Personal Teaching Aid

Sara Aerssens
Caitlin Lagrand
Putri van der Linden
Lina Murady
Tirza Soute
Leila Talha

Personal Teaching Aid

How student performance can be increased using present-day multimedia technology

Sara Aerssens 10792392
Caitlin Lagrand 10759972
Putri van der Linden 10768017
Lina Murady 10776389
Tirza Soute 10761977
Leila Talha 10756922

Project Report
Course Media Understanding
6 EC

Bachelor Artificial Intelligence

College of Science
University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Course coordinator
Dr. A. Visser

Informatics Institute
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

April 2nd, 2017

Abstract

Personal Teaching Aid was designed to battle the current shortage of primary school teachers in The Netherlands. The software consists of several modules, of which each specialises in a specific task. The first of these modules is the face recognizer, which either recognizes a familiar student or comes to know a new student by taking several pictures. The next module is a speech recognizer to handle a student's verbal answers to the problem set that it is given. These arithmetic problems are, dependant on the students arithmetic skills, generated by a self-implemented problem generator. The generated problems are presented in both a visual way, using a GUI, and in a verbal manner, using a text-to-speech module. These modules together form a basic, but complete product.

Contents

1 Introduction

The Netherlands is currently battling a shortage of primary school teachers[1], which has consequences for the achievements and development of young students. Previous studies show that a large class room size can negatively affect the performance of pupils in basic subjects due to lack of personalized aid and increased havoc in the group [Brühwiler and Blatchford, 2011]. Furthermore, the mathematical knowledge of the teacher significantly affects the performance of its students [Hill et al., 2005]. Recent studies also show that a virtual learning environment can improve student performance [Shih-Wei Chou, 2005]. These studies lead to the question of how student performance in basic primary school skills can be increased by providing personalized assistance with expert knowledge, using present-day multimedia technology. This project will thus seek to propose a solution to the current teacher shortage.

In the present study, a solution to this problem is found by creating personalized teaching aid software for primary school children. The study will focus on a tool to improve a student's arithmetic skills, although the program can be extended to other areas as well should it prove useful. Additionally, two studies were found in which traditional studying methods were compared to robot-assisted methods and the results showed that the use of a robot improved the children's concentration, interest and even academic achievement [Han et al., 2008] [Chin et al., 2014]. This lead us to add the possibility to use a robot in combination with our personalized teaching aid software.

The goal is to create a program with user-specific guidance features that will help students practice arithmetic problems in a way that is more personal than existing standard learning environments. This should help to relieve the workload of the teacher, and in turn offer each individual student more guidance in class. The additional use of the robot should help motivate and improve the concentration of the students, as well, as proved in previously mentioned studies. Since the program will be specifically built around arithmetic problems, the program must have proficient math knowledge in order to positively affect the performance of the students.

Section 2 will give an overview of the product that is developed. Section 3 will explain the methods used to develop the final product. In section 4, there will be given a method to evaluate the final product. In section 5, the results will be evaluated and section 6 will discuss these results. Finally, section 7 will present a conclusion.

[1] `http://www.nu.nl/binnenland/4308254/basisscholen-waarschuwen-groot-lerarentekort.html`

2 The Product

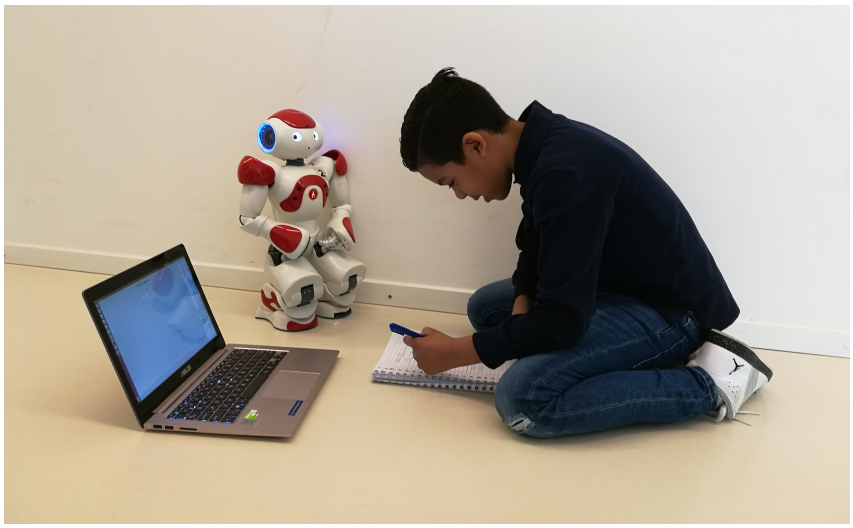


Figure 1: A student using the product.

In this section an overview of the developed product is given. The goal is to create a program that has human-like properties, so that it can be considered a basic simulation of a teacher. It has to recognise the students, have basic interactions with it (e.g. greeting, encouraging) and guide it through generated arithmetic problems. The final product will consist of the following components:

- **Facial recognition** The program starts by either recognizing a user that has used the program before or adding new users to the database. When the user is recognized, personalized data about the user will retrieved from the database.
- **Personalized user knowledge** The program stores and retrieves data about the user such as their name, progress and skill in different types of math problems.
- **Arithmetic problem generation** The program has to generate arithmetic problems according to the user's skills, which is updated after each completed set of problems. This way, the student can practice the problems they have difficulties with, or skip problems it is proficient in.
- **Text-to-speech** The program has to greet the user when it starts the program and read the generated problems out loud. It lets the user know when it has solved a math problem correctly or incorrectly.
- **Speech recognition** The student can interact with the program through speech by being able to respond to the problems out loud.

- **Graphical user interface** The communication through speech is complemented by a graphical interface that also shows the arithmetic problem on screen. It adds a visual component to the program and ensures a smooth interaction.
- **NAO Robot** The NAO robot[2] is used for the interaction between the student and the program. The NAO robot speaks and listens to the student and the camera of the NAO robot is used for face recognition.

Figure 2 shows a detailed overview of the program. The blue components are part of the Graphical User Interface (GUI), which starts the program and shows problems and the results of the user's solutions. The student's intermediate scores, however, are not shown. For the functioning of the program the final results of the session are sufficient. By leaving out the intermediate results the code becomes more efficient. Furthermore, it is unlikely that student's, certainly the younger ones, fully understand the meaning of the proportion between correct and faulty answers. The yellow segments are the interactive components, which uses the robot or computer's camera and speakers. The green parts are the personalized user knowledge.

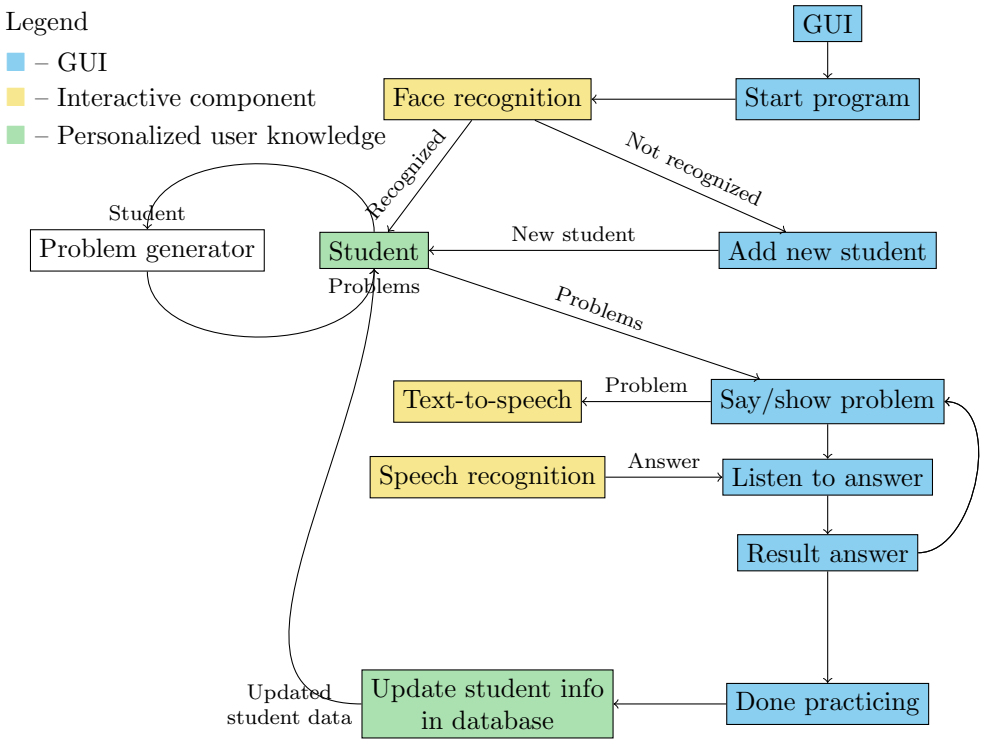


Figure 2: Program structure.

[2] https://www.ald.softbankrobotics.com/en/cool-robots/nao

3 Method

This section will describe the different components of the product in detail. The interaction of the program is separated into five parts. Section 3.1 will explain the method that was used for face recognition. In section 3.2, speech recognition will be clarified and text-to-speech generation will be described in section 3.3. The graphical user interface will be explained in section 3.4 and section 3.5 will go into detail about the interaction with the robot. Finally, a component to generate the math problems based on the user's level will be explained in section 3.6.

3.1 Facial recognition

As the program starts and the student is seated in front of the camera, ten pictures are taken in order to identify them by their facial features. The program identifies the student as the person that it was classified as most often with a confidence that was higher than the threshold. The threshold is set according to the amount of people that are in the database as this method was found to recognize people more often than a set, predefined threshold. Thus, the more people there are in the database, the lower the threshold. If the classification was most commonly lower than the threshold, the student is classified as unknown, after which the student's name is asked and ten new pictures are taken. The student gets to approve each of the ten pictures, which are added then to the database as training data.

The face recognition component is based on OpenFace [Amos et al., 2016]. The faces are detected in an image using Histogram of Oriented Gradients (HOG) [Dalal and Triggs, 2005], which works by converting the image to its HOG image and finding faces in the images by finding HOG parts of the image that look like similar HOG training images. Figure 3 shows an example of the result of a HOG image.

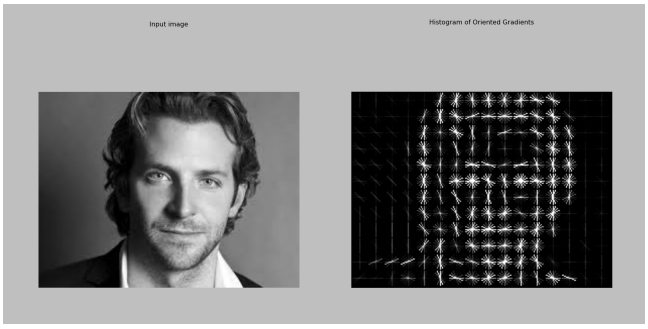


Figure 3: Result of a HOG image.

Secondly, face landmark estimation [Kazemi and Sullivan, 2014] is used to find

the exact position of the eyes and mouth in a detected face. This estimation is used to align the detected faces using affine transformations, after which a 128-dimensional representation of the face is formed with a Deep Convolutional Neural Network [Schroff et al., 2015]. These representations are used to classify the face in an image, which is done using a linear support vector machine.

3.2 Speech recognition

Speech recognition is used to recognize how a student responds to a provided problem. This means that the recognizer needs to detect whether the user gave any answer at all, the answer was correct or incorrect or the user was saying that it did not know the answer. The speech recognition is done using Google Cloud Speech API[3], which uses a threshold to detect the boundary between background noise and actual speech. Using the appropriate threshold, the tool returns an assumed sentence from the speech fragment. This sentence needs to be scanned for the presence of numerical values, which have to match the solution to the problem that was given to the student.

Google's speech recognition makes use of deep learning. Gaussian Mixture Models and Deep Neural Networks were used for the speech recognition. More specifically, they used Recurrent Neural Networks[4] (RNN) for this task. An advantage of this kind of neural net for this specific problem is that inputs and outputs can be cycled, and thus sequences of in- and outputs can be stored in an internal memory. The sequential nature of this neural net makes it highly applicable for sequential tasks such as speech recognition. However, a significant problem of regular RNN's is the *vanishing gradient problem*, which causes the gradient to decrease exponentially.

A large improvement was therefore made by making use of a Long Short-Term Memory (LTSM) network, which is a variant of a RNN architecture that does not carry the vanishing gradient problem. Google conducted experiments to evaluate the performance of the following neural network architectures on a large speech recognition task: DNN, RRN and LSTM. They extended the LSTM and did experiments between the standard LSTM and their extended version of the LSTM. The conclusion was that for a large speech recognition task the extended version of LSTM worked better than the standard LSTM. The extended version also worked better than the DNN. [Sak et al., 2014]

A simple recurrent neural network has an input layer, a hidden layer also known as the context layer, and an output layer. The network therefore has not only the input layer as input but also the context layer. The context layer is the input of the previous state. This means at time t the input is the layer at time t but also at *t-1*.[Mikolov et al., 2010] However, RNNs cope with the problem of

[3] `https://cloud.google.com/speech/`

[4] `https://research.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html`

learning long-term dependencies. Sometimes more context is needed. [Bengio et al., 1994] LSTM resolves this problem by introducting memory blocks into the network. Google extended this by adding a recurrent projection layer and a non-recurrent projection layer. [Sak et al., 2014]

Furthermore, since complex text understanding is not within the scope of this project, the method that was used to detect whether a user is saying that it did not know the answer is to hardcode some sentences with this meaning and detect whether the speech fragment contains any of these hardcoded sentences.

A difficulty that was tackled in the speech recognition component was that the API does not give consistent output, meaning that numbers are not always represented using their numerical representations. This is difficult due to the number of ways Dutch numbers can be represented textually, e.g. 1540 can be said as 'vijftienhonderd veertig', 'vijftienhonderd en veertig' or 'duizend vijfhonderd veertig'. Therefore, a number-text dictionary was created, which holds the numerical representations of a range of numbers as key and all their possible dutch textual representations/synonyms as their value.

3.3 Text-to-speech

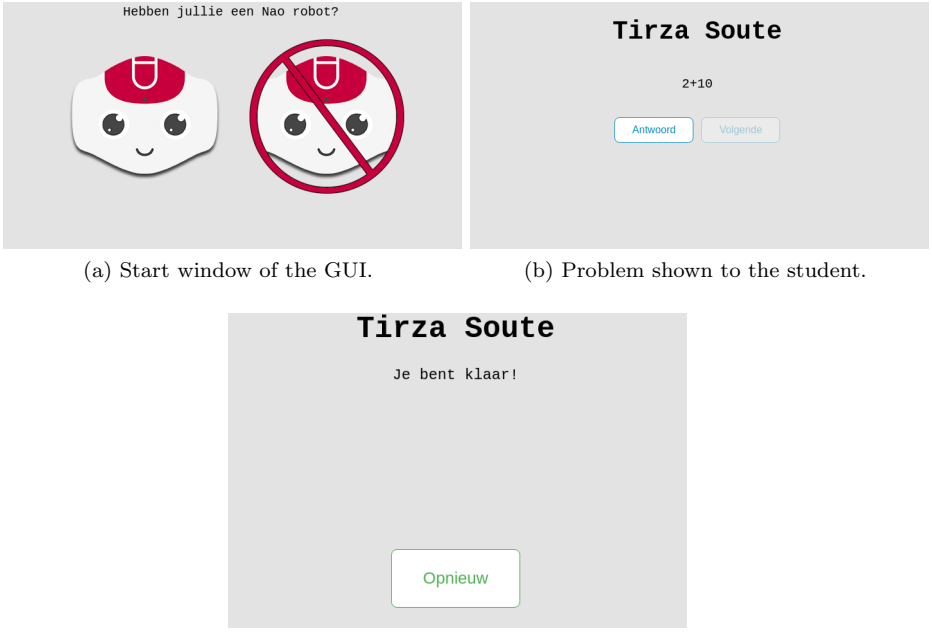
Text-to-speech generation is used to interact with the user. For example, it lets the user known whether their given solution is correct. Since the main goal of the text-to-speech generation is to have social interaction with the user, it should be able to communicate the same way people communicate with each other. For example, people normally greet each other when they meet or leave. The text-to-speech generation makes it possible for the program to conform to social conventions. Besides social interaction, it also says the generated problem out loud. The program was specifically designed for Dutch students and, therefore, the language of the speech is Dutch. The program can be used solely on computer or in addition with a Nao robot. When the program is used with a Nao robot, the built-in text-to-speech of the Nao is used. The following section will describe the text-to-speech techniques which are in use when the program uses solely the computer.

The tool used for the text-to-speech recognition is gTTS[5], which is a Python interface for Google's text-to-speech API. The technique behind Google's text-to-speech API is called *WaveNet* [van den Oord et al., 2016]. WaveNet uses a convolutional neural network with *autoregression* to predict sequential output values in speech, and thus using a completely probabilistic approach. Autoregression makes the assumption that an output value is a linear combination of its previous inputs. Therefore convolutional layers are used to take into account previous inputs. A problem with regular convolutional layers is that they challenge computational issues when their filters are large. However, large filters

[5] `https://pypi.python.org/pypi/gTTS`

are desirable to acquire a sufficient amount of inputs for prediction. Therefore WaveNet uses *Dilated Convolutional Layers*, which omits a certain number of inputs to reduce complexity.
A gTTS instance consists of a string and the language it should be pronounced in. The text is thus given as a string, after which gTTS converts the string to Dutch speech. The speech is saved as an mp3 file, which is used by mpg123[6]. This is an open-source audio player that can play the mp3 file. The sentences that are used to interact socially with the user, e.g greeting the user, are hard-coded. However, the sentences of the arithmetic problems are obtained through the problem generator. The output of the problem generator is used as input for the text-to-speech generator.

3.4 Graphical User Interface



(a) Start window of the GUI.

(b) Problem shown to the student.

(c) Program has finished.

Figure 4: Basic interface of the GUI.

The graphical user interface complements the speech communication that takes place through the NAO robot. Figure 4 shows the basic interface of the GUI. The program starts trying to recognize the student when the start button is clicked, which is shown in Figure 4a. If the student is recognized, the program starts the arithmetic problems and shows them to the user, which is displayed

[6] `https://www.mpg123.de/`

in Figure 4b. If the program does not recognize the student, the name of the student is asked. Figure **??** is an example of when an arithmetic problem is shown.

The main tool that was used for the GUI is Electron[7], which is an open-source framework that allows to build GUI desktop applications. It combines JavaScript, HTML, and CSS. The advantage is that the applications built within Electron are cross-platform application, meaning that the application can be used on Mac OS X, Linux, and Windows.

3.5 NAO robot

A NAO robot is used for the interaction between the student and the program. The camera of NAO is used to do face recognition and the speakers of the NAO robot are used to do text-to-speech. The decision to use a robot was made to create a more natural interaction for the student, since they get to interact with a robot instead of a computer. Furthermore, the robot is used to motivate the student as well. This is done using a congratulatory sentence and cheering motion if student completed it's session with success and a motivational sentence posture otherwise.

3.6 Problem generator

Instead of using an existing API for generating problems, such as Sowiso[8] or Khan Academy[9], it was decided to implement a problem generator from scratch. This way it is easy to alter the problem generator's settings any which way and at any given time. This is crucial since the requirements for the problem generator, which are stated below, might change over the course of time.

The objective of the newly implemented problem generator is to return a set of arithmetic problems that are adjusted to a child's arithmetic skills. If the student has trouble with multiplication, for example, these kind of questions are easier and asked more frequently than a subject that the student has already mastered. The frequency of each of the operators is based on the percentage of correctly answered questions per operator, using the following formula:

$$operatorProportion = (\frac{operatorCorrectnessPercentage}{totalOperatorsPercentage})^{-1}$$

The *operatorCorrectnessPercentage* is the percentage of correctly answered questions of an operator. The *totalOperatorsPercentage* is the sum of the

[7] `https://electron.atom.io/`
[8] urlhttps://sowiso.nl/
[9] `https://nl.khanacademy.org/`

inverse *operatorCorrectnessPercentage* of all operators. The operator proportion is calculated by taking the inverse of the proportion of one operator. The inverse is taken to make sure that an operator with a high correctness, is asked less often than an operator with a low correctness.

If a new student is added to the database, a list of operators and their scores is initialized. The initial operator is addition with a range from zero to ten. Only after the student has reached a certain level with this operator, the new operators subtraction, multiplication and division are added. The scores of each operator are kept separately by keeping track of the amount of correctly answered questions and the total amount of questions a student has answered.

Several things were added to increase the level of difficulty for a student:

- **Increasing the range of numbers used in questions** The range of an operator is increased by ten when at least twenty questions of that operator were answered after the last increment and a correctness of 80% was obtained.
- **Adding new operators** New operators are added when each operator was given for at least 40 questions and a correctness of 80% was obtained.
- **Increasing amount of components in a question** The amount of components is increased when at least 60 questions of that operator were answered after the last increment and when a correctness of 80% was obtained.
- **Combining multiple operators** Operators are combined when each operator had been given for at least 80 questions and a correctness of 80% was obtained. The range of the combination started at 0-10 and increased the same way as single operators.

The student's skills and an integer n that indicates the amount of questions to be generated are stored in a Student object, which is given to the problem generator. The problem generator randomly selects an operator from the student's skill set and two integers within the student's personal range. These are then made into an arithmetic equation. This process is repeated n times until the final problem set is returned by the generator.

4 Evaluation

To verify whether the components work correctly, qualitative testing is necessary to provide a stable indication of the accuracy of each individual component. The different components of the program are evaluated using precision, recall, and the F_1 measure. Precision measures how relevant the returned set is (1), while recall measures how many relevant objects were really recognized (2). The F_1-measure rewards strong recall and precision, while punishing every deviation towards neglecting one (3).

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{1}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{2}$$

$$f_1 = 2 * \frac{\text{recall} * \text{precision}}{\text{recall} + \text{precision}} \tag{3}$$

5 Results

This section will present the results that were obtained with the separate components. Firstly, in section 5.1, the results that were achieved using the face recognition component will be shown. Secondly, the results that were obtained using the speech recognition component will be shown in section 5.2. Finally, the obtained results of the text-to-speech generator will be shown in 5.3.

5.1 Facial recognition

To test the facial recognition component 10 persons were added to the database. The face recognition works well when the student looks into the camera and their entire face is visible clearly, both close and from far away. If a user who wears glasses takes them off, the recognizer can still identify them correctly. However, when their face is partly covered, e.g. their hand is in front of their nose, their face is not detected.

First, it was tested ten times whether each person was recognized as unknown when they were not in the database. Then, after they were added to the database and the program was trained again, it was once again tested ten times whether they were recognized correctly. Table 1 shows the results of the evaluation.

	True	False
Positive	84	25
Negative	77	14

Table 1: Evaluation of facial recognition.

From these values the precision is 0.77, the recall is 0.85, and the F_1 measure is 0.80.

5.2 Speech recognition

The speech recognition does not work accordingly all the time. The speech is only recognized correctly when pronunciation is clear and background noise is limited. These are usually not primary school circumstances, since young

children generally tend to articulate less well and background noise is usually unavoidable. These problems are due to the speech recognition software that is used. Plugging in an external microphone that is placed near the student's mouth, however, significantly improves the speed and understanding of the speech recognition module.

A data set of forty audio files of two different speakers was created to evaluate the program. The audio was recorded in a silent room using an external microphone. Each of the files contains a speech fragment of an integer between 0 and 1000. Table 2 shows the result of the evaluation.

	True	False
Positive	29	0
Negative	7	4

Table 2: Evaluation of speech recognition.

From the table, it can be concluded that the precision is 1, the recall is 0.88 and the F_1 measure is thus 0.94.

5.3 Text-to-speech

The text-to-speech API used works well. However, when pronouncing a $-$ it does not say 'min', but 'tot' instead. Furthermore, $*$ is pronounced as 'sterretje' instead of 'keer' and $\div$ as a fraction. For example, '$8:2$' is pronounced as 'acht tweede' instead of 'acht gedeeld door twee'. This was fixed by hardcoding the operators as the words they should be pronounced as. Also, the API does not allow the sound to be tweaked. For example, it is not possible to change the pitch. This results in some sentences not sounding as natural as others.

6 Discussion

This section will discuss the obtained results of the separate modules. Section 6.1 will discuss the the face recognition module, followed by a discussion of the speech recognition module in section 6.2. Furthermore, the text-to-speech component will be discussed in 6.3 and, finally, the problem generator will be discussed in section 6.4.

6.1 Facial recognition

Currently, the program just adds new users to the database, but does not train on new data yet. It would be better to start a new thread and train immediately so the next time the program starts, the new student is recognized immediately.

6.2 Speech recognition

The problems in the speech recognition tool are caused by the chosen software to recognize text in speech. A solution to this would be to use different software that is more flexible and can be trained to fit the needs of this program, e.g. return next best alternative sentences from speech to resolve ambiguity, being able to give hints so that the recognizer is biased to recognize certain words over others, which would be numerical values in this case.

6.3 Text-to-speech

The main goal of the text-to-speech is to have social interaction with the user. There are standard sentences used for different situations. However, this could feel rigid for the user. A different method that could be used to give a more social feeling to the interaction would be to use data of conversations between humans to learn how humans communicate.

6.4 Problem generator

The difficulty of implementing a problem generator that satisfies the requirements as stated in 3.6 was in deciding how and when to gradually increase the difficulty of the questions. There are many parameters, namely the different operators that are used, the combination of them and in what range the numbers lie to which they are applied.

Now that these difficulties have been tackled, the problem generator could be extended to give guided hints. Furthermore, advanced problems like simple mathematical equations could be added. This could be done easily without having to change much in the program, because the problem generator is its own class.

7 Conclusion

All components of the product work separately and together. The final product is complete and works as it should. Some components can be improved to optimise the interaction with the student, but, overall, a well-working basic product was developed. The research question that was asked was how student performance in basic primary school skills can be increased by providing personalized assistance with expert knowledge, using present-day multimedia technology. The hypotheses were that students would perform better on these arithmetic problems using the created tool and that the virtual environment would be a favoured work environment. The research question and the hypotheses can be answered by doing tests that utilise Personal Teaching Aid.

References

Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Christian Brühwiler and Peter Blatchford. Effects of class size and adaptive teaching competency on classroom processes and academic outcome. *Learning and Instruction*, 21(1):95–108, 2011.

Kai-Yi Chin, Zeng-Wei Hong, and Yen-Lin Chen. Impact of using an educational robot-based learning system on students' motivation in elementary education. *IEEE Transactions on learning technologies*, 7(4):333–345, 2014.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Jeong-Hye Han, Mi-Heon Jo, Vicki Jones, and Jun-H Jo. Comparative study on the educational use of home robots for children. *Journal of Information Processing Systems*, 4(4):159–168, 2008.

Heather C Hill, Brian Rowan, and Deborah Loewenberg Ball. Effects of teachers' mathematical knowledge for teaching on student achievement. *American educational research journal*, 42(2):371–406, 2005.

Vahid Kazemi and Josephine Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. URL `http://arxiv.org/abs/1402.1128`.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

Chien-Hung Liu Shih-Wei Chou. Learning effectiveness in a web-based virtual learning environment: a learner control perspective. *Journal of Computer Assisted Learning*, 2005.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.