Delft University of Technology

Conversational Agents
CS4270

Group Project Final Report

Authors:
Doreen Mulder (4575385)
Laura Ottevanger (4584953)
Stefan Petrescu (5352150)
Rohan Sobha (4565487)

October 30, 2021

1 Introduction

In this report we introduce, discuss, and elaborate on MATHew, a conversational agent created to assist children between the age of 10 and 12 in learning basic mathematics. MATHew is embodied by the furhat robot [AS17]. Furhat is a back-projected face that can emote and talk. MATHew also uses the speech recognizer that Furhat provides. This report provides some information on how MATHew has been implemented and tested. The GitHub repository for MATHew can be found here: `https://github.com/spetrescu/tudelft-conversational-agents-2021/`. Because the repository is private, access can be granted by Stefan Petrescu[1], please request access if you wish to access the repository.

2 Motivation

Where a few decades ago conversational agents only existed in science-fiction, today they are found virtually everywhere. The recent increase in technological capabilities have made these systems more elaborate, giving them more abilities to handle user requests, saving considerable time and effort, and because of this also money, compared to when a human were to perform these tasks.

We envision MATHew to play a role in giving elementary school-aged children equal opportunities when it comes to education [MOV21]. It is known that social class inequality has a significant effect on children's performance in school. Children of well-off parents often have access to tutoring out of school hours, giving them more opportunities to catch up on school materials when they don't understand something or fall behind on class material. This is not standard for children from lower income families, creating a gap of inequality in the classroom. We propose to use MATHew, our free to use math tutor agent, as a first step to close this gap.

But why would we use a socially aware dialog system as opposed to a more simple solution? A socially aware dialog system will respond to the social cues of the user, making the conversation feel more natural. We aim to make the user feel more comfortable and heard through this intuitive manner of interaction. We also hope that MATHew engages the user more, and create a more active participation from the user as opposed to non-socially aware methods.

3 Implementation

We wanted to move away from rule based approaches. Thus, although our system has paths that are rule-based, we offer the possibility for users to interact with an agent capable of doing Natural Language Understanding (NLU), Dialog State Tracking (DST), Dialog Policy (POL), and Natural Language Generation (NLG). For the math tutor itself, we wanted to be able to let a student perform a test, practice questions together or get some explanation. The student is only able to talk more freely with the math tutor when the student asks for explanation. This was done this way to give the student some structure while practicing while still able to ask more questions when necessary. For the approach that navigated away from rule-based dialog design, we implemented just a proof of concept (the agent is able to assist users in reminding them about various math tasks).

Four major components are used to construct the final system: The dialog manager, a subsystem for gaze modeling, a system for emotion detection, and the task-oriented dialog system. We use Furhat [AMBSG12], since their software development kit contains excellent starting points for the development of our system.

3.1 Dialog Flow

The dialog flow was the first thing that was implemented. There are several features to the dialog flow: firstly, the student is greeted by the Conversational agent and it will ask for the name of the student. Once it knows their name, it asks which subject the student wants to practice. The student can choose between multiplication, division, percentages and fractions. Once a subject has been chosen, the student can decide what to do next.

[1]Email address: petrescu-1@student.tudeflt.com

There are multiple options: the student can do a test, practice a single question, let the agent do a question or ask for explanation about a subject. If a student wants to do a test, the agent will ask how many questions the student wants to do and what the difficulty should be. Once this is set up, the questions will be generated and asked to the student. After the student is done with these questions, the agent will generate a grade for the student. A student can also decide to do a single question, the agent will generate an easy question for the student. As the student answers the question, the agent will ask for confirmation if they understood their answer correctly.

These confirmations prevent the agent from misunderstanding answers. For example, if the agent heard "fifty" when the student said "fifteen", it will ask: "So your answer is fifteen, is that correct?". The student now gets the chance to notify the agent of the mistake and to repeat their answer.

Last but not least, in order to access the 'Natural Language Processing' branch, the users can simply utter at the beginning of the conversation, any of the following words/sentences: 'task-oriented dialog', 'task dialog', 'oriented dialog', 'task'. At the time being, this branch was implemented as a proof of concept, as its tutoring capabilities are not the greatest – what it currently does is to remind users of possible math related topics (like a reminder). It is however, fully functional and seamlessly integrated within the code base.

3.2 Emotions

In order to have the conversation feel more realistic, we decided to incorporate emotion detection into our system's capabilities. Looking at relevant resources we adapted [Bal19] into a model capable of recognizing 4 emotions from a webcam feed: Happy, sad, frustrated, and neutral. In order to communicate with the agent, we implemented a ZMQ publish/subscribe server that communicates the user's emotions in real time.

We also send our user's spoken text to this emotion server and perform sentiment analysis on it. This is done using TextBlob [Lor20]. The server returns a polarity value back to the agent, ranging from -1 to 1, with -1 being negative and 1 being positive.

We ensure that the detected emotions are correct by combining the output detected from the camera with the output from sentiment analysis. For example, we only set the user's current emotion as "happy" when the camera detected "happy" and the polarity value of their last spoken text is equal to or greater than 0. Based on the the different emotions detected we take relevant actions.

These actions are described in `EmotionHandler.kt`, in the `performGesture` method. We implemented 6 different emotional responses: happy, confirm, uplifting, encouraging, calming, and neutral. Each emotional response alters the pitch, rate, and volume of the agent's voice, as inspired by the "Speech Features and Emotions" slide from the Affect lecture. Every emotional response, with the exception of the neutral response, also performs a gesture. This gesture is picked randomly from a predefined set of gestures. Additionally, we created a new "persistent smile" gesture for the happy emotion, which has its facial expression last longer than the default smiles that come with Furhat.

By default, the agent speaks in a neutral way, displaying microexpressions throughout the conversation. The users emotions are taken into account during a few stages in the flow, namely when the user answers a practice question, and when the user receives their grade after doing a practice exam.

MATHew then responds as follows:

- Happy when the user is happy
- Encouraging, then uplifting when the user is sad
- Calming, then uplifting when the user is frustrated
- Encouraging when the user gets the answer(s) wrong, and the user's emotion is neutral
- Uplifting, when the user gets the answer(s) right and the user does not look happy

Responses that consist of multiple steps (encouraging then uplifting and calming then uplifting) feel more natural than the default voice since the rate, volume and pitch of the voice are altered in-between the steps, giving the agent's voice some intonation.

3.3 Gaze Aversion

MATHew also includes modeled gaze aversion implemented through three conversational functions [AMG13]:

- cognitive
- intimacy
- turn-taking

These functions were manually labeled in our code as tags, based on the type of utterance from MATHew. They are placed right before an utterance to simulate looking away before speaking. The tagging implies that the gaze is then modeled after its respective distribution for both the gaze aversion duration and direction.

Each of these functions has their role in gaze aversion. Firstly, the **cognitive** tag implies that the agent has thought consciously of their answer before replying. Secondly, **intimacy** is used to draw the user's attention, such as in a greeting or saying farewell, but also in expression emotion to the user such as encouraging the student to do better. Lastly, **turn-taking** is used whenever the agent is expecting the user to speak such as asking questions or other ways of evoking responses.

Tagging an utterance can be exemplified by calling 'furhat.gazing(tag: ConversationalFunction)' that retrieves an appropriate gaze direction and duration. The gaze aversion is then executed before the `furhat.say()` call. The tags have hence been applied in every dialog state where MATHew is expected to speak. The aforementioned function is described in `gaze.kt`.

In general, there exists a correlation between the duration of gaze aversion and a conversational function which is represented as a Gaussian distribution. For the direction of gaze, an enumerated weighted distribution for looking up, down and to the sides was constructed based on the observation mentioned in the authors' paper.

3.3.1 Alternative method

In the current implementation of MATHew, there is no multi-modal input fusion that determines the gaze. That is to say, the gaze is expressed independently of audio signals, semantic analysis of words or turn-taking triggers.

The gaze lab assignment provided preprocessed data from the IFADV [vSWS+08] corpus. This data contains a mapping for each turn in a conversation to one of six discrete categories of turn-taking [OWE+12]. This method was considered for the implementation of gaze initially, but the lack of Furhat's capabilities rendered this method infeasible. Modeling gaze in this way requires to be able to determine who (i.e. Furhat, the user or both) is speaking every 10ms and taking into account the semantics of what is being said by the user. For example, an 'uh, huh' (backchannel) versus an 'ehm...' (indication of switching turns) makes a significant difference for how to model the gaze. As Furhat lacks the ability to monitor turn-taking continuously over a time series, the current implementation of gaze (aversion) has been designed to be independent of these modalities.

3.4 Task-oriented Dialog using Transformers

This branch of our system represents our intention to move away from rule-based dialog design. We managed to implement a proof of concept, more specifically, we trained a machine learning model able to assist users in accomplishing specific (math-related) tasks. We were inspired by the existing work on 'task-oriented dialog' agents. For more context, a task-oriented dialog is a "dialog system that aims to assist the user in completing certain tasks in a specific domain, such as restaurant booking, weather query, and flight booking, which makes it valuable for real-world business" [ZTZ+20]. Starting from a pretrained model, we used transfer learning in order to fine-tune the model for our domain related tasks. In the following subsections, we will first start by providing more context – Additional Context, and follow up with our implementation process – Implementation.

3.4.1 Additional Context

As we wanted our system to have more flexibility, we tried to navigate away from rule-based dialogues. We attempted multiple times to incorporate this into our system, and, more specifically, we actually tried to implement 3 papers into our system's already existing structure. Unfortunately, due to time constraints (and other reasons, such as lack of compute resources or appropriate hardware) we weren't able to incorporate all three. However, we managed to incorporate one of the three. We tried to illustrate this process in Figure 1.

We initially started by looking at Reinforcement Learning approaches for dialog systems. However, although we found lots of interesting resources, not all of them had readily available open-source implementations. Because we were also sort of time constrained, we looked for papers that had readily available code (and obviously, some had more detailed guidelines, compared to others). As our 'literature review' directed us towards 'task-oriented dialogues', we decided to try our best to incorporate some of the concepts into our system[2].

[TLH20] was our first approach. We spent, compared to the other two papers, the least amount of time in order to apply it to our assignment. The reason for which we did not focus too much on actually getting the implementation up & running was due to the fact that the paper itself, although very interesting, had little applicability to what we were trying to do. However, we feel like the hours spent understanding the codebase were not for nothing, as we walked away with having a better understanding of what we were after.

[TDKB21] was 'the one that got away'. Unfortunately, although we thoroughly tried to implement this (knowing in advance how difficult it'll be), we were unable to. We were actually in contact with the paper's authors, trying to get some hints regarding very specific details of the codebase. However, it wasn't all in vain, because, by going back and forth with the authors, they actually recommended one paper that was the one that we managed to integrate in the end. In comparison to our first approach, the second one would have been more applicable to what we were trying to go after, as it had the possibility of doing transfer learning to an already-trained agent.

[PLL+20] was the paper that we managed to add to our system. We believe that, although, this paper was the hardest one to understand, its authors did a great job by thoroughly describing the development process of using their approach. Not only this, but they also provide a well-documented code base, which was the starting point for our agent. Although we know that the agent itself (as a math tutor) is not the most useful thing out there, we are happy with our attempt at understanding how to go about incorporating these types of complex ideas into a project.

Thus, in the following subsections, we will discuss about the technical details of the current dialog branch, its software engineering & data science aspects.

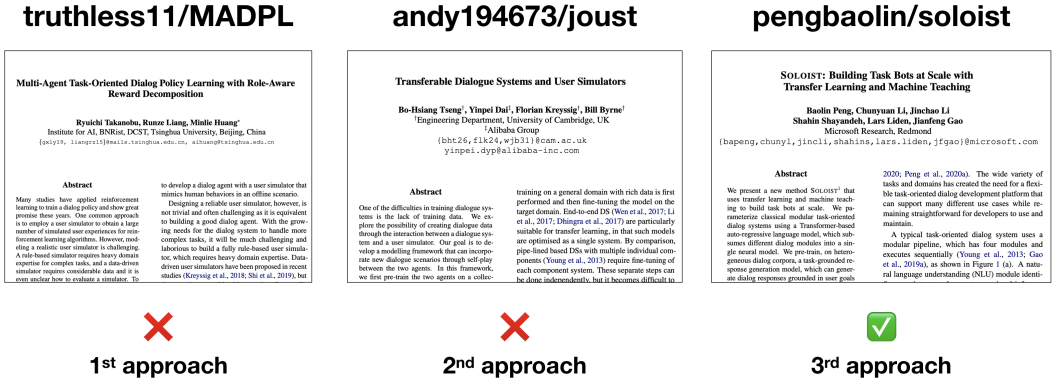


Figure 1: The three papers that we tried to implement. From left to right, [TLH20], [TDKB21], [PLL+20].

[2]Little did we know that the process of actually getting something to work will be so grueling

3.4.2 Implementation

In this section we will describe technical aspects reagarding the implementaion[3] of the 'Natural Language Processing' branch. We will start by discussing the software engineering related challenges and follow up with the process of training and using the model.

The Software Engineering Behind our System

We feel the need to mention that, although this may not be as relevant to the grading rubric as other things, we also tried to address the software engineering challenges encountered during this project. An overview of our system can be seen in Figure 2. If we were to summarise which were the main software engineering challenges, we would categorise them as follows:

- Refactoring the SOLOIST[4] codebase (component 5 in Figure 2) in order to create an effective way in which the user and system communicate. This meant refactoring the project's existing interface (that was the way in which the text input was being passed to and retrieved from the neural model). This way, we managed to create a way in which the user (speaking from Furhat) gets responses from the model (but also uttered by Furhat). Consequently, we seamlessly integrated the Transformer neural model (and all of its accompanying software) , making it seem like the user is having a conversation with Furhat itself.
- Creating the links between all the components required a significant amount of work – keep in mind that half of our codebase is in Python, and the other half in Kotlin (not to mention JavaScript which was the language with which the initial interface of SOLOIST was created). This is not necessarily ideal, but we had to cope with these aspects and overcome all technical challenges. Last but not least, having a model that inferred dialogues meant that we had to use blocking methods, which required lots of attention to details.

In the end, our system turned out more complex than we expected (in regards to the software engineering aspect). Due to this, we spent a significant amount of time making sure all the components were able to run together. Still, we enjoyed the challenge[5].

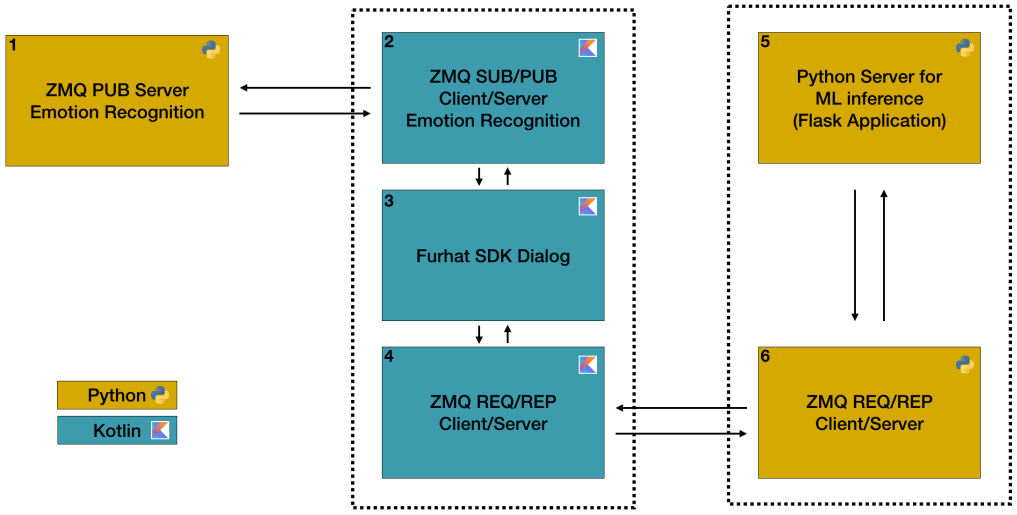


Figure 2: Our system's architecture consisting of 6 main software components; From left to right: (1)ZMQ PUB Server – its main role was to broadcast the inferred user emotions; (2)ZMQ SUB/PUB Client/Server – responsible for the sentiment analysis part; (3)Furhat's SDK – which binds everything together (Client/Server functionalities); (4)ZMQ REQ/REP Client/Server – communicates user dialog input for model inference; (5)Python Server (Back-end in Flask) – responsible for handling the interaction with the Transformer (our neural model); (6)ZMQ REQ/REP Client/Server – responsible for communicating the inferred dialog to Furhat.

[3]Access to the entire NLU branch can be obtained just by downloading the following archive: https://drive.google.com/drive/folders/1Wtz3tGZtTG6u3i7GEy3bKG6dAmbsICF0?usp=sharing. Keep it mind that the archive is about 4GB. Instructions on how to go about running the models can be found in our GitHub repository.

[4]Code available at: https://github.com/pengbaolin/soloist

[5]Code available at: https://github.com/spetrescu/tudelft-conversational-agents-2021

Training and Using the Transformer Neural Model

First off, we have to mention that we wish we had more time to actually do more data science related tasks. Unfortunately, we were unable to do so. However, we tried our best to understand what the paper did. Also, we tried to get the most out of this experience by actually training the model ourselves; we used GoogleColab (as our machines did not have CUDA compatibility) which we feel like helped us map out a little bit more of the unknown.

Because of the already existing infrastructure, having to train the model was not as hard as we would have expected. Just by running the SOLOIST guidelines we were able to get the model up and running. What we mean by this, is that it took a short amount of time until we had a model able to infer dialog information. All props to SOLOIST's authors for doing such a great job.

For future directions, we are seriously thinking of making our agent more useful. We know that the few-shot learning procedure would have been very good if we would have used our own training data. However, due to the fact that there were no guidelines for this (on how to go about creating a new ontology & new data set compatible with the framework), it would have been infeasible (due to the lack of appropriate time resources). Additional implementation details and recommendations can be found in our GitHub repository.

4 Results

4.1 Questionnaire

We conducted a short user study with one person from outside our group, after which the participant filled out the Networked Minds Social Presence Measure as described in [HB04]. The filled-in questionnaire can be found in Figure 4.

It is notable that the participant scored our system significantly lower than the mean value on both perceived emotional interdependence and perceived behavioral interdependence. During the debriefing the participant explained that they were hard of hearing and needed to rely on subtitles in the Furhat dashboard in order to make use of our system. For this reason, most of the behavior of the agent went unnoticed. This is an unexpected but interesting result which rises the question on how we can create better conversational agents that are usable for people who are hearing impaired.

4.2 Average accuracy of system's affect detection

For the average accuracy, we created a small test set of 20 pictures of ourselves expressing various emotions. We calculated the accuracy as the number of correct predictions over the total number of predictions. With 12 out of the 20 test images being correct, we obtain an accuracy of 0.6. The emotion detection works pretty consistently. We think this is partially caused by the fact that we only detect 4 emotions. All four of them are easily exaggerated when asked to be acted out in front of a camera, and thus, easier to pick up for the neural network. The system mostly goes wrong when it detects a second face when there is only one face in frame.

4.3 Additional user testing

We performed some additional user testing to test the system even further.

4.3.1 Testing plan

In our original testing plan, we wanted to test MATHew on the target audience (also see 6.2.1) and on adults. Because it is quite a difficult process to use children in scientific experiments, we decided to only test MATHew on adults for this paper. The original plan to give them a homework sheet and to make them pretend to make the homework was not entirely feasible to execute as it was difficult for the adults to pretend that they didn't know a certain basic math topic. The eventual test plan was the following: We let the adults walk through all of MATHew's features and then gave them the SASSI [Mil97] questionnaire and some additional questions. The additional questions were the following:

Questions with a rating scale from 1 to 7:

- The system, as is, is useful in helping people with math-related questions

- The explanation of the agent on the topics was clear to me
- I think that children would know how to use the system in a way that helps them with homework-related questions

Open questions:

- Did you encounter gaps in the dialog flow? If yes, at what point?
- Did you encounter any crashes, obvious errors, or other bugs when using the system? If yes, please describe them.
- Describe the interaction you were most satisfied with
- Describe the interaction you were least satisfied with
- Were there any features to the agent that were too distracting/annoying?
- Do you have any additional tips/improvements?

4.3.2 Test participants

There were 9 adults that participated in the user test. They range in age from 22 to 28, all of these adults are/have been students at a University.

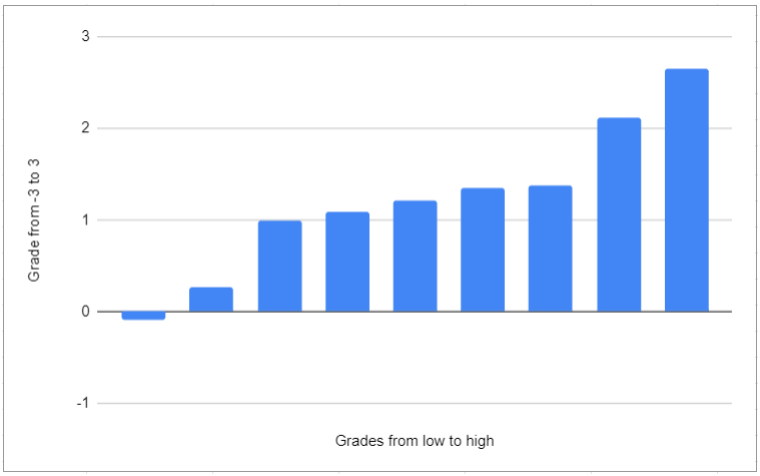


Figure 3: Average grade given by participants from low to high

4.3.3 Results

In the SASSI test, a person chooses a number that corresponds with if they agree with the statement. An 1 is means Strongly Disagree and 7 means Strongly Agree, 4 is neutral. We correspond the 1 to the value -3, 4 to 0 and 7 to 3. Furthermore, it is important to multiply the value with -1 if the question is a negative question. In Figure 3, the average grade per participant is shown. The averages are put from low to high. As you can see, only one person scored the system negative and two people were really positive, the rest were slightly towards the positive side.
The test participants were the most positive on the question: It's easy to learn to use the system. The question that was answered the most negative was the following: The interaction with the system is repetitive.

The open questions gave us very useful information. Especially the question: "Did you encounter any crashes, obvious errors, or other bugs when using the system? If yes, please describe them.". We noticed that the test that we had, still had some bugs, for example, if you said that you wanted 0 questions, it would break, the test result was wrong and it could suddenly crash. We also had a lot of test participants that complained about the speech recognition. They noticed that when they tried to say the word three, it would recognize tree and if they would answer four, it would recognize for. This made the test subjects a little more frustrated.

The final question was about what we could do to improve the agent. It is clear by the answers of the participants that the participants were frustrated by the repetition of the agent. When the agent doesn't understand what the user is saying, it will repeat the question. It takes a very long time to repeat a sentence and because the user is unable to interrupt, the user needs to wait a long time. It takes a long time for the interaction to take place.

5 Discussion

5.1 Limitations

5.1.1 What did not work and why do you think that is?

We wanted for the bot to be able to be interrupted by the user. The agent currently repeats all the subjects and all the options for the student to do. The bot also repeats the question when he didn't understand the user or if the user said something that is not a valid answer. This takes a very long time and a user easily gets annoyed by waiting on the bot to talk. We wanted to be able to allow the user to interrupt the agent. This unfortunately didn't work because the bot would be too late in listening to the user. It would miss important parts of the answer of the student as answers to questions are usually only one word.

We also recognize that the sentiment analysis needs more work. In its current state it only picks up on very clear sentences such as "I am very angry at you", which Mathew isn't able to respond to. In math-related conversation, the outputted polarity is always neutral, which is to be expected. We would have liked to use the sentiment analysis to handle extreme sadness or frustration in users, and create appropriate responses for these situations.

5.1.2 What are the strengths of your system and where are the weaknesses?

The strengths of our system are its predictability and diversity in training as a student can receive tutoring and practice their homework material in a variety of methods. Another strength is the improved emotion recognition system that determines the mood of the student and helps the student by trying to turn the mood positive. The biggest strength of our system is the fact that it isn't solely rule based but is a task-oriented dialog system so it is more flexible. We also see the emotional capabilities of our agent as a strength. The differences in voice when acting in a certain emotion are subtle, yet feel natural and add to the experience of having an engaging conversation

A weakness in the system is the repetition that it performs. For example, when the bot doesn't understand an answer, it will repeat the question until the user answers in way that the agent can understand. This can cause it to repeat a question over and over which is really frustrating and makes the agent look a lot more artificial and less like a human.

6 Conclusion

6.1 Results

After this project, an agent has been created that supports a student in learning some basic mathematical skills. The agent should assists and explains the student while the student is working on their homework. We developed a system that is capable of recognizing user behavior, and respond according to the spoken text sentiment and perceived user emotions from the camera feed. The agent is able to generate emotional responses, consisting of altered speech rate, pitch, and volume, as well as a gesture. In addition, this system is capable of gazing away from the user before it speaks for some amount of time and in random directions to simulate a more human-like feel of engaging with MATHew.

6.2 Future work

6.2.1 Additional user evaluation

Currently, we only tested the agent on adults. This is not entirely the target audience as we would like to focus on elementary school students aged between 10 and 12 years old. To test the agent

on the target audience, we have the following testing plan in mind.

We want to discover the likability and naturalness as perceived by our target audience, elementary school students aged between 10 and 12 years old. Usability and perceived effectiveness are measured with this target group as well, but the same experiments are repeated with a group of experts. We aim for this expert analysis to identify gaps in the dialog flow and other flaws within the system. Both students and experts are debriefed after each session, during which notes are taken by the researchers. Finally, either a questionnaire is filled in, or an interview is conducted.

The basic setup of the research can be described as follows. There is one session to identify a face and voice combination that is preferred the most by students. When the final combination is selected and implemented into the agent, we move on to identifying the usefulness and effectiveness of the assistant. This study is performed with two study groups: one being an expert evaluation, the other again performed with students.

To perform the evaluation, we will use the SASSI test [Mil97]. The SASSI test is a very broad, already existing test to measure a conversational agent. Because the SASSI test is not targeted towards the age of our target audience, we will simplify the questions to ensure that the target audience will have no difficulty in answering the questions. We also formulate our own questionnaire that relates more to the desired behavior we want the conversational agent to show, and gauges usability and effectiveness by relating more to the purpose of the agent: assistance with math-related questions. For example, questions may be asked to gauge the quality of the explanation and the difficulty of the questions. Both questionnaires can be filled in by the participant. This is either done on their own or by means of an interview.

Finding the preferred face
To make the agent more natural, a certain face and voice should be chosen. There are a lot of possible faces and voices that can be chosen. We propose the following plan for a user test.

First of all, a selection of face-voice combinations is picked by the researchers. A small standard program is written to go with these combinations. This program is the same for each combination: the bot can answer one pre-programmed question. The dialog flow may be rephrased to fit the 'speaking style' of the combination better.

The different bots are set up in one room. The participants, students aged 10-12 years old, are put in small groups, given a quest, and receive a small prize once the quest is completed. Each group receives a list of questions, and has to find the corresponding agent to get an answer to their question. The last agent will then guide the group of children to their prize.

After this treasure-hunt style of interaction, children are debriefed with an interview. Which agent assisted them the best, and which conversation did they like best and why? The combination that scores highest will be chosen to be the face and voice of the math tutor.

Effectivity of the agent
Finally, we want to evaluate the effectivity of the agent on the target audience. For this, we also had an experiment in mind.

The study participants will be the target audience; students from 10 to 12 years old. They will receive a sheet of math homework before and during the conversation with the agent. Participant are asked to first fill in the first sheet of math homework without the tutor. Participants are then requested to make use of MATHew to assist them with the second sheet of the homework. The sheets will be the same difficulty. Comparing the sheets for every student will give us data on the effectivity of the agent.

References

Samer Al Moubayed, Jonas Beskow, Gabriel Skantze, and Björn Granström. Furhat: A back-projected human-like robot head for multiparty human-machine interaction. In Anna Esposito, Antonietta M. Esposito, Alessandro Vinciarelli, Rüdiger Hoffmann, and Vincent C. Müller, ed-

itors, *Cognitive Behavioural Systems*, pages 114–130, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

Sean Andrist, Bilge Mutlu, and Michael Gleicher. Conversational gaze aversion for virtual agents. In Ruth Aylett, Brigitte Krenn, Catherine Pelachaud, and Hiroshi Shimodaira, editors, *Intelligent Virtual Agents*, pages 249–262, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Henny Admoni and Brian Scassellati. Social eye gaze in human-robot interaction: a review. *Journal of Human-Robot Interaction*, 6(1):25–63, 2017.

Atul Balaji. Emotion detection using deep learning. https://github.com/atulapra/Emotion-detection, 2019.

Chad Harms and Frank Biocca. Internal consistency and reliability of the networked minds measure of social presence. 2004.

Steven Loria. Textblob. https://textblob.readthedocs.io/en/dev/index.html, 2020.

FG Miller. Sassi: Application and assessment for substance-related problems. *Journal of Substance Misuse*, 2(3):163–166, 1997.

Stichting MOVE. Gelijke kansen voor ieder kind. https://www.stichtingmove.nl/missie-en-impact/opinieartikelen/kansenongelijkheid/, 2021.

Catharine Oertel, Marcin Włodarczak, Jens Edlund, Petra Wagner, and Joakim Gustafson. Gaze patterns in turn-taking. In *Thirteenth annual conference of the international speech communication association*, 2012.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. Soloist: Building task bots at scale with transfer learning and machine teaching. *arXiv preprint arXiv:2005.05298*, 2020.

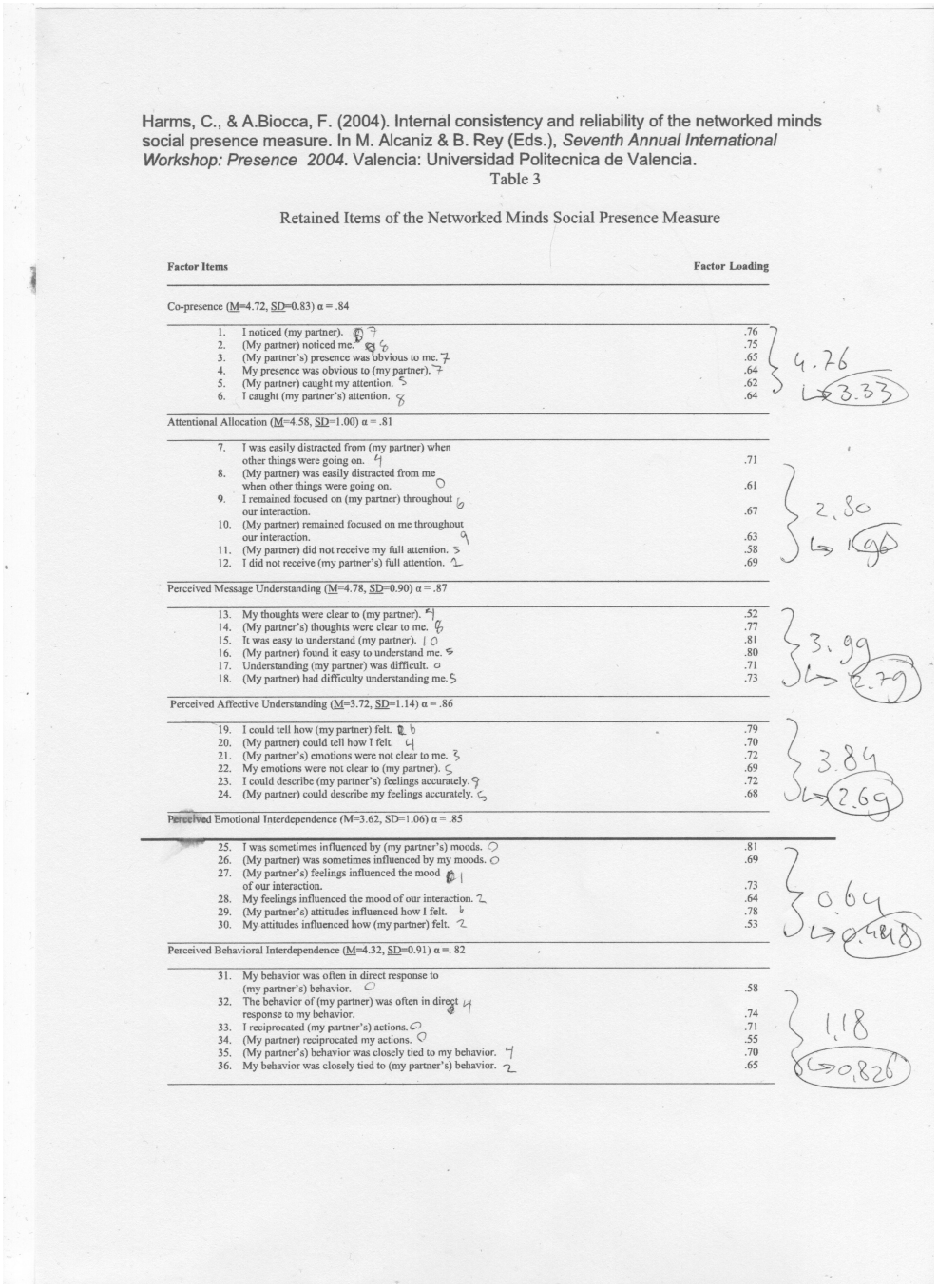
Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. Transferable dialogue systems and user simulators. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 152–166, Online, August 2021. Association for Computational Linguistics.

Ryuichi Takanobu, Runze Liang, and Minlie Huang. Multi-agent task-oriented dialog policy learning with role-aware reward decomposition. In *ACL*, pages 625–638, 2020.

Rob van Son, Wieneke Wesseling, Eric Sanders, Henk van den Heuvel, et al. The ifadv corpus: a free dialog video corpus. In *LREC*, pages 501–508, 2008.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, Minlie Huang, and Xiaoyan Zhu. Recent advances and challenges in task-oriented dialog system, 2020.

A Networked Minds Social Presence Measure



Harms, C., & A.Biocca, F. (2004). Internal consistency and reliability of the networked minds social presence measure. In M. Alcaniz & B. Rey (Eds.), *Seventh Annual International Workshop: Presence 2004*. Valencia: Universidad Politecnica de Valencia.

Table 3

Retained Items of the Networked Minds Social Presence Measure

Factor Items	Factor Loading
Co-presence (M=4.72, SD=0.83) α = .84	
1. I noticed (my partner).	.76
2. (My partner) noticed me.	.75
3. (My partner's) presence was obvious to me.	.65
4. My presence was obvious to (my partner).	.64
5. (My partner) caught my attention.	.62
6. I caught (my partner's) attention.	.64
Attentional Allocation (M=4.58, SD=1.00) α = .81	
7. I was easily distracted from (my partner) when other things were going on.	.71
8. (My partner) was easily distracted from me when other things were going on.	.61
9. I remained focused on (my partner) throughout our interaction.	.67
10. (My partner) remained focused on me throughout our interaction.	.63
11. (My partner) did not receive my full attention.	.58
12. I did not receive (my partner's) full attention.	.69
Perceived Message Understanding (M=4.78, SD=0.90) α = .87	
13. My thoughts were clear to (my partner).	.52
14. (My partner's) thoughts were clear to me.	.77
15. It was easy to understand (my partner).	.81
16. (My partner) found it easy to understand me.	.80
17. Understanding (my partner) was difficult.	.71
18. (My partner) had difficulty understanding me.	.73
Perceived Affective Understanding (M=3.72, SD=1.14) α = .86	
19. I could tell how (my partner) felt.	.79
20. (My partner) could tell how I felt.	.70
21. (My partner's) emotions were not clear to me.	.72
22. My emotions were not clear to (my partner).	.69
23. I could describe (my partner's) feelings accurately.	.72
24. (My partner) could describe my feelings accurately.	.68
Perceived Emotional Interdependence (M=3.62, SD=1.06) α = .85	
25. I was sometimes influenced by (my partner's) moods.	.81
26. (My partner) was sometimes influenced by my moods.	.69
27. (My partner's) feelings influenced the mood of our interaction.	.73
28. My feelings influenced the mood of our interaction.	.64
29. (My partner's) attitudes influenced how I felt.	.78
30. My attitudes influenced how (my partner) felt.	.53
Perceived Behavioral Interdependence (M=4.32, SD=0.91) α =. 82	
31. My behavior was often in direct response to (my partner's) behavior.	.58
32. The behavior of (my partner) was often in direct response to my behavior.	.74
33. I reciprocated (my partner's) actions.	.71
34. (My partner) reciprocated my actions.	.55
35. (My partner's) behavior was closely tied to my behavior.	.70
36. My behavior was closely tied to (my partner's) behavior.	.65

Figure 4: The filled-in Networked Minds Social Presence Measure questionnaire