Instructions for ACL-2016 Proceedings

Tom Byars, Cale Clark, Charlie Lyttle, Katie McAskill, Jack Miller, Zein Said, Abdullah Sayed, Laura Schauer, Jason Sweeney, Aron Szeles, Xander Wickham
School of Mathematics and Computer Science, Heriot-Watt University, Edinburgh
tjb10, cc164, cl157, klm12, jjm7, zs2008,
as512, lms9, js418, as472, aw127@hw.ac.uk

Abstract

This should be a 6-8 page conference paper with appendices, if relevant. Good reports from last year: 1 and 7

1 Introduction

from coursework spec: main research or technical question addressed
Socially assistive robots (SARs) are a crucial part of the future of many sectors, for example, education or healthcare (Gunson et al., 2022). Especially the latter depends on technology advancements as it is facing numerous obstacles in the future, such as increasing spendings and a growing percentage of older people. A serious lack of healthcare workers is already occurring, with 10 million more healthworkers needed worldwide by 2030 (Cooper et al., 2020; WHO, 2023). SARs can pose a solution to the problem by supporting healthcare in various ways, such as encouraging older people to keep living independently for longer or reducing caregiver burden (Cooper et al., 2020).

Healthcare SAR scenarios require SARs to be able to handle multi-party interactions as it is likely that more than one person will interact with the system. Compared to handling dyadic (two-party) interactions, handling multi-party conversations includes more complex challenges, such as Speaker Recognition, Addressee Recognition, Response Selection (summarised in "who says what to whom") and coordination of turn-taking (Addlesee et al., 2023; Johansson and Skantze, 2015).

Include here what exactly we examined about turn-taking

In this work, we propose a conversational system using a model trained on multi-party human-human conversation data. We collected the data from recordings of special "Who wants to be millionaire?" episodes where two candidates collaborated to answer the host's questions.

Include results here.

2 Background

from coursework spec: literature review / related work, including a critical analysis of the field, and commentary on applicability of the technologies and methods used in emerging technologies and application areas

2.1 Socially Assistive Robots

For healthcare, as well as for any other sector, the difficulty of successfully designing SARs lies in creating robots that can effectively converse with humans and adhere to social norms (Moujahid et al., 2022). The more expressive a robot is, the more it will be perceived as intelligent, conscious and polite (Moujahid et al., 2022). To achieve such a positive perception, multiple parts need to be combined into one conversational system, such as the ability to carry out visually grounded as well as task-based dialogues, to perceive and discuss its environment and to chit-chat (Gunson et al., 2022).

The SPRING project conducts research on a SAR robot deployed in an eldercare hospital reception area (Addlesee et al., 2020). The conversational system is deployed on the humanoid ARI robot produced by Pal Robotics (Robotics, 2023). ARIs capabilities can be extended with custom AI algorithms, in the case of SPRING-ARI a visual perception system, a dialogue system, and a social interaction planner (Addlesee et al., 2020). While the SPRING-ARI system successfully demonstrates that task-based, social and visually grounded dialogue can be combined with physical actions, it still lacks the ability to handle

conversations with more than one person simultaneously (Addlesee et al., 2020).

2.2 Multi-party Human Robot Interaction

As stated before, the endeavour to create conversational systems becomes considerably more difficult when dealing with multi-party interactions (Addlesee et al., 2023). Especially turn-taking poses a central problem. It is defined as follows:

> The rules of turn-taking organize the conversation into turns, during which one of the participants has the right to speak while the others agree to listen. (Żarkowski, 2019)

In dyadic conversations, there are only two roles a participant can take: speaker or listener, hence it is clear when and to whom the turn is yielded. In multi-party conversations, participants can take multiple roles, therefore turn-taking needs to be coordinated (Johansson and Skantze, 2015). Humans signal their intents mostly through gaze, but also through pauses, prosody, and body positioning (Żarkowski, 2019). To copy this behaviour, earlier models for conversational systems relied on silence time-outs to coordinate turn-taking, however, this approach is found to be too simplistic (Skantze, 2021). Instead, mimicking human turn-taking behaviour better by using a combination of verbal and non-verbal cues leads to robots that are better perceived (Moujahid et al., 2022).

State exactly the gap that we will fill - whatever that will be

3 Data Collection

- (Laura) Talk about multi-party data collection

Data collection was performed by the team. We first collected all available recordings of "Who Wants to Be a Millionaire" with two participants, which were transcribed using the YouTube API. To annotate these transcripts, we used the unified set of annotations shown in Table 1. This list allows us to capture as much information as possible, without saturating the data.

Intent	Description
question	The system presents the question
options	The system presents the options
chit-chat	Speech not related to the quiz
offer-answer()	A player presents an answer to the other player
offer-to-answer	A player signals that they know the answer
agreement	Agreement between players about the answer
ask-agreement	A player asks the other player for confirmation on their proposed answer
accept-answer	System considers answer the final answer
final-answer()	Players give final answer
confirm-agreement	The system tries to confirm the final answer with participants
confirm-final-answer	Participants confirm their answer is final

Table 1: Intents used for Data Annotation

3.1 Cohen's Kappa Coefficient

To ensure reliability and consistency, Cohen's kappa was calculated for a sample of the completed transcripts. It measures the reliability between raters on categorical data, while accounting for agreement happening by chance (Cohen, 1960). A sample of four transcripts are re-annotated by a team member, which amounts to approximately 15% of the total transcripts.

$$KappaScore = \frac{Agree - ChanceAgree}{1 - ChanceAgree} \quad (1)$$
$$KappaScore = \underline{\underline{0.9601}}$$

A Kappa score of 0.9601 is interpreted as "almost perfect agreement" (McHugh, 2012). From this, the annotation of transcripts can be concluded as reliable.

4 Design and Implementation

Our conversational system consists of several parts, the modular architecture is shown in Figure 1. This section outlines each module and describes how our system manages the process from the users' utterances to the system's response.

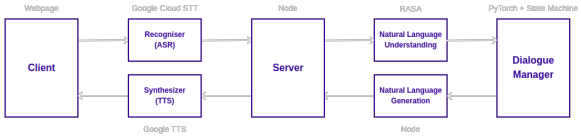


Figure 1: Architecture of the conversational system

4.1 Automatic Speech Recognition

The first step is to transform user's speech into text, in order to pass text further onto intent recognition (NLU). Transforming audio into text works through Speech-To-Text (STT) software. In recent years, STT systems have become more accurate and fast, however, none of the existing systems can yet reliably handle conversations in real time (Addlesee et al., 2020).

Our conversational system required two non-standard aspects from the ASR: (1) real-time transcription and (2) diarization. As our system conceptualised for usage on a robot, it must transcribe what the user is saying in real-time to avoid response delays. Time taken for a system to respond can only be marginally longer than the delay a human would leave before responding .

source!

Diarization is the process of determining the speaker in multi-party conversations. Therefore, to handle a "Who wants to be millionaire?" style game, our system must be able to diarise. This allows us to track the intents of each individual user and determine when users agree or disagree. We tried several STT systems including Amazon's Transcribe, IBM's Watson and locally running Pyannote. Our findings were that these are all valid options for transcription but lack real-time diarization. We settled with Google's Cloud Speech-to-Text API due to its high accuracy, real-time transcription capabilities, and customisability. As it is widely used, troubleshooting and integration resources were readily available. In addition, Google's API promised diarization capabilities, which, along with its real-time transcription capabilities seemed to fit our use case. However, in use, diarization was inaccurate, and it often grouped two separate speakers together or split sentences up seemingly at random. This became even more apparent when two users were speaking over each other, supporting the findings of (Addlesee et al., 2020). As this made Google's diarization unusable for our system, we decided to move to a set-up with two microphones (one for each user) and integrate them with the real time Google transcription.

4.2 Natural Language Understanding

Natural Language Understanding (NLU) takes in utterances of natural human language and classifies them into intents that a computer can use. RASA is an open-source framework for building conversational systems. We used its NLU tool to train on our collected data, after cleaning it . RASA NLU successfully created a model that can accurately recoginize new, unseen inputs, which are similar to the training examples. The phrases transcribed in the ASR component are passed on to the NLU model, which will label them with one of the intents from Table 1 and pass this information onto the Dialogue Management part.

word better

The model is also capable of entity extraction, which means it can recognize and label words of interest. In our case, we use entity extraction to find and label the answer to the question or a rejected answer given in an utterance .

add an example of this

4.3 Evaluation of NLU

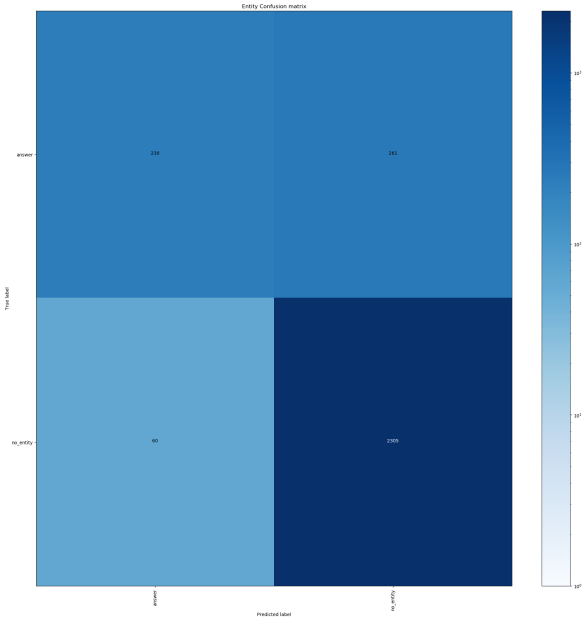
The NLU model has gone through multiple iterations, starting with very few training examples and being highly innacurate to a large amount of clean data producing a highly accurate and reliable model. Two models and their evaluations:

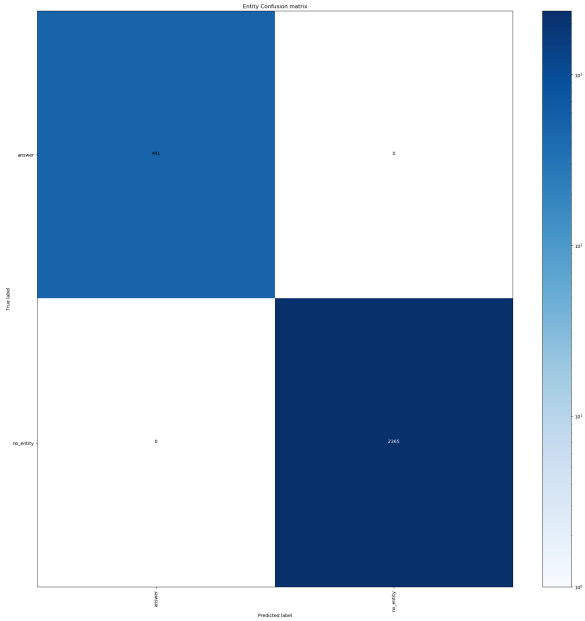
Original The original model is the first model trained on only 4 or 5 annotated transcripts, very little training data. This model was highly innacurate, constantly mislabeling intents, and extracting wrong answers or no answer at all. The Rasa training model used was not very efficient with a small amount of training data so this was expected.

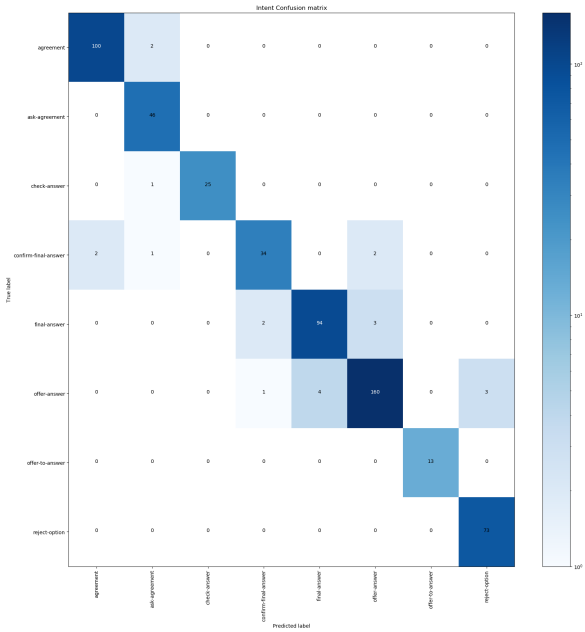
The diagram above shows the entity extraction confusion matrix, where you can see that more than half of the phrases with answers werent extracted. It did well with phrases without entities and rarely mislabed entities that werent actually answers, but its general tendency to not extract an entity was poor.

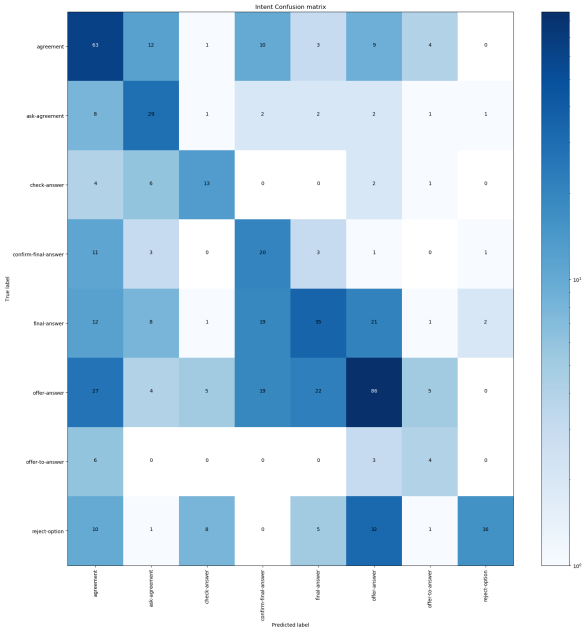
This diagram shows the confusion matrix for the intents. It shows a random scatter over the matrix meaning that the predicted labels and real labels often did not match up. The model in general was quite random and inconsistent to say the least.

Used Model The latest/best model, and the one actually used in our chatbot, was trained on around 25 transcripts with hundreds of training examples









for each intent. Furthermore, we went through all the training data and manually cleaned it of any examples that were misplaced or extracted from the transcripts wrong. The added amount of data and the cleaning proved to do wonders as the performance of the model was much more reliable and accurate.

As can be seen above, the confusion matrix is perfect as the model did not make any mistakes with entity extraction. Of course the model can't be perfect and this is only its performance on our test data, but still a crystal clear difference from the original model.

This diagram again shows the intents and how

much more accurate this model is compared to the original. It only has a few mistakes here and there, especially with the more ambiguous intents, but even us as humans confuse those intents often. Regardless, the model is leaps more accurate than its beginning.

- Short description of RASA
- Explain the aim (detecting intents)
- Refer to the intents described in 3 section 3
- Evaluation - show different versions of the model: difference in F-Score, Confusion Matrix, ...

4.4 Dialogue Management

The dialogue manager is composed of two components: a state machine and a Pytorch model running on top of it. This approach was chosen because, despite involving a conversation which requires intelligent behaviour from the dialogue manager, the game itself is defined by a rigid set of rules. There are ten questions, the conversation must see the system pass from each question to the next, and the game must end at a specified time. Running the model on top of the hard-coded logic of the state machine can ensure that this movement from state to state is observed.

reword "running the model on top of"

4.4.1 Neural Network - Cale

The neural network model was designed to manage dialogue for a three-person conversation. To do this, it has two inputs, the intent (decided by the NLU) and the user ID (user 1, user 2 or host). These inputs were both one-hot encoded, the intent being a vector of size 14 (one for each intent) and the user number being a vector of size 3. These two vectors were concatenated to produce an input vector of size 17. The network was then trained to predict the host's response for a given intent and user in a sequence. This output is a vector of size 5, one for each of the 4 system intents plus one for `no response`.

The model was trained using the dataset which we had originally labelled for NLU purposes. This provided human labelled sequences of intents and user IDs for each question. Each question would have its intents and user IDs input into the network sequentially, with the model's memory being cleared after every question. Every output from the system which is not `no response` would be input back into the network, allowing for the system to speak multiple lines of dialogue in succession. At the start of each question, the system is set up by first inputting the host having given the `question` intent, to which it will always output the `options` intent. This is required since every question in the training data is started by the host asking the question, then giving the options for the question.

The primary difficulty of training was the imbalance in the dataset. Since most of the time, the host will be only listening, `no response` is by far the most common output. This means that having the output as a single SoftMax vector of size 5 would be difficult, as the model would quickly learn to always output `no response`. Traditional undersampling and oversampling methods would also be difficult to make use of, since the `no response` output exists as part of a real conversation, which must be fed into the network sequentially. For example, uncommon system responses could not be oversampled, as they exist as part of a real question, most of which is often filled with `no response` from the host. Artificially increasing their frequency could not be done without altering the questions, which would diminish the integrity of the data.

To overcome this, a separate Sigmoid output was used for `no response` (as shown in the diagrams above). To train this model, whenever `no response` from the host was expected, the loss from the SoftMax vector was set to zero, otherwise L2 loss was used for all outputs. The system was implemented in PyTorch, with the model being trained for 30 epochs with a learning rate of 5x10-4.

In future, this architecture could be used for handling multi-person dialogues with more intents or more users by simply scaling the inputs, outputs, and hidden layers accordingly. The model could also be altered to allow for more inputs such as pauses or other non-verbal cues.

4.4.2 Dialogue Manager - Jack

The intent detected by the NLU system from a user's utterance is passed to the dialogue manager from where it is input into the PyTorch model. The model then decides on an appropriate action. The state machine checks if the output from the PyTorch model represents a sensible action. In case there are logical errors, the output will be overridden. For example, if the NLU detects that the user

intends to confirm final answer, but no answer has yet been given, the PyTorch model may nonetheless decide on `accept-answer`. As no answer has been given set, the state machine will overwrite this output.

Another reason for using the state machine was the limited transcript data. The PyTorch model learned off of the transcript data. There are user intents found throughout the transcripts, followed by a certain action (or no action). Therefore, the PyTorch model depended on a sufficient amount of transcript data to function adequately. Using a state machine allowed us to program explicitly how the chatbot is meant to respond to user intents, thus eliminating the need for so more transcript data.

insert Jack's figure

The state machine's configuration is defined inside a JSON file . When the action decided by the chatbot is overridden by the state machine, it is done according to this configuration. The action chosen by the state machine is a string of JavaScript code, and is found in the configuration object `flow` at `flow[state][intent]`, where `state` is the current state of the machine and `intent` is the name of the most recent user intent.

Throughout the course of a game, the dialogue manager stores relevant values, such as answers offered by the user, and these values also influence the behaviour of the state machine. An example of this can be found inside the 'seek-confirmation' state when the host asks the users if they would like to lock in an answer suggested by them. If a user responds by rejecting the answer they had just given, then the host will check to see if the answer they are rejecting is the same as the answer that they had just offered. If it is, then the host will assume that they are not wishing to lock in that answer. However, if the answer being rejected by the user is another one, then the host will assume that they do indeed wish to proceed with their offered answer.

A more secure approach to this situation would involve taking a record of all the answers ruled out by the users. The host would then lock in a final answer after a rejected answer if and only if the other two possible answers had also been previously rejected. We would take in a future implementation as it would reduce the risk of the host accepting answers while the users are still deciding.

4.4.3 Xander

Recurrent Neural Networks and Long-Short Term Memory are the two types of Neural Network that have been used throughout the project due the fact that they use context when calculating the outcomes of the Neural Network. A Neural Network has multiple hidden layers each has its own weights and bias. This causes each layer has this they act independently, which is not helpful when trying to identify a pattern between the successive inputs. RNN will have the same weights and bias for the hidden layers, but will supply the a state to each neuron. This state is then used to link the next input to form the next state. However, the issue with this is when you have very long pieces of data, especially when referring to dialogue. If there is a long sentence the RNN must do a lot of recursions in order to process every single word or rather state. Then there is the problem of vanishing gradient although this we are less observable throughout our project as the scope of our project was not that large. This is a known and is Long-Short Term Memory, a type of Recurrent Neural Network has been developed. Long-Short Term Memory, even as shown in our project has faster processing speeds. This is because it has the means of forgetting by identifying if a piece of input data changes, instead of doing what a normal RRN does and constantly goes back regardless of what the context of the data being given is. When deciding how to do the dialogue management when building the chatbot when were considering using the RASA Dialogue Management but had to move away from the toolkit. The reason behind this is although the sophisticated system in place with pre-existing system, using the inbuilt stories and intents within the system, however it was not built in with multiple users in mind. To achieve multi-person dialogue with more than just the chatbot and a single individual would not be able to use Rasa. Rasa was not built with multiple people in mind. This meant there was only the intent and response when developing the dialogue manager. For it to function it would have to be heavily worked on just to allow for the multiple people to be used. This ignores any issues with developing the dialogue manger that would take place and would be compounded. To use RASA it would have to train on each epoch twice one for each user to prevent it from falling into a pattern based on how second users may act within the

training data. Every intent would need to be created twice, once for each user. Alongside the problems once where the Diarization trying to identify the user there would be issues with the trying to work out which intent belong to which user and since they have both had the same set of training data this could cause a great deal of confusion within the system making it next to impossible to separate the different users. This is why we decided to go with building our own system as this way we could alter the fundamental tools that RASA gives us, obviously the main tools would not have been as robust as RASA as that has an entire development team behind it but it is simple unable to achieve what we required from our task at hand.

4.5 Natural Language Generation

To make the system feel more unique and less robotic, we made use of Natural Language Generation (NLG) for the phrases that the "host" says to the participants. We used OpenAI's API, specifically "gpt-3.5-turbo", the same version that is used in ChatGPT. This allowed us to prompt for different outputs for the system that convey the same information. For example, when receiving a correct answer, "Yes, that's it! Well done!" and "You got it! Great job!" are both possible outputs. There are 50 different options for each response the "host" can say.

5 Evaluation

from coursework spec: evaluation of the system and presentation of the results

5.1 Methodology

In this study, we performed extrinsic and intrinsic evaluations. The extrinsic evaluation focused on both subjective and objective measures of the system's performance. The subjective measures included the user's enjoyment and perception of the system's natural behaviour, while the objective measures included the number of turns taken and the agreement rate. Additionally, the correlation between correct answers and enjoyment was also examined. Overall, the evaluation aimed to assess the effectiveness of the system in engaging users and providing accurate responses.

The evaluation focused on intrinsic measures of individual components in a multi-party conversational system. The components were assessed separately, with ASR being evaluated using the word error rate, NLU using precision, recall, accuracy, and F1 score, DM IDK YET, and NLG using n-gram-based overlap with BLEU. The aim was to assess the performance of each component and identify areas of improvement.

Additionally, the evaluation also aimed to test the hypothesis that using verbal cues instead of silence cues in a multi-party conversational system increases user interaction and satisfaction with the system. This hypothesis needed to be statistically proved or disproved through significance testing.

5.2 Experiment Layout

The experiment followed a between-subjects design. Every participant was required to read and sign a consent form before they can play the quiz. This was an in-person experiment, with the quiz running on a laptop, where participants can see the questions and the options. Members of the experiment were required to play the quiz at least once. However, they were encouraged to play as many times as they can. After they no longer wished to play, they were asked to complete a questionnaire about their experience. The questionnaire queried them on their experience using a five-point Likert scale.

5.3 Results

- ASR results
- NLU result
- DM result NLG result
- questionnaire result

6 Conclusion

6.1 Ethical Reflection

7 Future Work

from coursework spec: suggestions for future work

To further improve on NLG within this system, some content moderation could be performed on the generations, to ensure that there are no inappropriate outputs, this can be done entirely within OpenAI's API. Also, the system could be updated to make use of GPT-4, which at this current time is not publicly available.

Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

References

[Addlesee et al.2020] Angus Addlesee, Yanchao Yu, and Arash Eshghi. 2020. A comprehensive evaluation of incremental speech recognition and diarization for conversational AI. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3492–3503. International Committee on Computational Linguistics.

[Addlesee et al.2023] Angus Addlesee, Weronika Sieinska, Nancie Gunson, Daniel Hernández García, Christian Dondrup, and Oliver Lemon. 2023.

[Cohen1960] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, Apr.

[Cooper et al.2020] Sara Cooper, Alessandro Di Fava, Carlos Vivas, Luca Marchionni, and Francesco Ferro. 2020. ARI: the social assistive robot and companion. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 745–751. ISSN: 1944-9437.

[Gunson et al.2022] Nancie Gunson, Daniel Hernandez Garcia, Weronika Sieińska, Angus Addlesee, Christian Dondrup, Oliver Lemon, Jose L. Part, and Yanchao Yu. 2022. A visually-aware conversational robot receptionist. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 645–648. Association for Computational Linguistics.

[Johansson and Skantze2015] Martin Johansson and Gabriel Skantze. 2015. Opportunities and obligations to take turns in collaborative multi-party human-robot interaction. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 305–314, Prague, Czech Republic, Sep. Association for Computational Linguistics.

[McHugh2012] Mary L. McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia Medica*, 22(3):276–282, Oct.

[Moujahid et al.2022] Meriam Moujahid, Helen Hastie, and Oliver Lemon. 2022. Multi-party interaction with a robot receptionist. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 927–931.

[Robotics2023] Pal Robotics. 2023. Ari - the social and collaborative robot. Accessed on: 2023-02-08.

[Skantze2021] Gabriel Skantze. 2021. Turn-taking in conversational systems and human-robot interaction: A review. *Computer Speech and Language*, 67:101178.

[WHO2023] WHO. 2023. Global health workforce statistics. Accessed on: 2023-02-07.

[Żarkowski2019] Mateusz Żarkowski. 2019. Multi-party turn-taking in repeated human–robot interactions: An interdisciplinary evaluation. *International Journal of Social Robotics*, 11(5):693–707, Dec.

A Supplemental Material, Appendix