an accuracy measurement for the final classifier the defined contexts are used in the accuracy measurement defined in Equation 4, Equation 5, and Equation 6, that can be found in the Accuracy section of this thesis.

IMPLEMENTATION

This section will introduce and explore the different classifiers utilized by the proposed approach described in the Proposed Approach section as well as the data sets the classifiers process.

Classifier Implementation

This subsection will provide details for each classifier utilized within the system as described in the Proposed Approach section regarding the training and prediction algorithm for each classifier, as well as classifier recreation steps. To implement any of the classifiers observed in this system, the classifier recreation steps always utilize the TPOT to train and test a new classifier. The Tree-based Pipeline Optimization Tool Training/Testing section found below will elaborate upon what the TPOT is, why it was utilized for this system, and how to use it to train and test a new classifier.

Tree-based Pipeline Optimization Tool Training/Testing. The immediately following sections pertain to a given classifier, Basic/WERKS/Audio/Final Defined Context Classifier, and all utilize the automated machine learning (AutoML) technique, the Tree-based Pipleine Optimization Tool (TPOT). The TPOT is used for creating and fitting (training/testing) the most optimal classification technique given the combined data set as described in the Data Sets section below. The exact same (combined) data set is utilized along with the same parameters in the creation steps for each classifier except for the Final Defined Context Classifier, which is trained on the predictions of the first three (base model) classifiers (Basic/WERKS/Audio Defined Context Classifiers). The variable `X` that is found in the code snippets of this section is the only variable during the training/testing process. The implementation for this variable is

described in detail for each of the immediately following Basic/WERKS/Audio/Final Defined Context Classifier sections. For the sake of brevity, the following steps will describe exactly how I implemented these classifiers, rather than generic instructions for any personal computer running any operating system. I also assume that tasks such as code editor download/installation and library installation/importation are trivial, thus, these details are omitted.

My personal computer, which I used for all classifier implementation, is running on the windows 10 operating system and all code was written with the python programming language (version 3.11.7) in the visual studio code source-code editor. The steps I followed for classifier implementation are as follows: Step 1. Open visual studio code and create a python file. Step 2. In the newly created python file, import the TPOTClassifier library and copy/paste the following code into the python file on a single line:

$$classifier = TPOTClassifier(n\_jobs = -1, generations = 5, population\_size = 5000, scoring = 'accuracy', verbosity = 3, random\_state = 7)$$

The parameters of this snippet serve the following purposes: n_jobs when set to -1 will utilize all available CPU cores during the training process, generations is the iterations the pipeline will run for optimization, population_size is how many individuals to retain per generation in the genetic programming, scoring sets the type of measurement to perform, verbosity will display more information in the terminal during training, and random_state is a seed to obtain the same results each time TPOT is ran (given the same data set). Step 3. Fit the combined data set to the optimized classifier with the following code snippet:

$$classifier.fit(X, y)$$

Finally, in Step 4. Run the python file. Step 4.a. This step is optional if saving the classifier is not desired. Import the pickle library and use the following code snippet to save the classifier:

$$with\ open('classifier.pkl','wb')\ as\ f:$$

$$pickle.dump(classifier.fitted\_pipeline\_, f)$$

These four steps are the same for each classifier described in the immediately following sections; however, there would be no difference in the classifiers if the data they were fit with was the same. The `X` variable found in the code snippets above differs such that the Basic, WERKS, Audio, and Final Defined Context Classifiers train utilizing the basic contexts, WERKS contexts, MEL and MFCC features, and the predictions of the three previous classifiers, respectively. The implementation for the `X` variable per classifier is described in full detail in the immediately following sections.

Basic Defined Context Classifier. To train the basic defined context classifier each data set found in the Data Sets section is iterated through to obtain each basic context per audio data as described in the Basic Context Detection section. The combined data set contains these basic contexts, which are utilized as feature data and split into 80% training data and 20% testing data to predict defined contexts. The TPOT API was utilized to optimize this classifier through its intelligent search methodology, previously described in the Tree-based Pipeline Optimization Tool Training/Testing section found above, with the training data `X` referring to the basic contexts. The TPOT training resulted in a gradient boosting classifier. The algorithm for this classifier type is presented in Algorithm 4 [39]. This algorithm consumes the training data by initializing the model with a constant value chosen by the AutoML TPOT and then the training data is looped through with the following four steps: First, compute pseudo-residuals.

A pseudo-residual is the difference between the observed and predicted value. This can be observed in the third line of Algorithm 4. Second, fit the base learner to the pseudo-residuals. Third, solve the one-dimensional optimization problem. Finally, step four is to update the model. Steps 1, 2, 3, and 4 can all be observed in Algorithm 4 on lines 3, 4, 5, and 6, respectively.

Algorithm 4 Gradient Boost

$\mathrm{F}_0\,(\mathrm{x}) = \arg\min_\gamma \sum_{i=1}^{N} \Psi(y_i, \gamma)$

For $\mathrm{m} = 1$ to M do:

$\tilde{\mathrm{y}}_{im} = -\left[\frac{\partial \Psi(\mathrm{y}_i, \mathrm{F}(\mathrm{x}_i))}{\partial F(\mathrm{x}_i)}\right]_{F(x)=F_{m-1}(x)}, i = 1, N$

$\{R_{lm}\}_1^L = L - terminal\ node\ tree(\{\tilde{y}im, \mathrm{x}_i\}_1^N)$

$\gamma_{lm} = \arg\min_\gamma \sum_{x_i \epsilon R_{lm}} \Psi(\mathrm{y}_i, \mathrm{F}_{m-1}(\mathrm{x}_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + v * \gamma_{lm} 1(x \epsilon R_{lm})$

endFor

WERKS Defined Context Classifier. To train the WERKS defined context classifier each data set found in the Data Sets section is iterated through to obtain each WERKS context per audio data as described in the WERKS section. The combined data set contains these WERKS contexts, which are utilized as feature data, split into 80% training data and 20% testing data to predict defined contexts. The TPOT API was utilized to optimize this classifier through its intelligent search methodology, previously described in the Tree-based Pipeline Optimization Tool Training/Testing section found above, with the training data `X` referring to the WERKS contexts. The TPOT training resulted in a MLP classifier. The equation for this classifier type is presented in Equation 1 below, where η is the learning rate and Loss is the loss function for the

MLP [40]. As the name suggests, the MLP is an artificial neural network that consists of multiple hidden layers. The neurons are connected by edges. An MLP begins with an input layer, then has the hidden layer in the middle, and finally has an output layer at the end. The input layer is only utilized to bring data into the classifier and feed it into the further layers. The hidden layer accepts the data from the input layer and applies a weight to it, hence the `$w$` symbol in Equation 1. Finally, the output layer produces an output from the weighted data from the hidden layer via an activation function.

$$w \leftarrow w - \eta(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial Loss}{\partial w}) \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \quad (1)$$

Audio Defined Context Classifier. To train the audio defined context classifier each data set found in the Data Sets section is iterated through to obtain the MFCC and MEL feature data per audio data. This MFCC and MEL feature data is then split into 80% training data and 20% testing data to predict defined contexts. The TPOT API was utilized to optimize this classifier through its intelligent search methodology, previously described in the Tree-based Pipeline Optimization Tool Training/Testing section found above, with the training data `X` referring to the MFCC and MEL features. The TPOT training resulted in an XGB classifier. The same algorithm that is found in the Basic Defined Context Classifier section above, Algorithm 4 Gradient Boost, is utilized by this classifier; however, the XGB classifier type makes use of L1 & L2 regularization. These regularization equations can be observed in Equation 2 and Equation 3 below [41].

$$L1 = \sum_{i=1}^{n} |y_{true} - y_{predicted}| \quad (2)$$

$$L2 = \sum_{i=1}^{n} \left(y_{true} - y_{predicted}\right)^2 \quad (3)$$

Final Defined Context Classifier. To train the final defined context classifier each data set found in the Data Sets section is iterated through to obtain basic, WERKS, and audio defined context classifier predictions to be utilized as a feature per audio data. This feature data is then split into 80% training data and 20% testing data to predict defined contexts. The TPOT API was utilized to optimize this classifier through its intelligent search methodology, previously described in the Tree-based Pipeline Optimization Tool Training/Testing section found above, with the training data `X` referring to the basic contexts. The TPOT training resulted in another XGB classifier, this is the same type of classifier as the Audio Defined Context Classifier. Thus, with this Final Defined Context Classifier being an XGB classifier, it utilizes the exact same algorithm as found in the Audio Defined Context Classifier section above. This XGB classifier type is presented in Algorithm 4 where L1 & L2 regularization are utilized, which can be found in Equation 2 and Equation 3 directly above this section [41].

Data Sets

The system utilizes three data sets – the first data set in use is the Ryerson audio-visual database of emotional speech and song (RAVDESS) [29], the second is a data set which was custom made as a smart home voice command (SHVC) data set, and the third is the crowd-sourced emotional multimodal actors (CREMA) data set [30]. The RAVDESS data set is not inherently meant to be used with the system, as it does not contain any direct smart home

commands which the system was modeled for. However, it was included to provide a comparison to the SHVC data set and to make observations on a more general data set. The same can be said for the CREMA data set, however it does provide 12 statements rather than the 2 that RAVDESS provides.

RAVDESS. The RAVDESS data set contains only two audio data statements as “Kids are sitting by the door” and “Dogs are sitting by the door”. These statements are expressed by the actors of the data set in 8 separate emotions, where the emotions are the following: Neutral, calm, happy, sad, angry, fearful, disgust, and surprised. The system is only modeled to handle the considered emotions as described in the Emotion Detection section. This means the system filters out the audio data from this data set where only those which are classified as calm, disgust, and surprised, are not utilized, where neutral was considered as the equivalent to normal.

SHVC. The system also makes use of the SHVC data set. This data set consists of 5000 voice commands which correspond to the following emotions: happy, normal, angry, sad, and fearful. These are the only 5 emotions under consideration by the system and therefore, no filtering of this data set was necessary. These emotions were chosen for the system as they have been found to produce discernable variation in the context of natural spoken language [42]. This data set was created with ten participants who recorded 50 smart home voice commands twice per emotion. These smart home voice commands were recorded in an indoors environment as expected of a smart home environment, in one or more sessions for recording completion. The SHVC data set was chosen for the system as it pertains to the use case of user

interaction with a virtual assistant in the smart home environment. All 50 smart home commands in the SHVC data set can be found in Table 2 below.

Table 2. Smart home voice command data set.

Command	
Go home.	How's the weather tomorrow?
Dial 911.	Is it cold today?
Call Mom.	Cancel the 2pm alarm.
Spell Banana.	Turn the volume down.
Call Police.	Set temperature to 75.
Play music.	What's the best operating system?
Cancel alarm.	Set an alarm for 8am.
Show camera.	Cancel all the alarms today.
Make coffee.	What is my flash briefing?
Lock doors.	Why are you so annoying?
Make it warmer.	Remind me to do laundry.
Play the news.	When is my son's birthday?
Read my calendar.	What do you think of siri?
I am sad.	How's traffic today in Springfield?
Turn on light.	I am not happy today.
Turn off fan.	What is on my calendar today?
Change your voice.	Are you afraid of the dark?
Who are you?	What is the value of Pi?
Lower the temperature.	What is zero divided by zero?

Table 2 continued

Command	
Set 10 minutes timer.	Is it going to snow today?
Tell me a joke.	How will be the weather at night?
Sing me a song.	Who's the president of USA now?
Tell me a story.	What's the Italian of "Good morning"
Read me a haiku.	How's the weather of Bangladesh now?
What's the weather today?	Do not answer to my kid.

CREMA. The CREMA data set consists of twelve statements which are the following: “It's eleven o'clock.”, “That is exactly what happened.”, “I'm on my way to the meeting.”, “I wonder what this is about.”, “The airplane is almost full.”, “Maybe tomorrow it will be cold.”, "I would like a new alarm clock.", "I think I have a doctor's appointment.", "Don't forget a jacket.", "I think I've seen this before.", "The surface is slick.", and "We'll stop in a couple of minutes". The amount of audio data items included in this data set totals out to be 7,442 original items. These statements are not inherently designed for consumption by a smart home system, but this data set was also included for comparison against the SHVC data set. These statements were spoken by 91 different actors; 48 of them were male and 43 were female actors all between the ages of 20 to 74. The actors were also of a large variety of racial and ethnic backgrounds. The statements the actors spoke were spoken in six different emotions including: Anger, disgust, fear, happy, neutral, and sad. To consume these statements, the system filters out the audio data with emotion classified as disgust and considers the audio data classified as neutral to be the equivalent to normal in the system.

Combined Data Set. The system also consumes a combined data set. This data set is not an established data set. Rather, this data set is temporarily loaded into random access memory (RAM) from each of the three main data sets considered by the system, which are RAVDESS, SHVC, and CREMA. To load this data set into RAM, each audio file of the RAVDESS, SHVC, and CREMA data sets were iterated through to obtain the "Command", "Emotion", "Basic Context", "WERKS Context", and finally, the "Defined Context". To iterate through these data sets and obtain the combined data set (variable combinedDataSet below), the code snippet found below can be ran in a python file with the pandas library imported as pd; I continue with the assumption that the data sets are on a Windows 10 machine, as well as in the same directory:

$$ravdessDataFrame = pd.read\_csv('RAVDESS.csv', encoding = 'latin - 1')$$

$$shvcDataFrame = pd.read\_csv('SHVC.csv', encoding = 'latin - 1')$$

$$cremaDataFrame = pd.read\_csv('CREMA.csv', encoding = 'latin - 1')$$

$$combinedDataFrames = [ravdessDataFrame, shvcDataFrame, cremaDataFrame]$$

$$combinedDataSet = pd.concat(combinedDataFrames)$$

Thus, each item/row of this data set corresponds to one full run of the proposed approach. This data set only exists as data loaded into the system memory at one time for training and testing purposes as described in the Classifier Implementation section. This data set in memory is 12,036 items in length and contains each considered item from the three main data sets.

EXPERIMENTAL SETUP AND EVALUATION

This section provides the experimental setup and evaluation for the system which will detail the defined contexts, evaluation, sample outputs, and accuracy. The 6 defined contexts the system utilizes for prediction and accuracy measurements will be introduced. Then, the evaluation will describe the accuracy measurements and confusion matrices for each system classifier, as was previously introduced in the Classifier Implementation section. Additionally, some sample outputs for the system will be provided. Finally, I will provide the accuracy equation utilized to obtain the discussed evaluation results.

Defined Contexts

The system was designed to work with 6 contexts rather than focusing on a smaller number of contexts to classify specific audio data as in [13], where sounds that are sampled from action movies are classified as one of two defined contexts, gun play or car racing. The 6 contexts chosen to be detected during evaluation by the system are security, light, temperature, weather, entertainment, and function. These contexts were chosen because they best fit the data sets under consideration by the system which can be found in the Data Sets section. The defined contexts are represented in the system by single digit integers as can be seen in Table 3.

Table 3. Defined contexts for WERKS approach.

Defined Context	Integer
Security	0
Light	1
Temperature	2
Weather	3
Entertainment	4
Function	5

Evaluation

To address research question 1, as asked in the Research Questions section of the Introduction of this thesis, this subsection will provide accuracy measurements for each of the classifiers considered by the system. The confusion matrices per classifier will also be provided over each specified data set.

Basic Defined Context Classifier. The basic defined context classifier produced accuracies of 65%, 22%, 28%, and 61% over the RAVDESS, CREMA, SHVC, and the combined data set, respectively. The confusion matrix for this classifier over the combined data set can be found below as Figure 7. Further, due to the Basic Defined Context Classifier being trained upon the basic context feature data extracted from the combined data set which was obtained via word embedding (described in the Classifier Implementation section), this classifier exhibits an extreme bias for the 'Weather' defined context. This extreme bias occurs within this classifier's predictions as the combined data set has a multitude of items with the 'Weather' defined context rather than the other defined contexts, thus, when this classifier consumes further

data, it attempts to predict the 'Weather' defined context exclusively. In contrast, Figure 7 also displays this classifier's bias for the 'Weather' defined context; however, not to the extreme that this classifier exhibits over the RAVDESS, CREMA, or SHVC data sets.

The Basic Defined Context Classifier's predictions over the RAVDESS data set were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with basic context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the RAVDESS data set, which consisted of 692 samples. The test data set contained 20% of the RAVDESS data set, which consisted of 173 samples. Of the 173 test samples, no samples were correctly predicted as either the Entertainment, Function, Light, Security, or Temperature defined context. There were 112 correct predictions for the defined context Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 65% accuracy score.

The Basic Defined Context Classifier's predictions over the CREMA data set were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with basic context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the CREMA data set, which consisted of 4,938 samples. The test data set contained 20% of the CREMA data set, which consisted of 1,234 samples. Of the 1,234 test samples, no samples were correctly predicted as either the Entertainment, Function, Light, Security, or Temperature defined context. There were 275 correct predictions for the defined context, Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 22% accuracy score.

The Basic Defined Context Classifier's predictions over the SHVC data set were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with basic context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the SHVC data set, which consisted of 4,000 samples. The test data set contained 20% of the SHVC data set, which consisted of 1,000 samples. Of the 1,000 test samples, no samples were correctly predicted as either the Entertainment, Function, Light, Security, or Temperature defined context. There were 283 correct predictions for the defined context, Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 28% accuracy score.

Figure 7 shows the confusion matrix for the Basic Defined Context Classifier's predictions over the combined data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with basic context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the combined data set, which consisted of 9,629 samples. The test data set contained 20% of the combined data set, which consisted of 2,407 samples. The confusion matrix displayed below as Figure 7 shows that of the 2,407 test samples, the number of correctly predicted defined contexts are as follows: 212 for Entertainment, 109 for Function, 172 for Light, 103 for Security, 252 for Temperature, and 638 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 61% accuracy score.

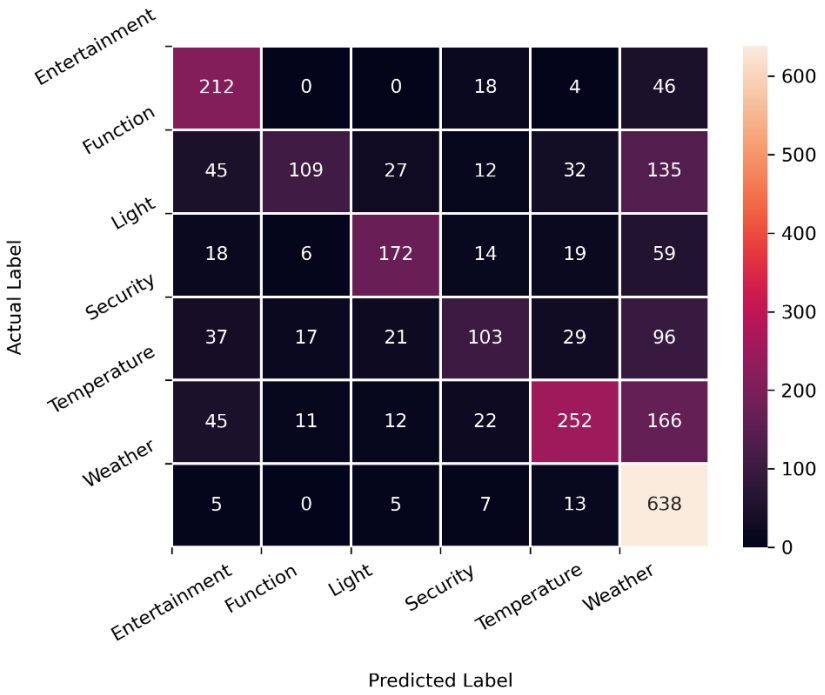

Figure 7: Confusion matrix for the Basic Defined Context Classifier given Combined data set.

WERKS Defined Context Classifier. The WERKS defined context classifier produced accuracies of 65%, 22%, 28%, and 67% over the RAVDESS, CREMA, SHVC, and the combined data set, respectively. The confusion matrix for this classifier over the combined data set can be found below as Figure 8. Further, due to the WERKS Defined Context Classifier being trained upon the WERKS context feature data extracted from the combined data set which was obtained via word embedding (described in the Classifier Implementation section), this classifier exhibits an extreme bias for the 'Weather' defined context. This extreme bias occurs within this classifier's predictions as the combined data set has a multitude of items with the 'Weather' defined context rather than the other defined contexts. Thus, when this classifier encounters further data, it attempts to predict the 'Weather' defined context exclusively. In

contrast, Figure 8 also displays this classifier's bias for the 'Weather' defined context; however, not to the extreme that this classifier exhibits over the RAVDESS, CREMA, or SHVC data sets.

The WERKS Defined Context Classifier's predictions over the RAVDESS data set were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with WERKS context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the RAVDESS data set, which consisted of 692 samples. The test data set contained 20% of the RAVDESS data set, which consisted of 173 samples. Of the 173 test samples, no samples were correctly predicted as either the Entertainment, Function, or Light defined context. There were 112 correct predictions for the defined context Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 65% accuracy score.

The WERKS Defined Context Classifier's predictions over the CREMA data set were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with WERKS context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the CREMA data set, which consisted of 4,938 samples. The test data set contained 20% of the CREMA data set, which consisted of 1,234 samples. Of the 1,234 test samples, no samples were correctly predicted as either the Entertainment, Function, Light, Security, or Temperature defined context. There were 275 correct predictions for the defined context, Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 22% accuracy score.

The WERKS Defined Context Classifier's predictions over the SHVC data set were obtained after training this TPOT AutoML model on the combined data set (described in the

Data Sets section) with WERKS context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the SHVC data set, which consisted of 4,000 samples. The test data set contained 20% of the SHVC data set, which consisted of 1,000 samples. Of the 1,000 test samples, no samples were correctly predicted as either the Entertainment, Function, Light, Security, or Temperature defined context. There were 283 correct predictions for the defined context, Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 28% accuracy score.

Figure 8 shows the confusion matrix for the WERKS Defined Context Classifier's predictions over the combined data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with WERKS context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the combined data set, which consisted of 9,629 samples. The test data set contained 20% of the combined data set, which consisted of 2,407 samples. The confusion matrix displayed below as Figure 8 shows that of the 2,407 test samples, the number of correctly predicted defined contexts are as follows: 265 for Entertainment, 153 for Function, 186 for Light, 79 for Security, 302 for Temperature, and 615 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 67% accuracy score.

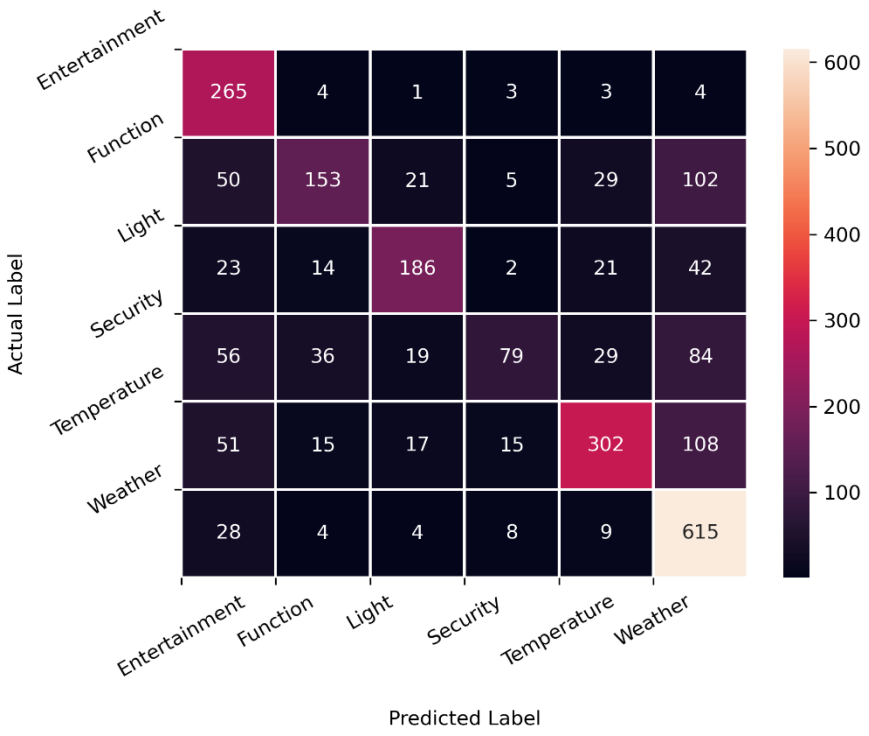

Figure 8: Confusion matrix for WERKS Defined Context Classifier given Combined data set.

Audio Defined Context Classifier. The basic defined context classifier produced accuracies of 71%, 72%, 76%, and 65% over the RAVDESS, CREMA, SHVC, and the combined data set, respectively. The confusion matrix for this classifier over these data sets can be found below as Figure 9, Figure 10, Figure 11, and Figure 12.

Figure 9 shows the confusion matrix for the Audio Defined Context Classifier's predictions over the RAVDESS data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with MEL and MFCC feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the RAVDESS data set, which consisted of 692 samples. The test data set contained 20% of the RAVDESS data set, which consisted of 173 samples. The confusion matrix displayed below as Figure 9 shows that of the

173 test samples, correct predictions were 7 for Entertainment, 8 for Function, and 3 for the Light defined context. There were 106 correct predictions for the defined context Security, thus the remaining samples were misclassified. This confusion matrix resulted in a 71% accuracy score.

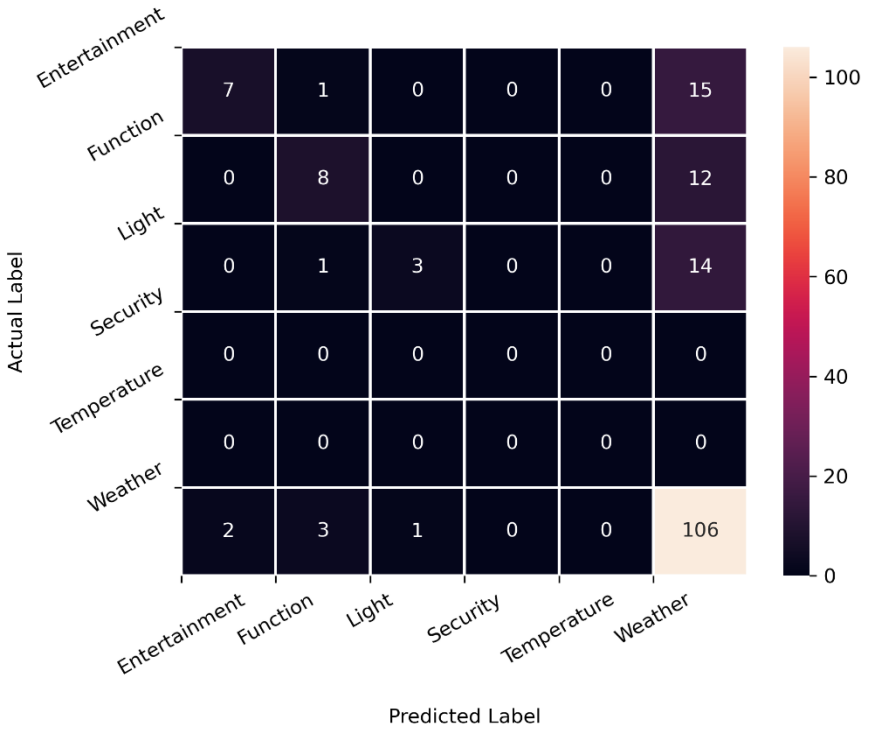

Figure 9: Confusion matrix for the Audio Defined Context Classifier given RAVDESS data set.

Figure 10 shows the confusion matrix for the Audio Defined Context Classifier's predictions over the CREMA data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with MEL and MFCC feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the CREMA data set, which consisted of 4,938 samples. The test data set contained 20% of the CREMA data set, which

consisted of 1,234 samples. The confusion matrix displayed below as Figure 10 shows that of the 1,234 test samples, the number of correctly predicted defined contexts are as follows: 59 for Entertainment, 181 for Function, 97 for Light, 209 for Security, 140 for Temperature, and 198 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 72% accuracy score.

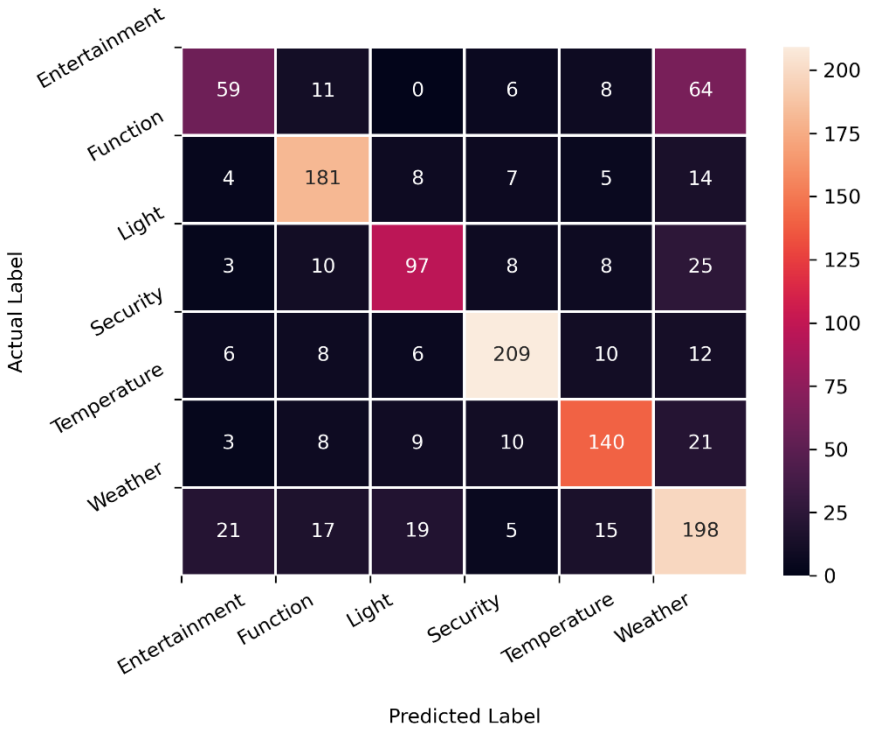

Figure 10: Confusion matrix for the Audio Defined Context Classifier given CREMA data set.

Figure 11 shows the confusion matrix for the Audio Defined Context Classifier's predictions over the SHVC data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with MEL and MFCC context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the SHVC data set, which

consisted of 4,000 samples. The test data set contained 20% of the SHVC data set, which consisted of 1,000 samples. The confusion matrix displayed below as Figure 11 shows that of the 1,000 test samples, the number of correctly predicted defined contexts are as follows: 91 for Entertainment, 62 for Function, 84 for Light, 51 for Security, 192 for Temperature, and 283 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 76% accuracy score.

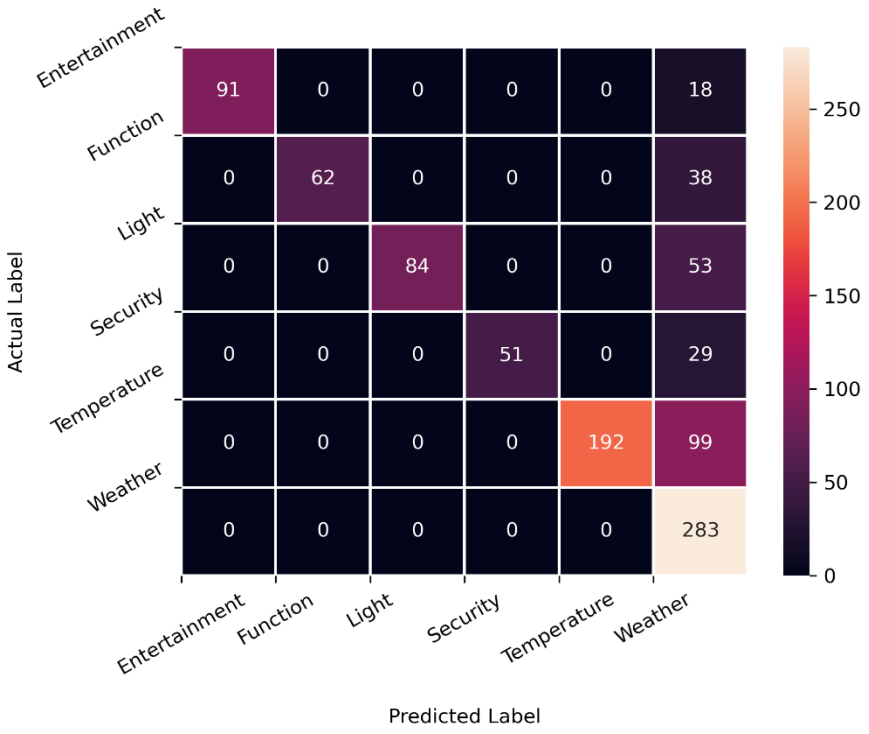

Figure 11: Confusion matrix for the Audio Defined Context Classifier given SHVC data set.

Figure 12 shows the confusion matrix for the Audio Defined Context Classifier's predictions over the combined data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with MEL and MFCC feature data, described in full detail in the previously presented Classifier

Implementation section. The training data set contained 80% of the combined data set, which consisted of 9,629 samples. The test data set contained 20% of the combined data set, which consisted of 2,407 samples. The confusion matrix displayed below as Figure 12 shows that of the 2,407 test samples, the number of correctly predicted defined contexts are as follows: 126 for Entertainment, 234 for Function, 141 for Light, 205 for Security, 341 for Temperature, and 521 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 65% accuracy score.

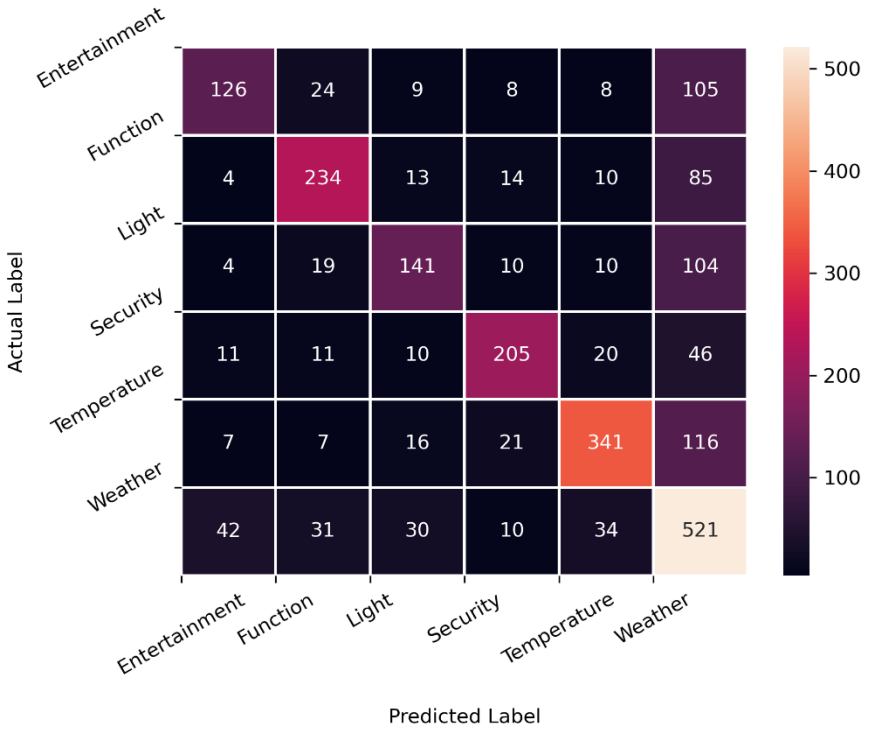

Figure 12: Confusion matrix for Audio Defined Context Classifier given Combined data set.

Final Defined Context Classifier. The basic defined context classifier produced accuracies of 70%, 69%, 67%, and 86% over the RAVDESS, CREMA, SHVC, and the combined

data set, respectively. The confusion matrix for this classifier over these data sets can be found below as Figure 13, Figure 14, Figure 15, and Figure 16.

Figure 13 shows the confusion matrix for the Final Defined Context Classifier's predictions over the RAVDESS data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with prediction feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the RAVDESS data set, which consisted of 692 samples. The test data set contained 20% of the RAVDESS data set, which consisted of 173 samples. The confusion matrix displayed below as Figure 13 shows that of the 173 test samples, correct predictions were 8 for Function and 3 for the Light defined context. There were also 108 correct predictions for the defined context Weather, thus the remaining samples were misclassified. This confusion matrix resulted in a 70% accuracy score.

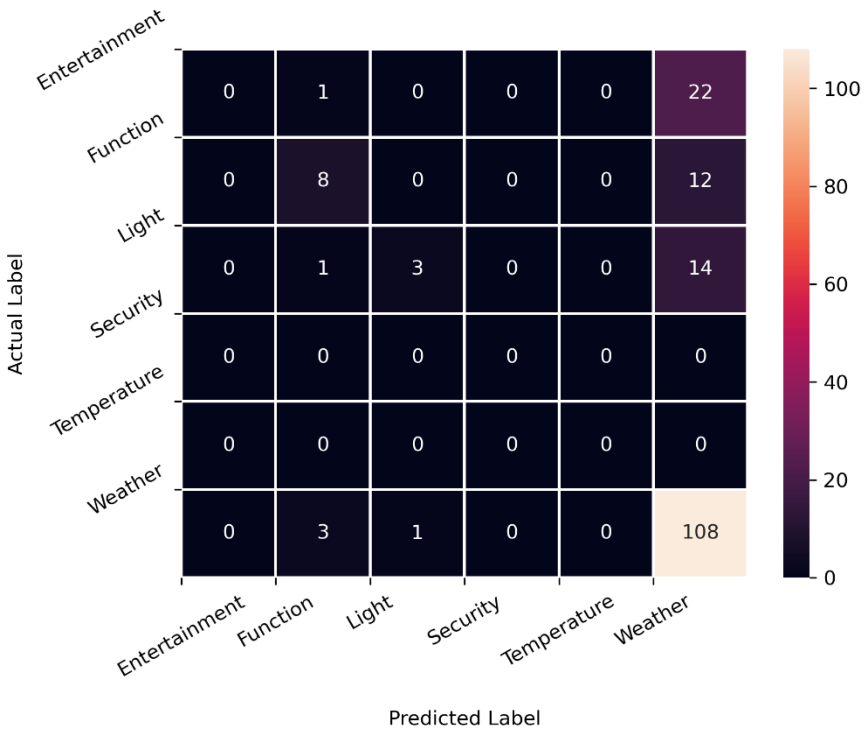

Figure 13: Confusion matrix for the Final Defined Context Classifier given RAVDESS data set.

Figure 14 shows the confusion matrix for the Final Defined Context Classifier's predictions over the CREMA data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with prediction feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the CREMA data set, which consisted of 4,938 samples. The test data set contained 20% of the CREMA data set, which consisted of 1,234 samples. The confusion matrix displayed below as Figure 14 shows that of the 1,234 test samples, the number of correctly predicted defined contexts are as follows: 181 for Function, 97 for Light, 209 for Security, 140 for Temperature, and 219 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 69% accuracy score.

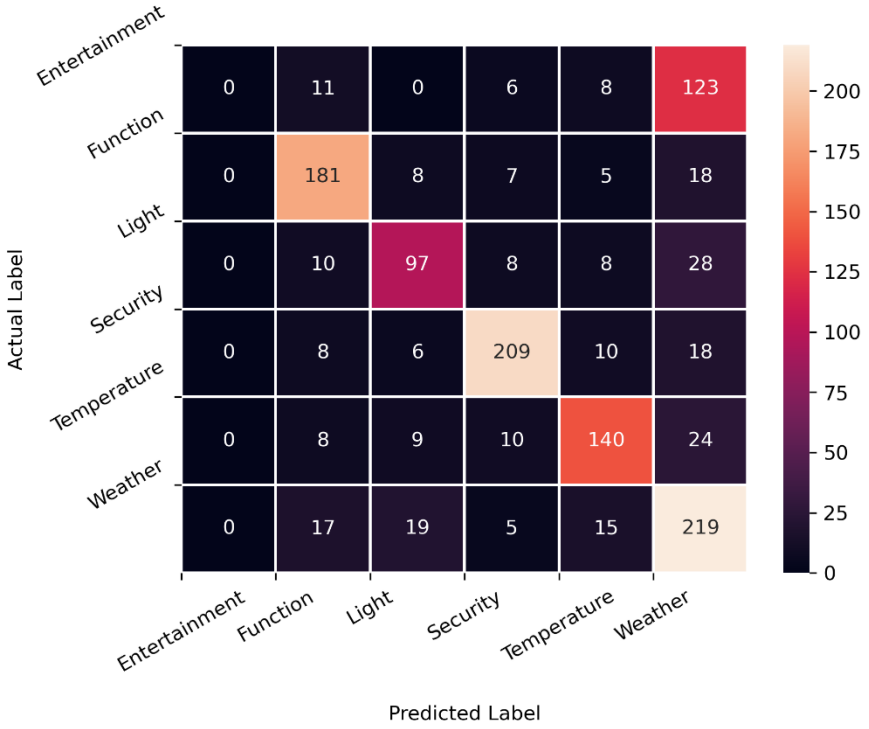

Figure 14: Confusion matrix for the Final Defined Context Classifier given CREMA data set.

Figure 15 shows the confusion matrix for the Final Defined Context Classifier's predictions over the SHVC data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with prediction context feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the SHVC data set, which consisted of 4,000 samples. The test data set contained 20% of the SHVC data set, which consisted of 1,000 samples. The confusion matrix displayed below as Figure 15 shows that of the 1,000 test samples, the number of correctly predicted defined contexts are as follows: 62 for Function, 84 for Light, 51 for Security, 192 for Temperature, and 283 for Weather. The

remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in a 67% accuracy score.

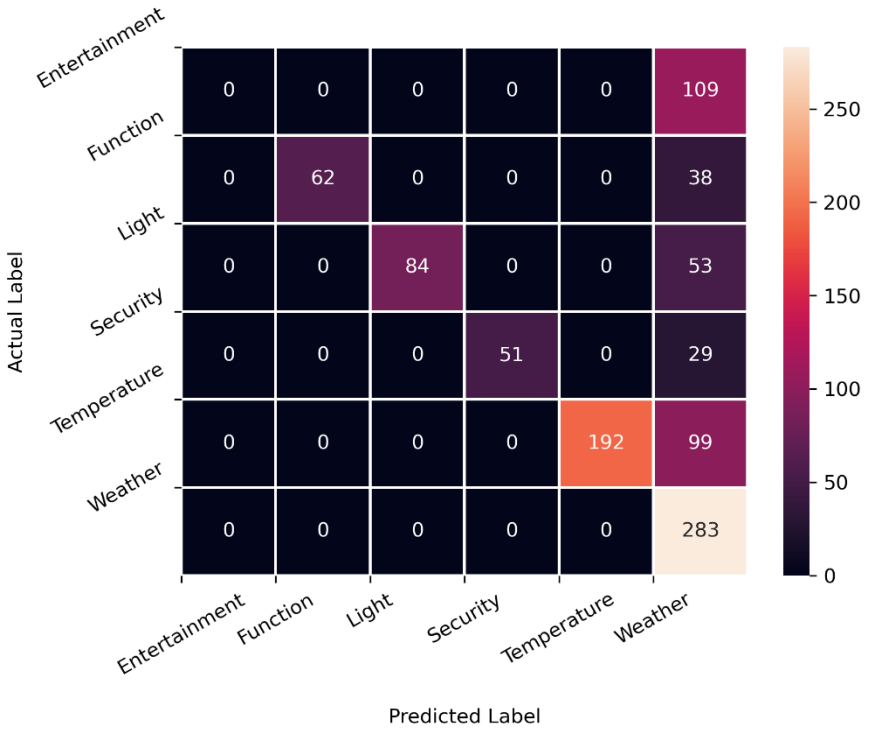

Figure 15: Confusion matrix for the Final Defined Context Classifier given SHVC data set.

Figure 16 shows the confusion matrix for the Final Defined Context Classifier's predictions over the combined data set. These predictions were obtained after training this TPOT AutoML model on the combined data set (described in the Data Sets section) with prediction feature data, described in full detail in the previously presented Classifier Implementation section. The training data set contained 80% of the combined data set, which consisted of 9,629 samples. The test data set contained 20% of the combined data set, which consisted of 2,407 samples. The confusion matrix displayed below as Figure 16 shows that of the 2,407 test samples, the number of correctly predicted defined contexts are as follows: 219

for Entertainment, 277 for Function, 227 for Light, 242 for Security, 410 for Temperature, and 529 for Weather. The remaining samples in the test data set were not correctly predicted, thus they were misclassified. This confusion matrix resulted in an 86% accuracy score.

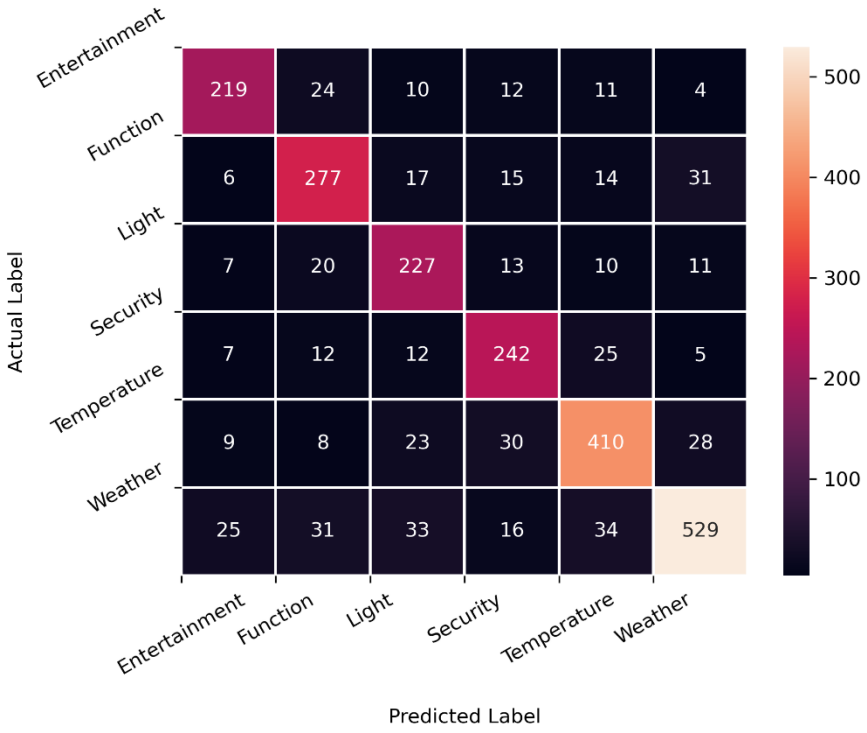

Figure 16: Confusion matrix for the Final Defined Context Classifier given Combined data set.

To assist interpretation of the data presented in the figures of this section, a summary of the data can be found in Table 4 below. The first column of this table provides four rows pertaining to the considered data sets which are RAVDESS, CREMA, SHVC, and the combined data set. The remaining columns each display the accuracy scores for a given defined context classifier (Basic, WERKS, Audio, and Final) per considered data set from the first column on a per row basis.

Each defined context classifier is modeled after an existing context detection methodology, except the WERKS and Final context detection methods, so Table 4 provides insight as to how they compare in terms of accuracy. To systematically measure and evaluate these context detection methods, I chose to utilize the standard "score" function that the TPOT AutoML model provides for every successfully trained classifier. This function can be reutilized for each successfully trained TPOT classifier and thus provides a high degree of reproducibility and systematic measurement. Equation 4, Equation 5, and Equation 6 found in the Accuracy section below are the equations that this "score" function utilizes to produce its accuracy measurements. The "score" function can be utilized with any successfully trained TPOT model. For example, if utilizing the python programming language with a TPOT model named as "classifier", feature data as "X", and target data as "y", then the following code snippet can be run to obtain the accuracy measurement:

$$classifier.score(X, y)$$

The Basic column displayed in Table 4 represents the basic context detection methodology, the WERKS column represents the WERKS context detection methodology, the Audio column represents an audio context detection methodology, and finally, the Final column represents my ensemble learning approach to context detection. The quantifiable benefits for these methodologies will now be discussed. The Basic defined context classifier provided an accuracy score of over 60% for both the RAVDESS and combined data sets; however, it provided accuracy scores under 30% on the CREMA and SHVC data sets. The WERKS context detection methodology provided very similar results to the Basic method, with accuracy scores over 60% for the RAVDESS and combined data sets, while the CREMA and SHVC data sets produced

accuracies under 30%. These accuracy scores for the Basic and WERKS methods provide insight that these methods are very similar, although the WERKS method does provide an increase of 6% for the combined data set, which is a statistically significant increase via chi-squared test with resultant p value as 4.9384e-26 (i.e. < 0.05).

The Audio context detection method provided an average accuracy of 71% over all considered data sets, with 76% being its highest accuracy, for the RAVDESSS data set. The lowest accuracy the Audio context detection method produced was 65%, over the combined data set. This means that for the combined data set (the data set that contains all data from RAVDESS, CREMA, and SHVC), the Audio method performed 4% better than the Basic method, however, the Audio method was not able to outperform the WERKS method. For the combined data set, the WERKS method provided 2% more accuracy than the Audio method, which is a statistically significant increase via chi-squared test with resultant p value as 1.3245e-86. Now, as for the Final context detection method (ensemble learning), Table 4 shows that it outperforms all other context detection methods for the combined data set by an average accuracy increase of 22%, thus, it appears to be the strongest context detection method for data consumption of at least 12,000 items. When consuming smaller amounts of data, it appears that the Audio context detection method could be preferred, as it provides an average of 73% accuracy over the RAVDESS, CREMA, and SHVC data sets. This contrasts with the 59%, 61%, and 69% average accuracies provided by the Basic, WERKS, and Final methods, respectively.

Table 4. Experimental results of context detection for various classification methodologies.

Data Set	Basic	WERKS	Audio	Final
RAVDESS	65%	65%	71%	70%
CREMA	22%	22%	72%	69%
SHVC	28%	28%	76%	67%
COMBINED	61%	67%	65%	86%

Sample Outputs

To address research question 2, as asked in the Research Questions section of the Introduction of this thesis, this section will provide sample outputs for the system. The sample outputs presented in this section were obtained from feature data extracted during emotion detection. The examples explored in this section are examples of context detection purposes for the given samples. The sample outputs for the system are presented in Table 5, Table 6, and Table 7 where the outputs are based on different stages of the initial approach given the RAVDESS, SHVC, and CREMA data sets. The different stages can be found in the third and fourth columns as Basic Contexts and WERKS Contexts, whereas the first column displays the input command, the second column displays the detected emotion, and the final column displays the defined context. The defined contexts are displayed as their numeric values which is how they are utilized within the system; however, the numeric values 0, 1, 2, 3, 4, and 5 correspond to the defined contexts as Security, Light, Temperature, Weather, Entertainment, and Function, respectively. These defined contexts are also discussed in further detail in the Defined Contexts section above.

The data in Table 5, Table 6, and Table 7 can be understood more easily if it is first understood why the defined contexts provide value to the system for a given short voice command audio data input. For example, when the audio data input for the system is “Kids are talking by the door” and the emotion detected by the emotion detection layer is ‘happy’, then the user might want the virtual assistant to automatically change the temperature to make the home more accommodating and comfortable for the guests. When the system is given the command “Dogs are sitting by the door” and the emotion detected was ‘sad’, then this may provide the virtual assistant with reason to open a doggy door so the user could allow the virtual assistant to let the dogs in instead of the user needing to open the door. If a user said to “play music”, the emotion of the user could be utilized to pick the genre of the music to play based on the user’s mood. When the system is asked “How’s the traffic in Springfield today” and the emotion is ‘fearful’, the user could be uneasy about the weather regarding their morning commute so the virtual assistant might provide the user with the weather along with route information for the traffic. If the user inquires with the system about the temperature outside being cold and they are ‘sad’, then the context detected by the system would be ‘temperature’, which would then allow the virtual assistant to automatically change the temperature of the user’s smart home to best meet their needs. When a user cancels alarms using their virtual assistant and they are ‘happy’, the virtual assistant could interpret this as the user is happy about cancelling the alarms which could then allow the system to detect the context as ‘entertainment’ and start playing some up-beat music or provide the user with some other form of entertainment to lighten their mood further. If the detected command is “Kids are talking by the door” while the user is ‘fearful’, then this could mean the virtual assistant

might need to prompt the user if they are safe. In the case when dogs are sitting by the user's door, then if they are angry this could mean they would like for the virtual assistant to do something to help remove the dogs from the doorway, such as turn on a bright light or play a thunder sound effect. The sample system outputs explored in this section were obtained from feature data extracted during emotion detection, thus, this feature data was repurposed for context detection purposes.

Table 5. Sample context outputs based on RAVDESS dataset.

Command	Emotion	Basic Context	WERKS Context	Defined Context
Kids are talking by the door	Happy	Kids talking	Blissful talking	2
Dogs are sitting by the door	Sad	Dogs sitting	Unworthy sitting	5
Kids are talking by the door	Fearful	Kids talking	Coward talking	0
Dogs are sitting by the door	Angry	Dogs sitting	Crabby door	1

Table 6. Sample context outputs based on SHVC dataset.

Command	Emotion	Basic Context	WERKS Context	Defined Context
Play music	Fearful	Play music	Mousey music	4
How's traffic today in Springfield	Fearful	Hows traffic	Apprehensive traffic	3
Is it cold today	Sad	Cold today	Gloomy today	2
Cancel all the alarms today	Happy	Cancel alarms	Overjoyed alarms	4

Table 7. Sample context outputs based on CREMA dataset.

Command	Emotion	Basic Context	WERKS Context	Defined Context
I'm on my way to the meeting.	angry	way meeting	argumentative meeting	3
I would like a new alarm clock.	happy	new alarm	decent alarm	4
Maybe tomorrow it will be cold	fearful	tomorrow cold	unnerving cold	3
The airplane is almost full.	fearful	almost full	coward full	0

Accuracy

To address research question 3, as asked in the Research Questions section of the Introduction of this thesis, this section will discuss the overall accuracy of the system and therefore provide insight as to the effectiveness of the system. The defined context prediction accuracy for the basic, WERKS, and audio defined context classifiers does not exceed 67%, as can be found in the previous figures in the Evaluation section such as Figure 7, Figure 8, Figure 12, and Figure 16; however, the WERKS and Final defined context classifiers did outperform the existing context detection methods given the combined data set by 6% and 21%, respectively. Thus, the incorporation of the user emotion into the context detection process does provide an increase to accuracy. To address the feasibility for directly integrating user emotion into the context detection process, the accuracy scores that each of these classifiers produced over the combined data set, described in the Data Sets section, was improved upon by 19% to equal 86% via the ensemble defined context classification layer seen in Figure 1. This was accomplished via the integration of the user emotion into the context detection process (described in the WERKS section previously discussed). The WERKS and Final defined context classifiers also provided statistically significant improvements compared to existing context detection methods presented in the Basic and Audio defined context classifiers, respectively (found in the Evaluation section above).

Further, in terms of practical application, the goal of the proposed approach was to provide an overall increase to context detection accuracy within the user's smart home environment using the methodology for context detection described in the Proposed Approach section. As described in the previous paragraph, the context detection methods introduced in

the Proposed Approach section did provide an overall (statistically significant) increase to context detection accuracy. The system was evaluated utilizing the basic bigram contexts, WERKS contexts, user audio data, and ensemble methodology. This meant that a focus on the accuracy measurement in which the number of defined contexts the TPOT classifier would correctly predict given these four context detection methodologies was applied. The accuracy score being utilized is known as the coefficient of determination and it is defined in Equation 4, Equation 5, and Equation 6 where 'u' is equivalent to the sum of squares of residuals and 'v' is equivalent to the total sum of squares:

$$u = \sum_{i=1}^{n} \left(y_i - f(x_i)\right)^2 \tag{4}$$

$$v = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{5}$$

$$R^2 = 1 - u/v \tag{6}$$

CONCLUSION

In this thesis, I hypothesized that extracting natural language feature data from user audio inputs to generate unique emotionally relevant context feature data would enhance the accuracy of classifying various predefined contexts in a context detection system for smart homes. To test this hypothesis, I proposed an approach to context detection that consisted of four major steps: 1. Emotion detection, 2. Basic context detection, 3. WERKS, and 4. Ensemble defined context classification. Step 1 used MFCC and MEL spectrogram feature data from user audio data to predict user emotion as one of the considered emotions – angry, fearful, happy, normal, or sad. Step 2 took the user input and performed audio data processing to "clean" it, which is then used to obtain a basic bigram context. The output from steps 1 and 2 are utilized in step 3, which consists of my emotion word bank synsets and WERKS methodology. The emotion word bank synsets are a total of four word banks that contain synonyms for the emotions angry, fearful, happy, and sad. The WERKS methodology used a combination of the emotion word banks, word embedding, and keyword search methods to produce a WERKS context by replacing the basic bigram context that was obtained from step 2. Finally, in step 4 an ensemble learning approach is utilized with a final defined context classifier to obtain a defined context from the predictions of three separate classifiers. The three classifiers utilized in step 4 of the proposed approach are the basic, WERKS, and audio defined context classifiers.

I evaluated my proposed approach over four audio file data sets – RAVDESS, SHVC, CREMA, and the combined data set. Each data set was consumed by my proposed approach to produce accuracy measurements. The defined context classifiers listed from least to greatest

prediction accuracy are as follows: basic, WERKS, audio, and final. Thus, I observed that the ensemble approach offered the best context detection accuracy.

In terms of limitations, my approach to context detection requires that user input be spoken in the English language. This means that if the system were to ever be utilized with a non-English speaking user, it would need to be trained on the language that the non-English user does speak. As for the severity of this limitation, it would likely be the case that the only loss for the system would be in terms of extra training time, as this would not increase the scope of the defined contexts. Thus, similar prediction accuracies should be expected as those already presented in this thesis. In addition, the current proposed approach makes no use of any sensors, nor sensor feature data, for context detection. In this thesis, I defined context as all the necessary information required to accurately assess a setting of a statement, event, or idea. The addition of sensors into the proposed approach would provide further information to accurately assess the setting of a user's environment. Thus, future work might be done to implement sensor feature data into the proposed approach for context detection to enhance performance. Further, the system is also reliant upon the user input to be said in a restricted number of emotions. The emotions considered in this thesis were angry, fearful, happy, normal, and sad. For the system to consider further emotions, additional word bank synsets would be required to be created and the emotion detection layer to be retrained. This work could be explored in the future, but as the considered emotions were chosen based on current research such as in [28], I would suggest waiting until further developments were discovered that merited the additional emotions.

Regarding practical applications, the emotionally relevant contextual data produced by the proposed approach may be interpreted through a virtual assistant to benefit the smart home. This would benefit the smart home with respect to providing a better user experience by automating smart home actions. The proposed approach could be implemented into existing virtual assistant technologies, which are exemplified by services such as Amazon's Alexa, Google Assistant, and Apple's Siri. Specifically, these services could utilize the proposed context detection methodology to assist in smart home action automation, thus improving upon the user experience. For example, if a user said "Kids are talking by the door" and they were feeling 'happy', then these services could utilize my proposed approach as part of an automated action process to change the temperature in the smart home, thus making it more accommodating for guests. Therefore, the novelty of my proposed approach to context detection lies not only in its attention to both the context and emotional state of the user, but also in its ability to address the ambiguity of the user needs by obtaining the emotionally relevant context. In the future, development could be done to incorporate emotion ensemble into my proposed approach. Emotion ensemble is a method of emotion detection in which for a longer dialogue, multiple emotions are detected and used as an accumulative prediction to one emotion. It would provide robustness and likely increase the accuracy of the emotion detection layer for longer audio inputs and conversations.

REFERENCES

[1] International Data Corporation, "Worldwide Spending on the Internet of Things is Forecast to Surpass $1 Trillion in 2026, According to a New IDC Spending Guide," [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS50936423. [Accessed 12 March 2024].

[2] IoT Analytics , "IoT connections market update—May 2023," [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/. [Accessed 12 March 2024].

[3] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal and S. W. Kim, "Multimedia Internet of Things: A Comprehensive Survey," *IEEE Access,* vol. 8, pp. 8202-8250, 2020.

[4] D. Fedotov, Y. Matsuda and W. Minker, "From Smart to Personal Environment: Integrating Emotion Recognition into Smart Houses," *IEEE Intl. Conf. on Pervasive Computing and Communications Workshops,* pp. 943-948, 2019.

[5] A. Patel and T. A. Champaneria, "Fuzzy logic based algorithm for Context Awareness in IoT for Smart home environment," in *IEEE Region 10 Conference*, 2016.

[6] C. H. Papadimitriou, H. Tamaki, P. Raghavan and S. Vempala, "Latent Semantic Indexing: A Probabilistic Analysis," in *Proceedings of the Seventeenth*

ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Association for Computing Machinery, pp. 159-168, 1998.

[7] R. Kosti, J. M. Alvarez, A. Recasens and A. Lapedriza, "Context based emotion recognition using emotic dataset," *IEEE transactions on pattern analysis and machine intelligence,* vol. 42, no. 11, pp. 2755--2766, 2019.

[8] N. Zargham, M. Bonfert, R. Porzel, R. Malaka and T. Dring, "Multi-Agent Voice Assistants: An Investigation of User Experience," 2021.

[9] C. Wilson, T. Hargreaves and R. Hauxwell-Baldwin, "Smart homes and their users: a systematic analysis and key challenges," *Personal and Ubiquitous Computing,* vol. 19, pp. 463--476, 2015.

[10] V. Claessen, A. Schmidt and T. Heck, Virtual Assistants, Humboldt-Universität zu Berlin, 2017.

[11] A. Chatterjee, K. N. Narahari, M. Joshi and P. Agrawal, "SemEval-2019 task 3: EmoContext contextual emotion detection in text," in *Proceedings of the 13th international workshop on semantic evaluation*, 2019.

[12] A. K. Sikder, L. Babun, H. Aksu and A. S. Uluagac, "Aegis: A context-aware security framework for smart home systems," *Proc. of Annual Computer Security Applications Conference,* pp. 28-41, 2019.

[13] W.-H. Cheng, W.-T. Chu and J.-L. Wu, "Semantic context detection based on hierarchical audio models," *Proc. of ACM SIGMM Intl. Workshop on Multimedia Information Retrieval,* pp. 109--115, 2003.

[14] W.-T. Chu, W.-H. Cheng, J.-L. Wu and J. Yung-jen Hsu, "A study of semantic context detection by using SVM and GMM approaches," *IEEE Intl. Conf. on Multimedia and Expo,* vol. 3, pp. 1591-1594, 2004.

[15] W. Dargie, "Adaptive Audio-Based Context Recognition," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans,* vol. 39, no. 4, pp. 715-725, 2009.

[16] M. M. Rana, M. T. Sultan, M. Mridha, M. E. A. Khan, M. M. Ahmed and M. A. Hamid, "Detection and correction of real-word errors in Bangla language," *IEEE Intl. Conf. on Bangla Speech and Language Processing,* pp. 1-4, 2018.

[17] M. Pal and R. Prasad, "Sarcasm Detection followed by Sentiment Analysis for Bengali Language: Neural Network & Supervised Approach," pp. 1-7, 2023.

[18] V. K. Patil, O. Hadawale, V. R. Pawar and M. Gijre, "Emotion Linked AIoT Based Cognitive Home Automation System with Sensovisual Method," *IEEE Pune Section Intl. Conf. (PuneCon),* pp. 1-7, 2021.

[19] L. Hardesty, "Explained: Neural networks," MIT News | Massachusetts Institute of Technology, 2017. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414. [Accessed 4 July 2023].

[20] D. Patil, R. Lokare and S. Patil, "An Overview of Text Representation Techniques in Text Classification using Deep Learning Models," in *2022 3rd International Conference for Emerging Technology (INCET)*, pp. 1-4, 2022.

[21] W. Luo, "Research and Implementation of Text Topic Classification Based on Text CNN," in *2022 3rd International Conference on Computer Vision, Image and Deep Learning and International Conference on Computer Engineering and Applications (CVIDL and ICCEA)*, pp. 1152-1155, 2022.

[22] D. Zhang and X. Liu, "Text classification of DGCNN model based on deep learning," in *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pp. 622-625, 2021.

[23] E. Roberts, "NLP - Overview," Stanford University, 2004. [Online]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/nlp/overview_history.html. [Accessed 12 March 2024].

[24] A. Z. Smola, Z. C. Lipton, M. Li and A. J., "Dive into Deep Learning," *CoRR,* vol. abs/2106.11342, 2021.

[25] C. H. Papadimitriou, H. Tamaki, P. Raghavan and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 1998.

[26] R. Murakami and B. Chakraborty, "Neural Topic Models for Short Text Using Pretrained Word Embeddings and Its Application To Real Data," *IEEE Intl. Conf. on Knowledge Innovation and Invention,* pp. 146-150, 2021.

[27] D. Yamunathangam, C. Priya, G. Shobana and L. Latha, "An Overview of Topic Representation and Topic Modelling Methods for Short Texts and Long

Corpus," in *2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, pp. 1-6, 2021.

[28] S. Guha and R. Iqbal, "DESCo: Detecting Emotions from Smart Commands," *IEEE Annual Computers, Software, and Applications Conference,* pp. 1620-1625, 2022.

[29] S. R. Livingstone and F. A. Russo, "The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English," *PloS one,* vol. 13, no. 5, p. e0196391, 2018.

[30] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova and R. Verma, "Crema-d: Crowd-sourced emotional multimodal actors dataset," *IEEE transactions on affective computing,* vol. 5, pp. 377--390, 2014.

[31] Merriam-Webster, "Happy," 2024. [Online]. Available: https://www.merriam-webster.com/thesaurus/happy. [Accessed 12 March 2024].

[32] Merriam-Webster, "Angry," 2024. [Online]. Available: https://www.merriam-webster.com/thesaurus/angry. [Accessed 12 March 2024].

[33] Merriam-Webster, "Sad," 2024. [Online]. Available: https://www.merriam-webster.com/thesaurus/sad. [Accessed 12 March 2024].

[34] Merriam-Webster, "Fearful," 2024. [Online]. Available: https://www.merriam-webster.com/thesaurus/fearful. [Accessed 12 March 2024].

[35] Y. Zheng, Y. Shi, K. Guo, W. Li and L. Zhu, "Enhanced word embedding with multiple prototypes," *Proc. of Intl. Conf. on Industrial Economics System and Industrial Security Engineering,* pp. 1-5, 2017.

[36] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access,* vol. 10, pp. 99129-99149, 2022.

[37] F. Hutter, J. Lücke and L. Schmidt-Thieme, "Beyond manual tuning of hyperparameters," *KI-Künstliche Intelligenz,* vol. 29, pp. 329--337, 2015.

[38] R. S. Olson, N. Bartley, R. J. Urbanowicz and J. H. Moore, "Evaluation of a tree-based pipeline optimization tool for automating data science," in *Proceedings of the genetic and evolutionary computation conference 2016*, 2016.

[39] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis,* pp. 367--378, 2002.

[40] G. E. Hinton, "Connectionist Learning Procedures," *Artificial Intelligence,* vol. 40, pp. 185--234, 1989.

[41] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016.

[42] F. Dellaert, P. Thomas and A. Waibel, "Recognizing emotion in speech," *Proc. of Intl. Conf. on Spoken Language Processing,* vol. 3, pp. 1970--1973, 1996.

APPENDICES

Appendix A: Stop Words

Stop Words			
just	before	our	yourself
between	because	can	who
wasn't	from	ma	theirs
so	about	itself	did
if	why	you'll	him
that	most	hadn't	against
yours	ve	mightn	while
more	don't	down	isn't
needn't	having	had	ain
they	of	aren	and
this	its	doesn't	we
these	my	didn	to
haven	he	hadn	than
herself	during	you're	she's
mustn	should	them	with
such	an	other	was
been	a	but	once
it	some	shan't	her

Stop Words			
any	re	by	shouldn't
what	whom	hasn	needn
me	hers	you've	have
over	d	you'd	being
both	should've	don	through
ll	it's	couldn	wasn
wouldn	out	there	all
i	you	are	their
few	couldn't	ourselves	do
that'll	each	further	myself
himself	when	weren	where
now	be	shan	no
own	shouldn	how	below
not	yourselves	has	themselves
does	here	above	y
very	or	mightn't	she
into	then	were	nor
haven't	didn't	isn	after
at	hasn't	too	is
off	ours	the	doesn
weren't	s	your	doing

Stop Words			
aren't	am	up	which
only	o	t	until
on	again	m	those
wouldn't	won	mustn't	as
won't	his	same	for
in	will	under	

Appendix B: Emotion Word Bank Synsets

Appendix B-1. Happy synsets in happy emotion word bank.

Happy Synonyms			
able	convenient	golden	privileged
absorbed	correct	good	profitable
acceptable	crazy	gratified	promising
accidental	decent	halcyon	proper
adequate	decorous	harmonious	propitious
advantageous	delighted	heartening	providential
anxious	deserved	hopeful	qualified
appeased	distracted	hot	queer
applicable	dotty	impassioned	rapt
appropriate	ecstatic	intoxicated	rapturous
apt	elated	involved	rejoicing
ardent	encouraging	jocund	required
auspicious	engaged	jolly	requisite
balanced	engrossed	jovial	respectable
beaming	enraptured	joyful	rhapsodic
beatific	entranced	joyous	rhapsodical
becoming	euphoric	jubilant	right
befitting	exhilarated	just	rightful
beneficial	exuberant	justified	rosy

Happy Synonyms			
benign	exultant	kosher	sanguine
blessed	fair	laughing	satisfactory
blest	favorable	lighthearted	satisfied
blissful	favored	lucky	seasonable
blithe	felicitous	meet	seemly
blithesome	fervent	merry	serendipitous
bright	fervid	mirthful	serviceable
buoyant	feverish	mollified	silly
capable	fit	needed	smiling
chance	fitted	nuts	suitable
cheerful	fitting	obsessed	sunny
cheery	flukey	occupied	thankful
chuffed	fluky	opportune	thrilled
coincidental	foolish	optimistic	tickled
companionate	fortuitous	overjoyed	timely
competent	fortunate	pacified	tolerable
concerned	full	passionate	trained
condign	gay	placated	unexpected
congruous	gifted	pleased	unforeseen
consonant	glad	preoccupied	upbeat
content	gladsome	prepossessed	contented

Appendix B-2. Angry synsets in angry emotion word bank.

Angry Synonyms			
acrid	cranky	infuriate	riley
acrimonious	cross	infuriated	roiled
aggravated	disagreeable	inimical	seething
angered	disapproving	irascible	shirty
annoyed	disputatious	irate	sizzling
antagonistic	distant	ireful	smoldering
antipathetic	dyspeptic	irritable	smoldering
antisocial	embittered	livid	snappish
apoplectic	enflamed	mad	sore
argumentative	enraged	malevolent	sorehead
ballistic	exasperated	ornery	soreheaded
bearish	foaming	outraged	spiteful
belligerent	fretful	passionate	steaming
bilious	frigid	peevish	stormy
bitter	fuming	perturbed	sulky
boiling	furious	petulant	testy
bristling	fussy	piqued	ticked
bristly	grouchy	pugnacious	touchy
burning	grumpy	quarrelsome	unfriendly
cantankerous	hopping	querulous	unpleasant

Angry Synonyms			
choleric	hot	rabid	vengeful
churlish	huffy	rancorous	vindictive
cold	icy	rankled	virulent
contentious	incensed	ranting	vitriolic
contrary	indignant	raving	wrathful
cool	inflamed	resentful	wroth
crabby	inflammable	riled	wrought

Appendix B-3**.** Sad synsets in sad emotion word bank.

Sad Synonyms			
abhorrent	disgusting	heartrending	regretful
abominable	disheartened	heartsick	rueful
affecting	disheartening	heartsore	saddened
aggrieved	dishonorable	heavyhearted	saddening
agonized	dismal	hopeless	saturnine
anguished	dispirited	ignominious	scandalous
bad	dispiriting	inconsolable	shameful
beastly	disquieting	infamous	shocking
black	disreputable	inferior	somber
bleak	distressed	joyless	sombre
blue	distressful	lachrymose	sordid

Sad Synonyms			
brokenhearted	distressing	lame	sorrowful
cheerless	disturbing	lamentable	sorry
comfortless	doleful	lousy	stinking
contemptible	dolorous	low	suicidal
crestfallen	down	lugubrious	sullen
dark	downcast	melancholic	sunk
darkening	downhearted	melancholy	tearful
dejected	drear	meritless	teary
deplorable	dreary	misbegotten	touching
depressed	droopy	miserable	troubled
depressing	elegiac	morbid	uneasy
desolate	elegiacal	morose	unfortunate
despairing	forlorn	mournful	unhappy
despicable	funereal	moving	unquiet
despondent	gloomy	murky	unsavory
detestable	glum	notorious	unworthy
disappointed	gray	odious	upset
discomforting	grey	pathetic	wailing
discomposing	grieving	perturbing	weeping
disconsolate	grievous	pitiable	woebegone
discouraged	hangdog	pitiful	woeful

Sad Synonyms			
discouraging	hateful	plaintive	worried
discreditable	heartbreaking	poignant	worthless
disgraceful	heartbroken	poor	wretched

Appendix B-4. Fearful synsets in fearful emotion word bank.

Fearful Synonyms			
accentuated	dismayed	hellacious	shocked
acute	dismaying	hideous	shocking
affrighted	disquieted	horrendous	shrinking
afraid	disquieting	horrible	shy
aggravated	distressing	horrid	skittish
aghast	disturbed	horrified	spineless
agitated	disturbing	horrifying	spooked
alarmed	dread	hysteric	spooky
alarming	dreadful	hysterical	startled
almighty	eerie	intense	startling
anxious	eery	intensified	stressed
appalled	emphasized	intensive	terrible
appalling	enhanced	intimidated	terrified
apprehensive	excruciating	intimidating	terrifying
atrocious	exhaustive	jittery	terrorized

Fearful Synonyms			
awful	explosive	jumpy	thorough
blistering	exquisite	keen	threatening
careful	fainthearted	macabre	timid
cautious	fearsome	magnified	timorous
chicken	ferocious	monstrous	tremulant
chickenhearted	fierce	mousey	tremulous
concentrated	forbidding	mousy	troubling
coward	formidable	nervous	trying
cowardly	frightened	nightmarish	unadventurous
cowed	frightening	panicked	uneasy
craven	frightful	panicky	unheroic
creepy	funky	perturbed	unnerved
dastardly	furious	perturbing	unnerving
daunted	ghastly	phobic	upset
daunting	grewsome	poltroon	vehement
deep	grisly	profound	vicious
deepened	gruesome	prudent	violent
demoralizing	gutless	pusillanimous	wary
dire	hard	redoubtable	weird
direful	harsh	rigorous	worried