Universiteit Leiden
The Netherlands

Bachelor Computer Science

Improving Public Speaking Performance by Facial Emotion, Body Language and Speech Recognition based Feedback

Damian Olav Domela Nieuwenhuis Nyegaard

Supervisors:
Dr.ir. D.J. Broekens & Prof.dr.ir. F.J. Verbeek

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

30 July 2019

Abstract

Speaking in public efficiently is relevant in a modern context. In the interest of advancing this area, we discuss the evaluation of a real-time performance feedback program on improving public speaking. Through a browser-based user interface, the user's facial emotion, body movements and speech is tracked in order to provide dynamic feedback that informs the user of common mistakes made during presentations. This project aims to contribute towards providing accessible software to improve public speaking and analyze the experimental findings of testing this software.

1 Introduction

Storytelling, pitching, presenting and public speaking have become an integral part of our careers and lives, as helping others understand our intentions and or problems can help us reach solutions we cannot reach on our own. Public speaking/oral communication is considered to be one of the most valuable abilities in the labor force[1], is considered to be a highly prevalent and disabling factor causing anxiety[2] and is considered to be a more common fear than even the fear of death[3]. In the interest of decreasing anxiety and fear towards public speaking and improving this ability, we create a controlled virtual environment in which verbal and non-verbal communication are automatically analyzed. Three existing frameworks allow us to correctly track verbal and non-verbal communication: speech, pose and facial expression recognition. To give appropriate feedback to the user, it is important to differentiate good from bad presenting in the context of the above three categories.

1.1 Motivation and related work

Other similar efforts have been made in order to potentially improve. Alternative approaches relative to this project are tackling anxiety directly as well as using a virtual environment to give dynamic feedback to the user. At Çankaya University, Turkey, 6 novice software engineers who had been confirmed to have public speaking anxiety, were subjected to a virtual environment designed

to simulate an auditorium[15]. In this simulated auditorium, another user is in control of creating anxiety causing phenomena, such as the amount of people in the virtual audience. Though there are similarities, this program is specifically aimed towards helping people who suffer from public speaking anxiety, while this program's target audience is universal, anxiety in this context isn't required for the user to potentially improve his/her public speaking skills. In 2010, a classifier was developed which handled speech emotion classification[16]. This classifier took the user's conversational activity, emphasis, influence and mimicry as feature inputs to infer in real-time what personality traits are relevant at that time. A very impressive correlation of about 0,85 was achieved by this program. However, it is important to note that the output of this classifier was in the form of the five-factor model[19], which approximates personality, instead of the basic emotions. By opting for the five-factor model, it focuses more on the the user's personality than the conveyance of emotions. Though there is much discussion considering the 6 Basic Emotions[20], in the context of practicing a speech, the Basic Emotions can be seen as "acted emotions" as this is close to a performance. In a similar study, a Multimodal Corpus[17] was designed to deliver an automated assessment of the user's public speaking skills. In other words, this project aimed to take human scoring out of the grading process of a presentation. This project took in speech delivery, speech content and non-verbal behaviors as input features to achieve a correlation of 0,477 for its Support-Vector Machine regression model, in comparison to a data set created by monitoring actual human scoring by experts. Unfortunately, this project isn't in real-time, has no concrete feedback besides the grading itself, doesn't take into account the user's expression and isn't convincingly accurate. Lastly, in 2015, an ambitious piece of software was programmed in [18]. This project simulates a virtual audience in real-time, who's behavior (nodding, clearing throat etc.) is augmented dynamically based on a performance assessment of the user's presentation, which receives eye gaze, facial expressions, gestures and voice quality as input parameters. This software manages to achieve a remarkable correlation of 0,745 between its automated prediction and an expert human assessment. This project's translation of a performance assessment to the behavior of a virtual audience is very impressive, but might lack the clarity of simple feedback, as the user needs to interpret the audience's behavior for himself as well. In addition to this, in comparison to our program, it lacks the ability to control how specific excerpts of the presentation are conveyed, creating a more static win-state for the user, as there is only 1 good way to present an excerpt.

Due to the apparent lack of one single project which delivers an accessible approximation of emotional conveyance by the user as well as provides clear dynamic feedback in real-time, this software was developed.

2 Research question

Can a program developed to give dynamic real-time feedback using speech, facial emotion and body movement recognition as input cause more awareness regarding what aspects of his or her speech need to be improved by testing emotional conveyance? Testing whether or not this program actually had a concrete effect on improving emotional conveyance is left open for future work. In addition to this, does this program offer a comfortable user experience to novice users? This program will deliver real-time feedback in the form of colored icons that are placed next to the current sentence in the speech to make clear to the user very quickly what he or she did correctly and/or incorrectly. During the development of this program, design decisions were made in the interest of providing an accessible end product.

2.1 Hypotheses

The evaluation of the feedback on, in combination with the recognition of the actions of the user, will yield results that hint at the user becoming more aware of what aspects of the speech need to be improved.

3 Methodology

To investigate if technological aids can help presenters become more aware of what aspects of the speech need to be improved, first the program in question needs to be outlined in its entirety. In this section, the three frameworks which form the basis for this project are explained, alongside several design decisions. Facial expression recognition is required to give a basis for the feedback given, real-time speech recognition is required to determine where in the speech we are currently, allowing us to give feedback on the appropriate text excerpt and pose estimation is required to give

feedback on the dynamicity of the user's pose. Following the outlining of the program, an experiment wherein users will test out the program with a prewritten speech will take place, after which these users fill in a questionnaire to determine their satisfaction with certain facets of the program.

3.1 Materials

3.1.1. OpenFace & OpenPose

Well-known programs to approximate Facial Expressions and Body Poses, are "OpenFace"[24] and "OpenPose"[25] respectively. Due to their recognition when it comes to projects of similar nature, they were considered in the earlier stages of this project as the frameworks of choice to estimate the Facial Emotion and Body Pose aspects. Both programs offer an array of features and options to the user. Fundamentally however, these programs were developed in Python, a programming language known for its accessibility and available libraries, but certainly not for its computational performance. These programs run either on a CPU alone or with the addition of a Nvidia GPU using CUDA. The fact that these pieces of software require a formidable CPU (and preferably a powerful CUDA-enabled GPU to avoid a reduction of ~50x performance on Ubuntu [6]) would force this project locally with certain strict requirements or on a capable server, diminishing its accessibility and complicating the developing process significantly. With respect to user accessibility, I opted for implementations of these aspects in Javascript (applicable to speech recognition as well), for these reasons, OpenFace and OpenPose are not used in this project.

3.1.2. TensorFlow and PoseNet (Javascript)

Tensorflow [2] "is an end-to-end open source platform for machine learning." Among its implementations, is one for JavaScript, a library in which Machine Learning models are developed and trained in JavaScript and deployed in Node.js or in the browser. PoseNet[5] is a collaboration between the developers behind Tensorflow (Google Brain Team) and Google Creative Lab[7]. PoseNet is a JavaScript written program which is fundamentally based on a Tensorflow Machine Learning model, developed for real-time pose estimation in the browser, accepting live webcam feedback as input. Capable of either approximating multiple persons or a single person with its algorithm, the latter yielding better performance, but requires only one person to be in frame to avoid incorrect results. In this project, single-person estimation is opted for, for its superior performance.

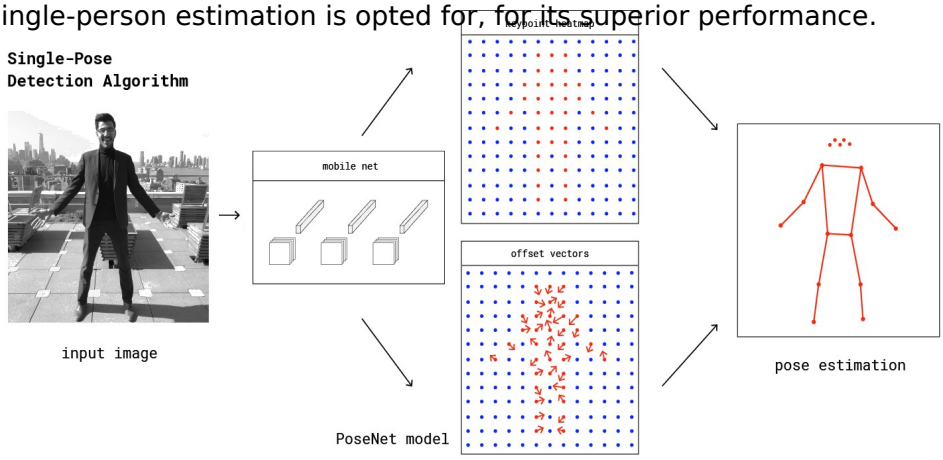


Figure 1. PoseNet Convolutional Neural Network Algorithm

Fundamentally, pose-estimation is done by a Convolutional Neural Network in two separate phases:
1. An input RGB Image is fed to a Deep Learning Network, weights of importance are assigned to aspects in the image (the keypoints of the body in this case).
2. A single-pose decoding algorithm is used to decode poses, pose confidence scores, keypoint positions, and keypoint confidence scores.

In table 1 we present a list of which body parts are recognized by PoseNet:

Index:	**Body Part:**
0	Nose
1	Left Eye
2	Right Eye
3	Left Ear
4	Right Ear
5	Left Shoulder
6	Right Shoulder
7	Left Elbow
8	Right Elbow
9	Left Wrist
10	Right Wrist

11	Left Hip
12	Right Hip
13	Left Knee
14	Right Knee
15	Left Ankle
16	Right Ankle

Table 1. Recognized Body Parts by PoseNet

3.1.3. Basic Emotions of Ekman

The six basic emotions of Ekman[9] consist of: happiness, sadness, disgust, surprise, fear and anger. These emotions are labeled as "basic", and as such form the set known as the "Basic Emotions". In Ekman's research, each of these emotions are defined by a combined position of the lower face, eyelids, eyebrows and forehead. Due to Ekman's definitions, we have access to his Facial Action Coding System[21], meaning we can scientifically map facial movements to human facial emotions.

The original paper that introduced the concept of Ekman's 6 Basic Emotions[9] came out in 1972. Since then, many have questioned the existence of this notion[26], as the 6 Basic Emotions are likely to represent a more complex set of signals. In this project we are considering the conveyed emotions as acted emotions, as the program monitors a performance. Making the academic skepticism towards Ekman's work not applicable to our situation.

3.1.4. Face API (Javascript)

JavaScript face recognition API[8] is a program that, similar to PoseNet, is built upon a TensorFlow Machine Learning Model. This TensorFlow based software has been developed and made publicly available, meaning it can be used to track facial actions of the user to conclude which of the six basic emotions of Ekman (combined with a neutral facial expression) are expressed at the current time.
The first step in the process of emotional recognition, is face detection. Several detectors are available[22], of which I opted for "Tiny Face Detector"[10]. A detector which is not quite as accurate as other options, but makes up for that by having relatively very fast performance.

After this, the input image is subjected to a Face Expression Recognition Model[23]. This model was trained on a variety of images from publicly available datasets.

In figure 2 some examples of input are presented, along with the given output of the recognized emotion and its confidence score (minimum of 0, maximum of 1):

Figure 2. The recognized emotions by Face-API (in addition to a Neutral expression)

In practice, each emotion listed is based on the interpretation of one of Ekman's 6 basic emotions, as shown in Table 2:

	Ekman's definition		
	Brows-Forehead	**Eyes-Lids**	**Lower Face**
Happy	Not relevant.	Eyes relaxed or neutral in appearance.	Outer corners of lips raised.
Sad	Eyebrows drawn together with inner corners raised and outer corners lowered.	Eyes may be looking downward or eyes may show tears.	Mouth either open with partially stretched trembling lips, or closed with outer corners pulled slightly down.
Disgust	Eyebrows drawn down but not together.	Lower eyelids pushed up and raised.	Raising cheeks; mouth either closed with upper lip raised and lower lip forward and/or out, or closed with upper lip pushed up by raised lower lip.
Surprised	Raised curved eyebrows; long horizontal forehead wrinkles	Wide opened eyes.	Dropped-open mouth
Fear	Raised and drawn together eyebrows.	Eyes opened.	Mouth corners drawn back, but not up or down.
Angry	Brows pulled down and inward.	Upper lids appear lowered.	Either lips tightly pressed together or an open, squared mouth.

Table 2. Ekman's detailed definitions of the 6 basic emotions

3.1.5. Google Web Speech API (Javascript)

Google's Web Speech API[11] is an open source JavaScript program developed to implement speech recognition that allows fine control and flexibility over its capabilities.
The API is built as an event handler, in the case that the user speaks, the spoken sentence is converted into a string in memory when the user is done with said sentence. The speech recognition functions as a timestamp signifying at what part of the speech the user is currently at, allowing the program to label the correct sentence with the current displayed expression.

In order to improve the speed of this project, intermediate results are enabled. Intermediate results are generated whilst the speaker is still mid-sentence, with somewhat lower accuracy.

Unfortunately, the Google Web Speech API requires the user to stop talking in order for it to be considered as the final result. This adds a lot of latency to finishing a sentence and the user needs to halt his speech every now and then. To speed this up and mediate the latency, this program works with said intermediate results from the Google Web Speech API. If the intermediate results match with the first two words of the following sentence in the speech, our text matching algorithm concludes that we are currently at that sentence. This results in a more improved flow of the program's recognition of the speech.

3.1.6. String Similarity Function

We receive the spoken text from the Google Web Speech API in the form of a string in memory. Unfortunately, the accuracy isn't quite 100% as each speaker has a distinct dialect and a distinct level of articulation. Due to the speech recognition not being one to one, we need to recognize a spoken sentence which is *similar enough* to the associated sentence in the speech.

In order to recognize a *similar enough* string, we need to compare two strings and apply an assertion of correlation between the two strings. This is done by a similarity function in JavaScript which is based on the Levenshtein distance. In this algorithm, computations use the length of the two string, making a distinction between the longer string length(L) and the shorter string length. The Levenshtein distance is a measure of similarity between the two strings which uses the amount of deletions, insertions and substitutions necessary to transform the first string into the second string(T), returning a value between 0 and 1 which signifies the level of similarity between the two strings (0 meaning that there was no correlation and 1 meaning that the strings are identical).

$$Levenshtein\,distance = (L) - (T / L) \quad (1)$$

3.1.7. Average Speech Velocity

Given that speech recognition is a component of our program, applying it in a manner that also directly gives feedback to the user is an effort worth the trouble of implementation, as the speaker speaking at an understandable pace is important to bring across information. This implementation boils down to the analysis of the speech velocity of the user. In the English language, based on the combined effects of syllable weight (stressed or unstressed), syllable position in the sense-group (final or non-final) and syllable type (closed or open) have been joined by P. Delattre[12] to form an average duration of time per syllable for all of these three factors (in centiseconds):

			English	*German*	*Spanish*	*French*
Stressed	Final	Closed	40.81	36.15	32.13	34.12
		Open	33.45	29.75	24.50	24.57
	Non-final	Closed	25.88	24.56	25.88	
		Open	19.19	19.72	20.23	
	Final	Closed	25.62	27.81	23.03	
		Open	21.24	17.69	18.52	
Unstressed	Non-final	Closed	15.50	17.51	19.27	19.19
		Open	12.02	13.22	18.16	13.74

With this decision tree, we can determine an upper and lower bound of speech velocity (in syllable per centisecond). The upper bound being a combination of stressed syllable weight, final syllable position and closed syllable type (40.81 centiseconds), while the lower bound is set by a combination of unstressed syllable weight, non-final syllable position and open syllable type (12.02 centiseconds). If the speaker's speech velocity is slower than this lower bound or faster than this upper bound, feedback is given to the user that his speech velocity is too fast or too slow.

3.1.8 Feedback System

Naturally, in the interest of improving the quality presentation, the results of the software's analysis are to be made clear to the user. The communication between the software and the user comes in the form of real-time feedback with icons which convey the feedback in a visual manner without text. These icons are dynamically attached to the end of a spoken sentence in the User Interface after it has been recognized/marked by the Google Web

Speech API. Below an image can be found detailing which icon is supposed to convey what feedback:

1. The user is not correctly facing the screen.

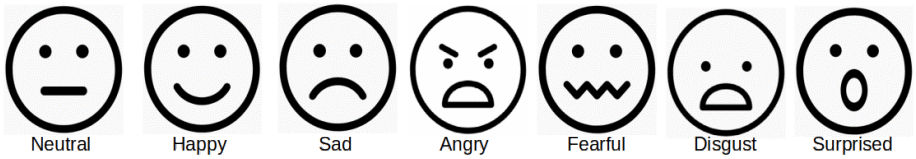
2. The user is talking relatively fast.

3. The user is talking relatively slow.

4. The user is moving relatively little.

5. The user displayed the correct emotion for this text excerpt.

6. The user displayed the incorrect emotion for this text except.



1. Using the aforementioned PoseNet framework, we can recognize 17 keypoints of the human body. 5 of which are facial keypoints: The nose, the left ear, the right ear, the left eye and the right eye. In a situation where PoseNet is not able to identify one of these 5 facial keypoints, we can conclude that the user is facing away from the screen, which also hinders the JavaScript face recognition API and its ability to identify what expression the user is making, alongside the lack of eye-contact gives people the feeling that they're aren't fully in communication[13].
2. Compared to the lower bound of the average speech velocity discussed in 3.1.7, the user is talking relatively slow. This icon is a subtle hint to the well-known fable of "The Tortoise and the Hare".
3. Compared to the upper bound of the average speech velocity discussed in 3.1.7, the user is talking relatively faster. This icon is also a subtle hint to the well-known fable of "The Tortoise and the Hare".
4. Using the PoseNet component of our program, we can detect the velocity of the user's movement by tracking the difference of the coordinates over time. More on that in section 3.1.9.
5. According to the comparison between the highlighted text excerpt and what emotion was displayed during the user's speech during that segment, the user accurately conveyed the right emotion, which is why positive feedback is given.
6. According to the comparison between the highlighted text excerpt and what emotion was displayed during the user's speech during that segment, the user accurately conveyed the incorrect emotion, which is why negative feedback is given.

Two colors are used to fill the icons, red and green. Red being associated with a negative valence and green being associated with a positive valence[14], which is why positive feedback is green and negative feedback is red. Along with 1-5 of the icons, all of the 7 expressions displayed are always displayed with either a red or green color, which it shares with either the 4^{th} or the 5^{th} icon. If a text excerpt was set to be conveyed happily, but the user conveys it angrily, a red happy icon is given as feedback, meaning that the conveyed emotion was incorrect, the user was supposed to display happiness in this segment. In the case that the user does dispay happiness during this segment, a green happy icon is shown to signify that happiness was correctly conveyed.

3.1.9. The N Frame Average Function

In order to make conclusions for emotional expressions and velocity of momentum more accurate, we need to only arrive at such conclusions when the evidence towards it is profound. For emotional expressions, this means that we only have confidence that a certain emotion is displayed, if it's been around for some N time. The same applies to velocity of movement, we can only correctly give feedback to the user whether his pose is static if it has been so for some N time.

Here are the two base formulas, which apply to both velocity of movement and emotional expressions:

$$NSum = (N-1) * (R) + I \quad (2)$$
$$R = (NSum / N) \quad (3)$$

The first formula is an update rule for the sum of the previous N frames by adding the current frame information(I), to (N-1) times the current running N frame average(R). The current frame information(I) contains the value for the current frame of the aspect being measured, being either movement velocity or emotional expression confidence scores.
The second formula is an update rule for the N frame average, this is done simply by dividing the new sum of the previous N frames by N.

3.1.9.1 Velocity Tracking

PoseNet approximates the 17 skeletal keypoints from the input it receives from the webcam per still frame, visual points are drawn and over time, the still frames add up to the video feedback the user receives. Per frame update, the relative location (in the form of an x and y coordinate) of all 17 keypoints are calculated. If compared to the previous frame's 17 keypoints, we can calculate the velocity in distance per frame. Below are the applied formulas:

$$VelocitySum = (N-1) * runningNFrameAvg + currentFrameVelocity$$

$$runningNFrameAvg = (VelocitySum / N)$$

Here we see the application of aforementioned formulas to the velocity of the user's movement. An important annotation to make is that to calculate the velocity of the user, we need to use the Pythagoras function. This is due to the fact that, if based on the coordinate (x, y) tuples, calculating the distance per frame/velocity is done using this formula for every frame:

$$currentFrameVelocity = \sqrt{|Xnew - Xold|^2 + |Ynew - Yold|^2}$$

In order to calculate the distance from one point to another in a 2 dimensional space, we can use the absolute horizontal movement and the absolute vertical movement as parameters for the Pythagoras function to get the movement along the hypotenuse. As the hypotenuse movement is measured per frame, it is seen as velocity, which is updated every frame.

3.1.9.2 Facial Expression Tracking

After every successful analysis of the current frame's facial expression by the JavaScript face recognition API, we receive confidence scores towards each of the 6 basic emotions and the neutral expression. To avoid sudden conclusions, we apply the same two formula's, but somewhat differently:

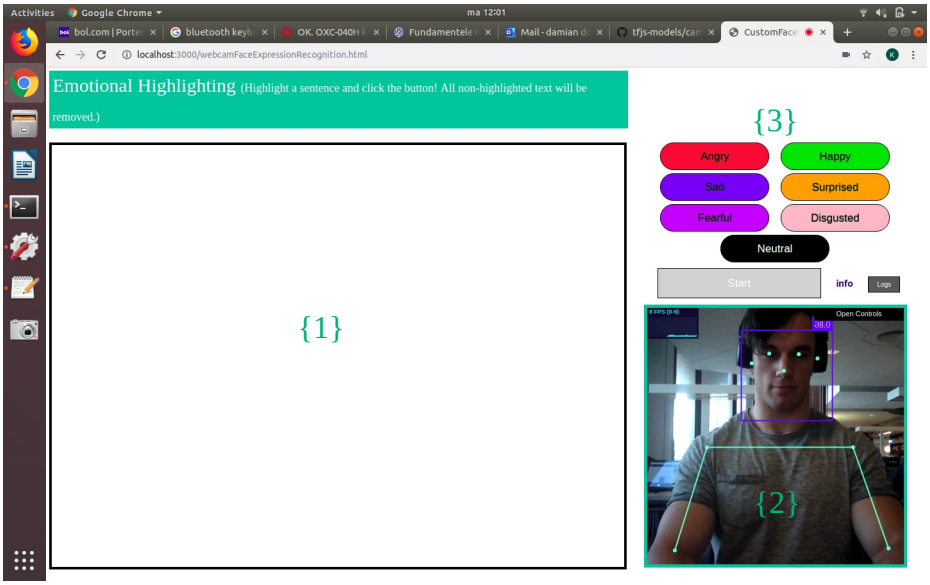
$$emotionSum[i] = (N-1) * emotionNAvg[i] + currentEmotion[i]$$

$$emotionNAvg[i] = (emotionSum[i] / N)$$

These operations take place in a for loop of 7 iterations, one for each emotion. The highest of the emotionNAvg elements its index is tracked, in order to know what emotion over N frames currently has the highest confidence score on average.

3.1.10. The User Interface

The software's front end is what connects the software's back end and the user. Here is an example of what the browser looks like during the execution of our framework:



{1}. **The Input Field:** In this field, the user is supposed to input his speech in text form, after which the user is able to highlight each distinct sentence with an emotion from the emotion set presented in {3}. In the case that the user highlights a sentence as a selection, the selection is checked on being a correct sentence (ending on a full stop). If so, the sentence's class in HTML is changed to the appropriate emotion, which triggers the font color of this sentence to the associated color in {3} to make clear to the user that it has correctly been highlighted.

{2}. **Webcam Video Frame:** The user is informed in real-time about the software's accuracy through the usage of visual markings. Firstly, a blue rectangular box with confidence score attached marks the successful recognition of the user's face and expression, secondly, up to 17 keypoints on the human body can be detected, which, if the confidence score for the respective keypoint is high enough, will be displayed on the Webcam Video Frame as light blue points. As a consequence, the user can change his posture to help the software pick up keypoints and recognize his or her face.

The PoseNet framework is developed to have a Graphical User Interface which offers a small window monitoring the performance of the pose estimation software as well as options to the user which alter the parameters for the algorithm.

{3}. **The Emotion Set:** A set of buttons which includes the 7 aforementioned expressions of ‘neutral’, ‘happy’, ‘sad’, ‘angry’, ‘fearful’, ‘disgusted’ and ‘surprised’. These buttons need to be clicked by the user to mark a highlighted sentence with the respective expression. Hovering over each button shows an example of how the respective emotion icon would look. Below these 7 buttons we have the start button, which starts the presentation by starting speech recognition (Pose estimation and Face recognition are already running, but with speech recognition being the catalyst of our feedback, starting it also starts the presentation). Once the start button has been clicked, it is either colored in with green or red. Green suggesting that speech recognition is enabled and the software is active, red suggesting that the program has stopped. Next to the start button we have an info element which, by hovering the cursor over it, details what feedback the user can receive, as is visible in this paper under heading 3.1.8. Lastly, the log button transforms the content {1} into the previously given feedback to the user, which might come in useful for review.

3.2 Experimental setup / approach

As we need to measure to what extent users have become more aware of what aspects of the speech need to be improved, the experiment is structured to make assertions about the experience of the user, which is done by handing out a questionnaire regarding the experience. With a sample size of N = 10, this experiment gives the user a short sample speech, which is submitted into the User Interface with the following emotional highlighting:

Hello everyone.
Thank you all for coming.
Welcome to today's presentation.
This is an experimental simulation to test the capabilities of this software.
The text is divided into parts, which are all highlighted with different emotions.
The speech recognition is used as an indicator to know what part of the speech the user is currently at.
The user's movements are tracked in order to stimulate a dynamic pose while presenting.
This is done through the use of icon feedback in the User Interface.

The user is seated at a distance of around half a meter from the webcam, a distance which will make it possible for the pose-estimation to track the movement of the user's torso, arms and head while also allowing the microphone to accurately recognize the user's speech.
Before the experiment, the program is concisely explained to the user, to ensure that the user knows what the aim of the experiment is.

3.3 Measures

After using the program, the user is asked to fill in said questionnaire, which poses several questions regarding the aspects of the program which have been developed in this project and the aspects which have been developed by others (to determine where the possible weak and strong points of this program lie). Each question is to be answered with a grade from 1 to 5, matching the user's thoughts (5 being perfect and 1 being terrible).
The questionnaire's form and its corresponding questions are detailed on the following page, with the results being visualized afterwards:

Questionnaire Bachelor Project Damian Domela: GRADE

1. How intuitive was the User Interface? 1 2 3 4 5

2. How well did the program recognize your speech? 1 2 3 4 5

3. How well did the program track your movements? 1 2 3 4 5

4. How well did the program recognize your expressions? 1 2 3 4 5

5. How helpful was the real-time feedback? 1 2 3 4 5

6. How good/stable would you rate your internet connection? 1 2 3 4 5

7. How intuitive were the icons seen in the User Interface? 1 2 3 4 5

8. How well did the program track your velocity of speech? 1 2 3 4 5

9. How well does text highlighting work? 1 2 3 4 5

10. How would you rate the entire program? 1 2 3 4 5

11. How much has this program helped you become more aware of what aspects of the speech need to be improved? 1 2 3 4 5

Other reaction(s) (optional):

4 Results

x	UI	Speech recognition	Movement	Expression	Real-time feedback	Internet connection	Icons	Speech velocity	Text highlighting	Program	Awareness
Test 1	5	3	5	3	4	4	5	1	4	4	3
Test 2	5	4	4	2	4	4	5	2	3	5	3
Test 3	5	3	4	4	4	3	5	1	5	5	2
Test 4	5	2	5	3	5	3	4	2	4	4	2
Test 5	5	3	5	3	4	3	5	2	3	4	3
Test 6	5	3	5	2	5	3	5	2	3	4	2
Test 7	4	3	5	3	4	3	5	2	3	4	2
Test 8	5	3	5	2	5	3	5	2	4	4	2
Test 9	5	2	5	2	4	3	5	2	3	4	2
Test 10	5	3	4	2	2	3	4	1	5	4	2
Avg/5	***4.9***	***2.9***	***4.7***	***2.6***	***4.1***	***3.2***	***4.8***	***1.7***	***3.7***	***4.2***	***2.3***

4.1 Results Discussion

{1}. **How intuitive was the User Interface? [4.9]** The User Interface is clearly structured and apparently this translated well to the test subjects. Every utility which the user has access to within the framework on the front end is visible in one single screen without the need for scrolling or clicking, minimizing the potential confusion at run time for the user.

{2}. **How well did the program recognize your speech? [2.9]** Seeing as the Google Speech API for Javascript[11] is a separate framework but also needs to be implemented into our program, whether the performance of the speech recognition is a factor for which this project should be held accountable is somewhat of a gray area. However, the manner in which the speech recognition was implemented is largely similar to the suggested implementation in [11], meaning that the accuracy of the speech recognition should be a feat for which this project shouldn't be held responsible in its entirety. With that being said, as mentioned before, slight adjustments were made to make the user experience more comfortable as well as increase the speed at which sentences are recognized. This has somewhat improved the accuracy of the speech recognition, meaning that partially, this project helped improving the speech recognition. This decent grade of 2.9 does hint at the fact that the accuracy is quite far away from 100%, which is unfortunately the case.

{3}. **How well did the program track your movements? [4.7]** The PoseNet[5] framework, with proper lighting, yields excellent dynamic results in tracking the user's movements. This is directly visible in the relatively high grade of 4.7 it received from the test subjects.

{4}. **How well did the program recognize your expressions? [2.6]** The Face API[8] is decent at recognizing several expressions, namely "angry", "happy", "neutral", "surprised" and "sad". The other two expressions; "fearful" and "disgusted" are very difficult for the software to have them recognized. In addition to this, the framework isn't built to recognize emotions on the user's face while

he or she is speaking, which also lowers the accuracy of the approximation of expressions in this context. Both these factors are reflected in the mere mediocre feedback grade of 2.6 from the test subjects.

{5}. **How helpful was the real-time feedback? [4.1]** The real-time feedback is concise and dynamically disappears when it's no longer relevant, making what feedback is relevant more clear. The real-time feedback is situated directly next to what sentence is currently active and is dynamically based on the running frame average of what confidence score for what expression is the highest, increasing the accuracy. This apparently resulted in the test subjects giving a great feedback grade of 4.1 for this aspect of the project.

{6}. **How good/stable would you rate your internet connection? [3.2]** The test subjects were located at two different places. 2 out of the 10 test subjects rated their internet connection a 4 out of 5 and the other 8 rated it a 3 out of 5. The strength of the user's internet connection is important as the speed and latency of the speech recognition are dependent on the user's internet connection. All in all, we can conclude that the internet connection of the test subjects in this experiment were slightly above average.

{7}. **How intuitive were the icons seen in the User Interface? [4.8]** The User Interface was designed to make sure that before the speech recognition has started, during the text highlighting portion of the process, an automatic pop-up window has notified the user about what icon is associated with what expression. For icons that aren't expressions, the User Interface provides an "info" field which, when the cursor hovers over it, informs the user about the usage of the other icons. This forces the user to be informed with the meaning of the icons before starting the presentation, which is reflected in above feedback grade of 4.8.

{8}. **How well did the program track your velocity of speech? [1.7]** As mentioned before, the upper and lower bound for the speech velocity per syllable are very precise and strict. Our only option to determine how fast the user is speaking, is by using the Google Speech API, which adds a lot of latency, which makes it hard to determine the precise length of how long the user was speaking per syllable. The most precise manner was to use intermediate results and measure how long the user took to pronounce the syllables from the first two words of the current sentence. However this results in an unstable prediction of whether the user speaks too fast or too slow, as it's based on a prediction of only the first two

words. This is why the 1.7 feedback grade from the test subjects is very low, as they received inaccurate feedback on their speech velocity.

{9}. **How well does text highlighting work? [3.7]** Highlighting one or multiple sentences is very easy and requires only two clicks per operation. The expression buttons are colored adequately and are easily accessible. Every now and then the next sentence is also highlighted with the current emotion, but that's due to the range of a text portion being highlighted by the cursor is somewhat tricky to get right. The decent feedback grade of 3.7 from the test subjects hints at this portion of the framework working well.

{10}. **How would you rate the entire program? [4.2]** The combination of the PoseNet, Face API and Google Speech Web API frameworks is rather novel. Though the accuracy isn't great, the program is functional in its entirety. The test subjects' feedback of 4.2 grade is quite high, expressing the interest of the test subjects towards the program as a whole and their comfort accessing the program.

{11}. **How much has this program helped you become more aware of what aspects of the speech need to be improved? [2.3]** Becoming more aware of what aspects a speech needs to be improved on is quite a difficult feat to accomplish. The fact that the facial expression recognition doesn't work great in this context as well as the speech recognition not being precise enough in its timing to accurately give feedback on the velocity of speech from the user makes the real-time feedback from the program less valuable. This is reflected in the test subjects' mediocre feedback grade of 2.3, which is less than sufficient.

5 Conclusion and further research

5.1 Shortcomings of the Program

The Google Web Speech API requires a connection to the internet to work, which in turn, requires this whole project to have an internet connection. As the speech velocity is determined by the Google Web Speech API, and the speed at which the speech recognition works slightly varies due to it being dependent on an internet connection, determining precisely how long the speaker took to pronounce a certain amount of syllables is difficult. Not only is the speech recognition an obstacle in this aspect, but the built-in microphone for a computer as well as the Face API require the user to be close

to the computer to get as accurate results as possible in speech recognition and expression recognition respectively. A possible solution for this would be to use an external microphone, but this would decrease the accessibility of the program as well. The built-in microphone results in the webcam not picking up the lower parts of the user's body as they're outside of the video frame, so the movement tracking is based on just the upper body. Unfortunately, some of the 7 expressions are difficult to have recognized and speaking disturbs a human's expression and thus the software's ability to track it. In addition to this, both PoseNet and the Face API require good lighting to achieve a high confidence score towards their respective recognition.

5.2 Conclusion

The merging of the Google Web Speech API, TensorFlow's PoseNet and Face API frameworks combined into a multifaceted Back End to our self-developed User Interface Front End. In addition to The goal of this merging of frameworks was to improve the user's conveyance of emotions during a speech. The User Interface was responsible for notifying the user to what extent he or she was successful in conveying the desired emotions, which, according to the results in our experiment, provided a comfortable user experience. The user interface used a feedback system based on colored icons to quickly bring across feedback to the user. To correctly implement speech velocity, a string similarity function was developed to check whether or not the spoken sentence matched the current sentence in the speech. To avoid hasty conclusions, an N frame average function was implemented in order to only make conclusions when there is a consistent confidence towards it over the span of N frames. In the experiment, test subjects were confronted with the program's interpretation of their ability to convey the correct emotions in their speech in the form of real-time feedback and thus possibly making the user more aware of what aspects of his speech need to be improved. The goal of the experiment was to test to what extent the developed application provided a comfortable user experience, while also checking whether the user became more aware of what aspects of his or her speech need to be improved. When looking at the results of the experiments in the form of the feedback grades from the test subjects, though the sample size is small, it hints at a great and intuitive user experience and a small to negligible effect towards sparking more awareness for the user, with it receiving a 2.3 out of 5 from the sample group. Furthermore, the User Interface, PoseNet, the real-time feedback, the icons and the program as a whole received more than positive feedback, while the average speech

velocity, expression and speech recognition received less than positive feedback.

5.3 Future Work/Research

A legitimate research towards whether or not the tool improved the user's ability to correctly convey emotions during a presentation is needed to make conclusions about the developed application having an actual effect on the user. This would require a significantly larger sample size for its experiment. Also, in this project, the basic emotions of Ekman were chosen as a basis to give feedback on, however, the amount of bodily energy or arousal the user is portraying towards the audience could be an alternative solid basis to conclude how well the user is presenting. This alternative is a great foundation to implement a different approach to this project in future research.

Bibliography

1. Dwyer, K.K., Davidson, M.M.: Is Public Speaking Really More Feared Than Death? *Communication Research Reports* · pages 7-8 · April 2012.

2. Google Brain Team, Tensorflow, November 9th 2015, <https://www.tensorflow.org>

3. P. C. Kyllonen. Measurement of 21st century skills within the common core state standards. In Invitational Research Symposium on Technology Enhanced Assessments. *Technology Enhanced Assesment,* pages 7–8, May 2012.

4. C. B. Pull. Current status of knowledge on public-speaking anxiety. *Current Opinion,* Page 37, January 2012.

5. Google Brain Team, Google Creative Lab, PoseNet, April 6th 2018 <https://github.com/tensorflow/tfjs-models/tree/master/posenet>

6. Gines Hidalgo, OpenPose Issue: CPU version – Running Without CUDA, January 18th 2018 <https://github.com/CMU-Perceptual-Computing-Lab/openpose/issues/375>

7. D. Oved, Real-time Human Pose Estimation in the Browser with TensorFlow.js, May 7th 2018 <https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5>

8. Vincent Mühler, JavaScript face recognition API, May 19th 2018 <https://github.com/justadudewhohacks/face-api.js>

9. Ekman, P. Universals and Cultural Differences in Facial Expressions of Emotions. *Personality Processes and Individual Differences,* Page 251-252, March 13th 1987

10. The MatConvNet Team, Tiny Face Detector, July 2017 <https://github.com/peiyunh/tiny>

11. G. Shires, Voice Driven Web Apps: Introduction to the Web Speech API, May 29th 2019 <https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API>

12. P. Delattre, A Comparison Of Syllable Length Conditioning Among Languages. *International Review of Applied Linguistics in Language Teaching*, Page 186, Table 1, September 1966

13. M. Argyle, J. Dean, Eye-Contact, Distance and Affiliation. *Sociometry,* Page 289, September 1965

14. A.C. Moller, A.J. Elliot, M.A. Maier, Basic Hue Meaning Associations. *Emotion* Page 901, December 2009

15. M.D. Nazligul, M. Yilmaz, U. Gulec, M.A. Gozcu, R.V. O'Connor & P. Clarke, Overcoming Public Speaking Anxiety of Software Engineers using

Virtual Reality Exposure Therapy. *Systems, Software and Services Process Improvement,* page 5, August 12th 2017

16. T. Pfister, P.Robinson, Speech Emotion Classification and Public Speaking Skill Assessment. *Human Behavior Understanding,* page 7-8, August 2010

17. L. Chen, G. Feng, J. Joe, C.W. Leong, C. Kitchen, C.M. Lee, Towards Automated Assessment of Public Speaking Skills Using Multimodal Cues. *ICM '14,* page 3-4, November 12th-16th 2014

18. T. Wörtwein, M. Chollet, B. Schauerte, L. Morency, R. Stiefelhagen, S. Scherer, Multimodal Public Speaking Performance Assessment. *ICM '15,* page 44-45, November 9th-13th 2015

19. J.M. Digman, Personality Structure: Emergence of the Five-Factor Model. *Annual Review of Psychology*, page 421, February 1990

20. P. Ekman, Are There Basic Emotions? *Psychological Review,* page 552, February 10th 1991

21. P. Ekman, E.L. Rosenberg, What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS). Page 13, January 1997

22. Vincent Mühler, Available models Face-API, May 19th 2018 <https://github.com/justadudewhohacks/face-api.js#available-models>

23. Vincent Mühler, Face Expression Recognition Model Face-API, May 19th 2018 <https://github.com/justadudewhohacks/face-api.js#models-face-expression-recognition>

24. B. Amos, B. Ludwiczuk and M. Satyanarayanan, OpenFace: A general-purpose face recognition library with mobile applications. *Technical Report,* page 5-8, June 2016

25. Z. Cao, G. Hidalgo, T. Simon, S. Wei and Y. Sheikh, OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. *Transactions on Pattern Analysis and Machine Intelligence, page 6-7,* December 18th 2018

26. R.E. Jack, O.G.B. Garrod and P.G. Schyns, Dynamic Facial Expressions of Emotion Transmit an Evolving Hierarchy of Signals over Time. *Current Biology,* page 5, January 20th 2014