Open Speech Command Translator System for Executing Positive, Negative, and Neutral Decisions

Oscar García Bruckman
Electrical Engineering
Luis Vicente, Ph. D
Electrical and Computer Engineering & Computer Science Department
Polytechnic University of Puerto Rico

Abstract *— This research proposes a system that uses an open speech command and translates it into a Positive, Negative, or Neutral decision. An ordinary voice command targeted to an object is generally composed of two parts: the object identification, and the desired instruction to execute from it. Using this structure, several voice features could be analyzed such as pitch, intensity, formants, among other statistical characteristics to recognize if a pattern exists. For this research, the "object" segment was ideally converted to text using existing speech-to-text tools; on the other hand, the "instruction" segment was ideally processed by applying a Naïve Bayes Classifier to classify instructions as Positive, Negative, or Neutral with up to 72% accuracy. This idea could be used in external systems for people with special-needs, among other yet unimagined applications. The system will enable improved natural interaction between humans and computers by using a basic but effective open communication.*

Key Terms *— Command, Decision, Language, Translator*

INTRODUCTION

The objective of this research is to design and propose a system that uses a flexible and open speech command as input, analyze its emotions and characteristics, and execute an output of a desired signal for the use of other existing external systems, based on the research results published in [1]. The criteria or parameter to decide what type of output to execute is determined on the emotion prosodic and acoustic features of the instruction signal segment. The extracted characteristics will ideally indicate if the instruction of an input command is naturally classified as a positive, negative, or neutral instruction. With this information, the system will declare and select a predefined output according to the decision made.

There are several theoretical emotion classifier systems that exist, however their operation with natural speech commands has always been a challenge since individual emotion classification is not yet a concrete science, even for humans [2]. This article will explain what techniques could be implemented, according to the research performed, in the design of the Open Speech Command Translator (OSCT) System, and will also explain and decide, what set of features can affect and may be determinant, in a Positive, Negative, or Neutral decision using a simple Naïve Bayes Classifier. Of course, the accuracy of the system will be dependable on the effectiveness of the features, projection techniques, and training data selected for this classifying method. Therefore, the conclusion of this investigation will give light to what set of features and components should help in the creation of an Open Speech Command Translator system.

BACKGROUND

The concept of a human communicating a command to a living being, which in turn, receives an uncertain but yet understandable instruction, and using its past experiences, executes an action, is something that many pet-owners encounter every day. For example, if a dog is on top of the owner's bed and the owner wants to order him out, a speech command like "<pet-name>, get down" should work on the dog, but also, there are many simple derivatives of this command that would work as well, like "<pet-name>, please get down now". This is possible since the dog is not literally interpreting what the owner is saying but (In a very

simplistic way) basing its decision on its experience using peculiar acoustic characteristics of the owner's voice. This relatively simple processing behavior, however, executes the desired action in an instant. This same behavior could be used to an advantage in scenarios where a basic action is required to be executed by a flexible speech command, but the processing resources are a constraint.

There are very sophisticated systems that convert all the spoken words to text, such as [3], records several important characteristics including the word's multiple relationships with other words, analyzes connections, and possible sentence context, using at the same time complex data structures and data mining algorithms, among other things, to then finally, make a decision. Clearly, this seems like an overshot in scenarios where, for example, you just need to make a special device active or not depending on a flexible speech command input. Other similar systems don't rely in using speech-to-text technology, but in turn rely in voice signal characteristics, statistic features, and algorithms [4]; a concept which is similar to the OSCT, with the exception that these systems have multiple emotional classes as output and require more training resources. This type of emotion classifier is probably more thorough on its speech analysis since it needs to provide multiple emotion classes as output. However, in many cases this amount of computational resources is not an option for the special scenarios previously discussed.

INITIAL RESEARCH

There are certain systems that are by default designed for very specific uses: Lifts, alarms, sensors, triggers, doors, lights, etc.; which come with a limited number of actions, in this case, 3 possible actions ("Do something", "Do the opposite", "Do nothing"). For this research, these will be the target applications for designing the OSCT system.

As the basis for my investigation, there are several sources that provide different experimented strategies to recognize patterns and use special features for describing basic emotions on a speech signal [1][4][5][6]. With the techniques described by Mohammed E. Hoque, et al. [1] in their research, and the high accuracy rates reached by these techniques, an efficient and accurate OSCT system can be built for "Translating" specific speech command signals to positive, negative, and neutral output decisions. The specific features that the OSCT system will incorporate will be described later on this article.

As part of the initial preparation to begin this research, multiple test-case speech commands were recorded from an external and unbiased person (Gender: female). This was done in order to build a corpus to calculate features from, which could feed a training data matrix for the classifier. These recordings were performed with Spanish commands in a relaxed environment with relative low-noise conditions. The commands were each recorded into a ".wav" file using a single channel input, a sampling rate of 8000Hz, and a sample format of 16-bit per sample PCM (Pulse Code Modulation) [7][8].

After analyzing the exact time interval that took each of the voice commands to complete, the commands were observed to be completed naturally in approximately 3 seconds or less (See Figure 1 to Figure 4). Using this observation, the time length for the input commands will be established to 4 seconds, as part of one of the OSCT system's rule. The voice command clips will be programmatically trimmed at the end of the 4 seconds to safely record the wanted voice command and limit the ending silences on the signal. Given this time length and sampling rate (8000Hz), the input discrete signals will have an exact total of 32,000 samples describing the signal. This duration (4 seconds) will be selected as a standard length for the input command signal.

As for the content in the input voice command simulations, the examined user was only informed to execute speech commands directed to several

particular objects, and devices that could theoretically listen to these commands, and no further detail of the investigation or its goal was explained to the test subject. This was done to create natural unbiased samples of different speech commands targeted to particular objects. A "Positive" command was recorded to stimulate an active reaction on objects such as windows, doors, gates, bathroom, house, alarms, and also possible domestic appliances like coffee-maker, refrigerator, stove, lights, and among several others. Also, a variety of opposite commands aimed to execute an off/deactivate related action were recorded to simulate what the output of the "Negative" decision could be.

After visually analyzing the different signal plots, two prominent characteristics were observed among all speech command tests. The speech commands were composed generally by two segments, the "object's identification" segment and a variable "instruction" segment. Also, the pause between the <object name> and the <instruction> segment was the longest pause found in the signal. This was a predicted behavior which can also be observed for a speech command targeted, for example, to the dog previously mentioned. A person tends to call the pet's name and pauses naturally for a very brief instant (To get his attention) before emitting the command. The plots in Figure 1, Figure 2, Figure 3, and Figure 4 illustrate a few speech command examples and confirm how this same behavior existed when the commands were directed to "listening" objects, and devices:

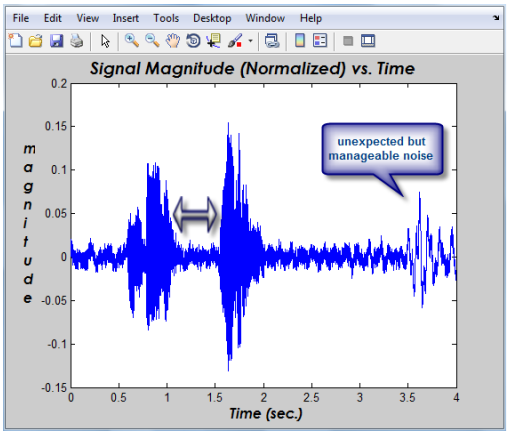

Figure 1
"Ventanas, abran" (Windows, open up)

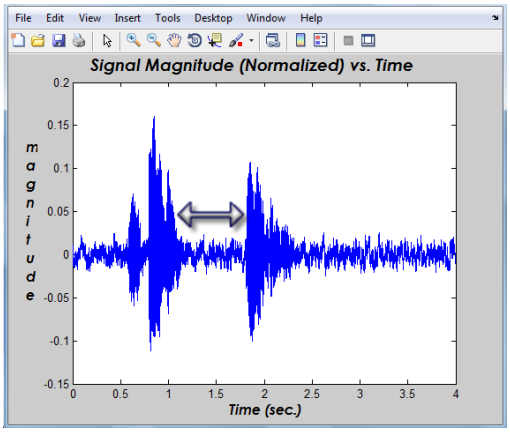

Figure 2
"Ventanas, cierren" (Windows, close)

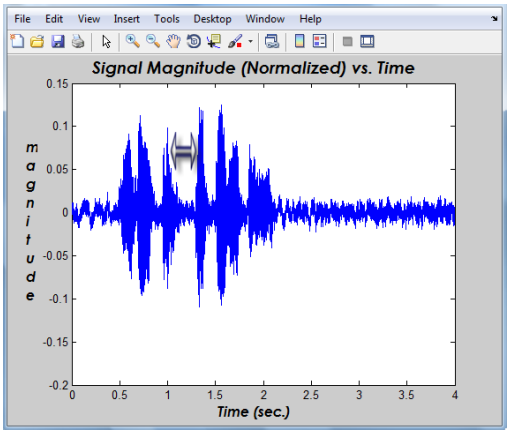

Figure 3
"Noticias, actualízame" (News, update me)

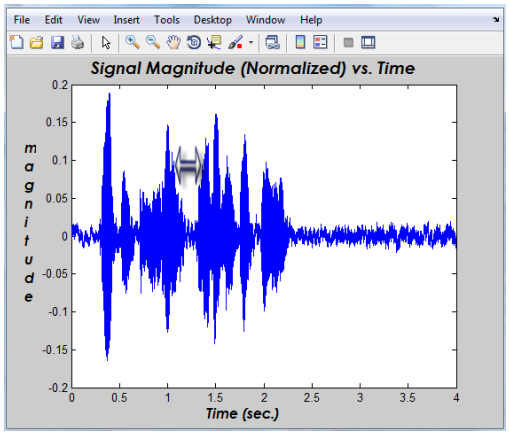

Figure 4
"Cafetera, cuélame café" (Coffee-maker, make me some coffee)

As can be observed on these plots, a significant pause separates the <object name> from the <instruction>. This important pause will help to determine how to "cut" the signal in the right places in order to send the <object name> segment to the speech-to-text system, and the <instruction> segment to the "Positive, Negative, or Neutral" classifier. In order to detect this special pause and distinguish it from the rest, a special Pause Detector was designed just for this purpose.

To implement this special Pause Detector, first, the input signal must be segmented into equal-length time-frames across the signal. Then, the average power will be calculated for each frame, resulting in a pre-determined number of points representing the average power plot of the signal. Since the signal is really a discrete signal composed of 32,000 samples, to obtain the average power of each frame, the following steps were performed; For each frame, the squared magnitude of each of its samples needed to be obtained (1), to obtain the sample's energy value. These values were then added to obtain the frame's energy sum, E_n (2), where k is the sample instance inside the frame, n, and K_n is the total instances in the frame. After obtaining E_n, it's then divided by the time-frame, Δt, in order to finally obtain the average power value P_{avg_n} of the frame (3).

The total time for the time-frame is an input parameter for the detector and it will define how much definition is used for calculating the power averages. Located at Figure 5, is a power signal of the input signal illustrated in Figure 2. Specifically, Figure 5 is using 40ms time-frames, which in 4sec. equals to 100 frames or 100 average power data points. The plot at Figure 6, displays the physical difference of the same signal when the length of the time-frame is changed to 10ms, equaling 400 total average-power data points.

$$E_k = S_k^2 \tag{1}$$

$$E_n = \sum_{k=1}^{K_n} E_k \tag{2}$$

$$P_{avg_n} = \frac{E_n}{\Delta t} \tag{3}$$

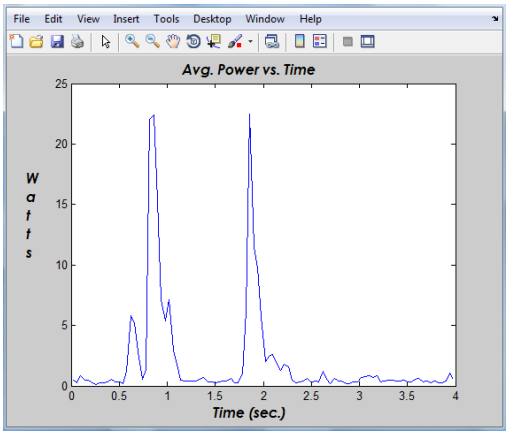

Figure 5
40ms Frames, 100 Points Average-Power Data

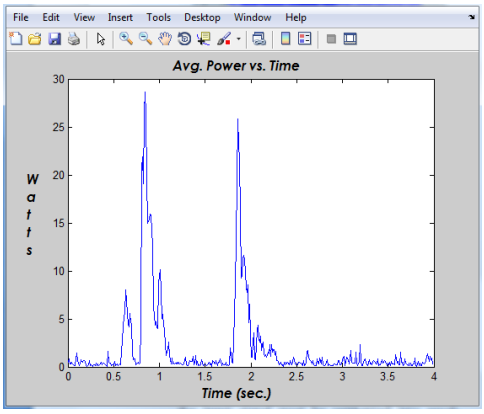

Figure 6
10ms Frames, 400 Points Average-Power Data

According to several research sources, such as [9], a commonly used time-frame to execute these calculations is between 10ms and 20ms, however this may vary depending on the application. After making several performance tests and experimenting with different values, I validated that the 10ms time-frame is indeed adequate, and therefore selected this time-frame for this research. These 10ms time-frames (Or 400 average-power data points) will be selected as a standard for all tests. Using these average-power data points we can establish a basic, but effective, threshold by calculating the straight-average value, Avg_{N_n}, from this dataset (4) (Where N_n is the total data points, 400). After applying this threshold to all the test signals, this value was observed to be an excellent approximation of the ideal threshold. This is mainly due to the fact that the total number of power data points of silent noisy frames is significantly greater than the total number of data points of actual voiced frames; therefore, this straight-average calculation is more influenced by these low power values, making the threshold nearly ideal for classifying pauses.

$$Avg_{N_n} = \frac{1}{N_n}\sum_{n=1}^{N_n} P_{avg_n} \quad (4)$$

This simple straight-average value was selected as a standard threshold for our pause detector design. The method selected satisfies the scope for this research, however, different threshold selection methods can be used to provide even better quality to the pause detection. In Figure 7's illustration, the threshold established is plotted over the power signal from Figure 6. After selecting the threshold, the data points were evaluated in order to determine which ones were below this threshold. In order for a set of data points to be considered a true pause, the pause is required to have a beginning transition, where data points crossed the threshold from above, and an ending transition, where data points crossed the threshold from below. After visually examining all significant pause durations on the test signals, these were observed to be well below 1 second on all instances. This observation was applied into the design in order to filter out the significant-pause candidates. To be a pause candidate, the maximum duration of a true pause must not be more than 1. This is done to discard extremely long pauses that will, most likely, be unwanted silences or accidental noisy data points located at the near end of the command (See Figure 1). The longest pause among these gathered candidates is the one selected officially as the "Significant Pause".

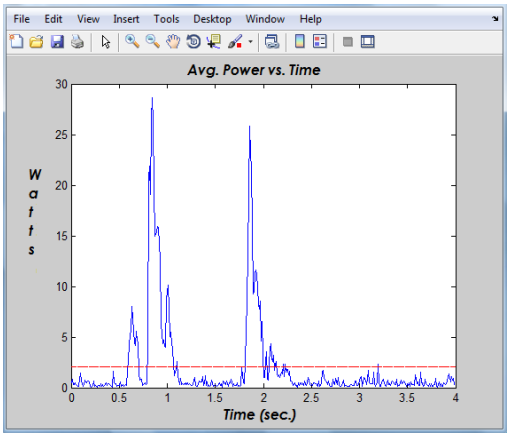

Figure 7
Threshold Marked over the Average Power Plot of Figure 6

As you can carefully observe in Figure 7, there are noise data points located barely above the threshold, between the seconds ~2.23 and ~3.19. These types of pauses meet all the previously mentioned requirements; however, they are, of course, not a valid pause. This problem repeated itself in some of the test signals, causing the pause detector to select incorrectly the significant pause in 5 out of 20 instances. However, after analyzing the real-time frequency spectrum of the original signals (Using [10]), it was observed that these silent/noisy power values were more concentrated in the low frequencies (Figure 8). Due to this observation, a Two-point Difference Filter (A high-pass filter) [11] was applied (5) to the original discrete signal, $s[k]$, in order to obtain the signal with low frequency components damped, $y[k]$, with the purpose of improving the pause detection quality.

$$y[k] = (s[k] - s[k-1])/2 \quad (5)$$

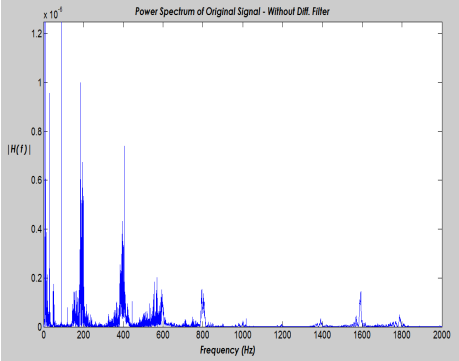

Figure 8
Frequency Spectrum without Differential Filter

After running our pause detector, the improvement was significant. The previous success rate of the Pause Detector before using the filter was 75% (5 misdetections out of 20 signals), but after applying the high-pass filter (See Figure 9 and Figure 10), the success rate of the Pause Detector was ~100% (20/20). An example of one of the successful detections of the "Significant pause" can be observed throughout Figure 11, Figure 12, Figure 13, and Figure 14. The filter selected so far has proven to provide accurate results; however, different filters can be used to provide even better quality to the pause detection. For the scope of this research, this filter will be the standard choice for cleaning these sudden peaks of noise from the input signal. With this component, the pause detector has been able to handle so far any input command in normal room noise conditions and identify the two segments being extracted, the object segment (Figure 14) and the instruction segment (Figure 15), provided that the command has this structure.

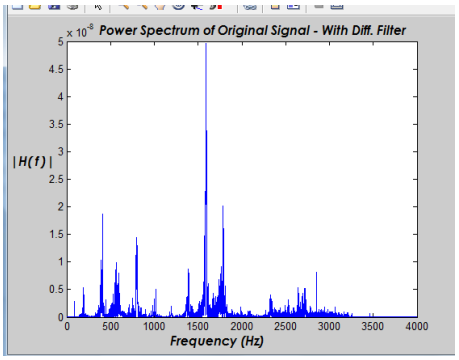

Figure 9
Frequency Spectrum after Applying Differential Filter

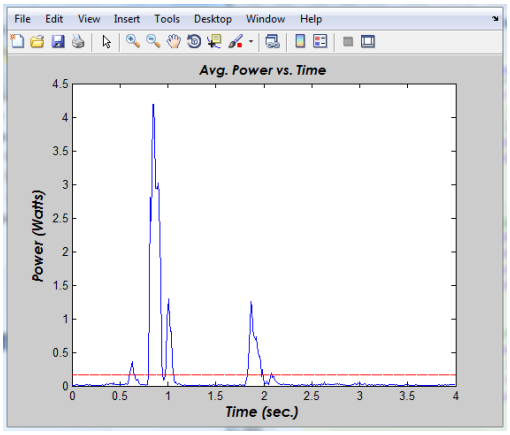

Figure 10
Avg. Power vs. Time Plot of Filtered Signal

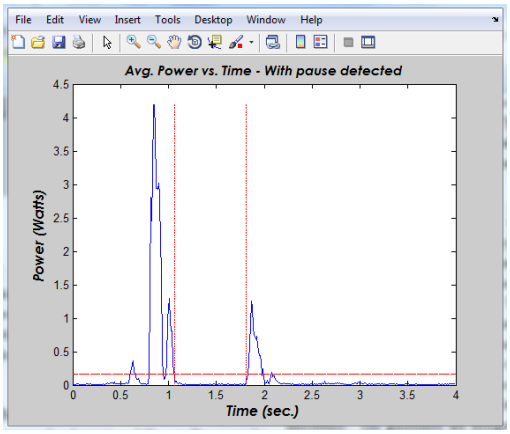

Figure 11
Significant Pause Detected and Marked in Time by the Pause Detector

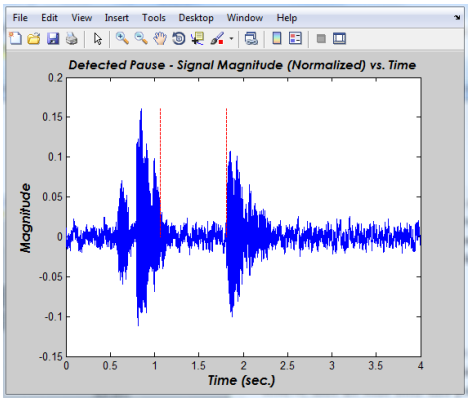

Figure 12
Significant Pause Detected in Original Input Speech Command

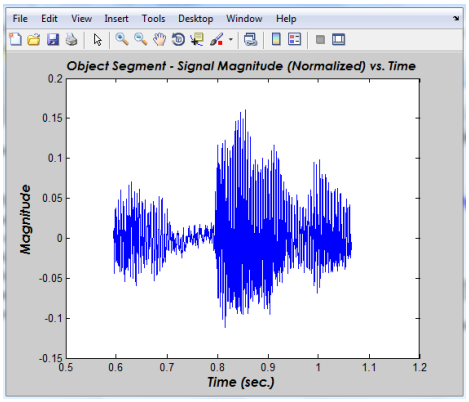

Figure 13
"Object Segment" of Input Command in Figure 2

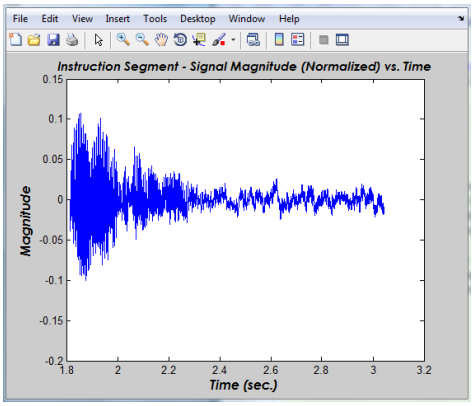

Figure 14
"Instruction Segment" of Input Command in Figure 2

PROCESSING SEGMENT-STRUCTURE

The diagram below, at Figure 15, shows how the proposed OSCT system's input structure should be processed. This concept diagram defines only the most important elements expected in order to achieve the system's basic functioning. Of course, alternative components can be substituted in the design depending on the application. As the diagram illustrates, the process flow initiates with the recording of the input speech command. If a corpus of speech commands is already prepared, it can also be input into the system. As part of the parameters of the initiation, the "Type" of input will be required to be specified in order for the system to either use the input to train the system or perform a real-case execution.

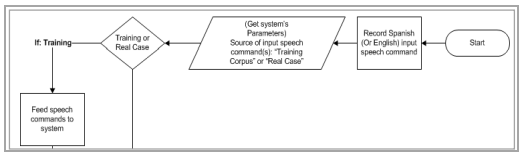

Figure 15
System's Preparation and Input Parameters

After the command(s) is/are introduced to the system, each individual input is then directed to a "4 second trimmer". This element will trim the signal time length to 4 seconds, in case the input is longer than expected. This trimmed signal will then pass through the "Pause Detector" for detecting the "Significant Pause" among the signal. The information about the location of the significant-pause will then be sent to the "Segment splitter, trimmer, and labeler" who will locate, cut, and identify the signal segments; object segment and instruction segment. After splitting these segments, the process-flow directs these signals towards their corresponding processing components (See Figure 16 and Figure 17). These will begin by analyzing, first, the object segment by using special speech recognition software which will provide the object's specific textual identification for the OSCT. The procedures taking place in the following components will be described further.

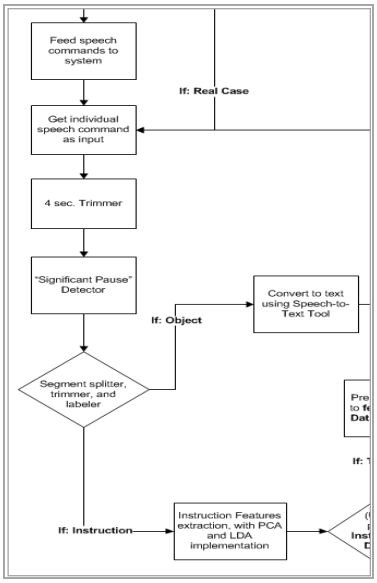

Figure 16
Object and Instruction Segments are Identified before Processing

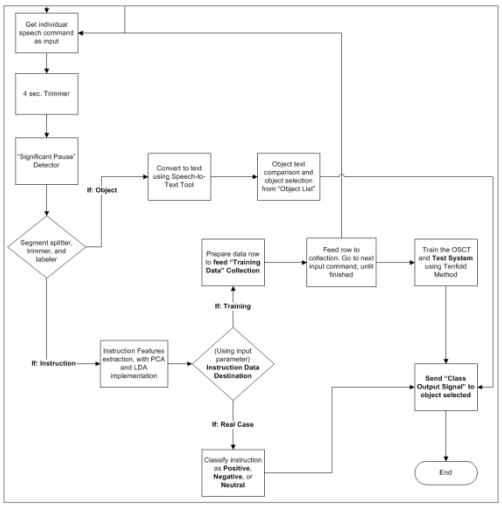

Figure 17
Object and Instruction Segments OSCT Components and Procedures

OBJECT TEXTUAL IDENTIFICATION

Once the "Object" signal segment is set apart, the segment is then sent to a Speech-to-Text application. This is done obtain a clear textual word that will directly identify the corresponding object in the system's registered "Object List". The system's list will be required to have all the names, labels, or nicknames assigned and connected to all the entities or devices that receive as input an "Activate", or "Deactivate" type of command. This means that any connected object should have a unique identifiable name assigned in the system before using its operations. Ideally, this identification process should take place before the "Instruction" segment analysis, in this way the analysis operation can take advantage of the identity of the object and adapt to this information in order to improve the accuracy of the instruction classification.

There are multiple open-source as well as proprietary applications that can convert any spoken word to text with high accuracy and efficiency, such as [3], [12], [13], and [14]. After investigating trustworthy, efficient, and commonly used open-source applications, CMU Sphinx Toolkit [14] was found to be a reliable candidate for speech-to-text conversions for the proposed system in this research. The CMU Sphinx Toolkit is actively used in speech recognition research as demonstrated by this list of research publications [15]. This toolkit has a variety of packages for different tasks and applications, such as the "Pocketsphinx" which is written in C, making it very useful for implementing on devices [16]. The object signal segment can be sent to this speech recognition engine which will provide the correct textual identification of the object for the OSCT system. Once the object is properly identified, the instruction signal segment will be analyzed. After this process is complete, the system will contact the selected object and execute the corresponding action depending on the classification of the input "Instruction".

INSTRUCTION CLASSIFICATION

There are multiple features that can be extracted from a speech signal. After researching for the most commonly used features among speech analysis publications, the following list of features were selected after studying the results obtained from [1], which, after implementing and comparing different classification techniques, achieved up to ~83% of accuracy in positive and negative emotion classification. These features were also voted as reliable in several other emotion classification and speech processing publications, including [4], [5], [6], [7], [17], [18], and [19]. Following the results of [1], the features elected for the system are:

- **Pitch:** Minimum, maximum, mean, standard deviation, absolute value, quantile, ratio of voiced and unvoiced frames
- **Duration:** Pause time between two disjoint segments of F0 (Or Pitch), and the vertical distance between the F0 segments which symbolizes voice breaks
- **Intensity:** Minimum, maximum, mean, standard deviation, quantile
- **Formant:** First formant, second formant, third formant, fourth formant, fifth formant, second/first formant, third/first formant
- **Rhythm:** Speaking rate

In order to maximize the Naïve Bayes classification accuracy up to 72.2%, the projection techniques, Principal Component Analysis (PCA) [20] and Linear Discriminant Analysis (LDA) [21], had to be applied. With the combination of these techniques we can de-correlate the data redundant feature space, and also project them into lower dimension. These are then used by the classifier for achieving even better classification results. It's by extracting these features and applying these techniques that the OSCT system's instruction classifier is expected to achieve an average classification accuracy of 72%.

CONCLUSION

Today, human-computer interaction is growing fast, and many technology experts agree that in the near future it will grow faster since past technologies are used to create new ones. Due to this reality, technology follows a never ending exponential trend. Speech command recognition systems, such as Smart-Phones voice dial services, Nuance's "Dragon Naturally Speaking", automated Call Centers, among many other ASR's (Automatic Speech Recognition) applications, are nearly a glimpse of what awaits for us in human-computer "relationship". It's toward a more natural direction that speech recognition research should be directed, and it's why I am proposing this idea. The idea which involves using the OSCT system for responding to commands or messages, just as you would expect from a primitive intelligence being. To reach natural intelligence by simulating, at least, its instinct's listening behaviors, is the goal for the ideal OSCT system. The initial proposed design of the components is relatively simple in order to simulate appropriately the primitive learning mechanism of nature. The essence of this idea is to follow a natural direction for speech recognition evolution, and also establish an essential primitive basis for this direction. I believe the correct order towards adapting computers into human behavior should be adapting computer-into-animal-into-human, rather than converting computer-to-human directly.

The proposed OSCT system could ideally reach an accuracy of up to 72% by using a Naïve Bayes Classifier for classifying the instruction segment. However, this accuracy could reach approximately 80% by choosing other classifying methods according to the published investigation. Depending on the computational resources and constraints, this could make the system even more reliable for common every day interactions.

REFERENCES

[1] Hoque, M. E, *et al.*, "Robust Recognition of Emotion from Speech", In Proc. IVA, 2006; August-20-2012, Retrieved from: http://web.media.mit.edu/~mehoque/Publications/IVA-Hoque-Yeasin-Louwerse.pdf

[2] Handel, S., "Classification of Emotions"; August-20-2012, Retrieved from: http://www.theemotionmachine.com/classification-of-emotions

[3] "Dragon Naturally Speaking, Software Developer Kits", Nuance's Dragon SDK Client Edition; September-13-2012, Retrieved from: http://www.nuance.com/for-developers/dragon/index.htm

[4] Datcu, D., *et al.*, "The recognition of emotions from speech using GentleBoost classifier", International Conference on Computer Systems and Technologies – CompSysTech, 2006; September-23-2012, Retrieved from: http://mmi.tudelft.nl/pub/dragos/37.pdf

[5] Pokorny, F., "Extraction of Prosodic Features from Speech Signals", Graz, 2011; September-21-2012, Retrieved from: http://iem.kug.ac.at/fileadmin/media/iem/projects/2010/pokorny.pdf

[6] Wang, Y., *et al.*, "An investigation of speech-based human emotion recognition", Multimedia Signal Processing, IEEE 6th Workshop, 2006, pp. 15-18

[7] Jang, J. R., "Audio Signal Processing and Recognition", ch.4-2; September-28-2012, Retrieved from: http://neural.cs.nthu.edu.tw/jang/books/audiosignalprocessing/index.asp

[8] "Write WAVE (.wav) sound file", MathWorks, Matlab Documentation Center, v.R2012b; October-05-2012, Retrieved from: http://www.mathworks.com/help/matlab/ref/wavwrite.html

[9] Assmann, P., "Speech Perception Lab, Recording and analysis of American English vowels, Sliding Windows

Analysis", August 2011; September-17-2012, Retrieved from: http://www.utdallas.edu/~assmann/hcs7367/lec4.pdf

[10] "Foobar2000, Advanced Freeware Audio Player", foobar2000, v1.1.15; September-17-2012, Retrieved from: http://www.foobar2000.org

[11] Xin, J., *et al.* "Filtering and Convolutions", iCamp-UCI Interdisciplinary Computational and Applied Mathematics Program, Math 77A Lecture; September-28-2012, Retrieved from: http://math.uci.edu/icamp/courses/math77a/lecture/filtering.pdf

[12] "HTK Hidden Markov Model Toolkit", HTK Open Source Speech Recognition, Toolkit Documentation; September-25-2012, Retrieved from: http://htk.eng.cam.ac.uk/

[13] "SPRACHcore Speech Recognition Software", SPRACHcore Open Source Speech Recognition, Toolkit Documentation; September-25-2012, Retrieved from: http://www1.icsi.berkeley.edu/~dpwe/projects/sprach/sprachcore.html

[14] "CMU Sphinx, Open Source Toolkit For Speech Recognition", CMU Sphinx Toolkit; September-25-2012, Retrieved from: http://cmusphinx.sourceforge.net/

[15] "CMU Sphinx Research Papers List", CMU Sphinx Toolkit Documentation; September-25-2012, Retrieved from: http://cmusphinx.sourceforge.net/wiki/research/

[16] "CMU Sphinx, Pocketsphinx Software", CMU Sphinx Toolkit Editions; September-25-2012, Retrieved from: http://cmusphinx.sourceforge.net/wiki/download

[17] Bettelheim, R., *et al.* "White Paper: Speech and Command Recognition for Voice Controlled Devices", Arcturus Networks, September 2010; August-30-2012, Retrieved from: http://www.arcturusnetworks.com/2011/08/31/white-paper-speech-and-command-recognition/

[18] Keun, H. K., *et al.*, "Study of a Vocal Feature Selection Method and Vocal Properties for Discriminating Four Constitution Types", Hindawi Publishing Corp., Evidence-Based Complementary and Alternative Medicine, Volume 2012

[19] Mower, S., *et al.*, "A Framework for Automatic Classification Using Emotion Profiles", IEEE Transactions on Audio, Speech, and Language Processing, Vol. 19, No. 5, July 2011

[20] Richardson, M., "Principal Component Analysis", May 2009; October-01-2012, Retrieved from: http://people.maths.ox.ac.uk/richardsonm/SignalProcPCA.pdf

[21] Julian, R., "Using LDA", Lilly Research Laboratories; October-01-2012, Retrieved from: http://miner.chem.purdue.edu/Lectures/Lecture10.pdf