Detecting agreement in a Multi-party Quiz

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

Abstract

Today, conversational systems are expected to handle conversations in multi-party settings, especially as Socially Assistive Robots (SARs). However, practical usability remains difficult as there are additional challenges to overcome, such as speaker and addressee recognition and turn-taking. In this paper, we present our work on a multi-party conversational system, which invites two users to play a trivia quiz game. The system detects users' agreement or disagreement on a final answer and responds accordingly. Our evaluation includes both performance and user assessment results, with a focus on detecting user agreement. We have configured the system to be used on the ARI robot deployed in a hospital waiting area as part of the SPRING project.

1 Introduction

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 being experienced worldwide, with 10 million more health workers needed 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 longer or reducing caregiver burden (Cooper et al., 2020).

Healthcare SAR scenarios frequently require these robots to handle multi-party interactions as it is likely in these situations that more than one person will interact with the system (i.e., a hospital waiting area). Therefore, we propose a conversational system that plays a game of "Who wants to be millionaire?" with two users. Our system impersonates the host while participants can collaborate on general knowledge trivia questions.

The conversational system is 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 questions. The system's aim is to guide users to collaborate and ultimately agree on an answer to the question. The code for this project is open-source and available on GitHub[1].

Our system is developed to be fitted onto an ARI robot in the course of the SPRING-ARI project. This project encompasses several European universities collaborating on evolving socially assistive robots. The work is tested on an ARI robot, which is deployed in a hospital waiting area in France.

2 Background

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).

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,

[1] https://github.com/JsnSwny/tarrant

2023b). 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 conversational systems

In general, most conversational systems are only able to handle dyadic (two-party) conversations (Mahajan et al., 2022; Moujahid et al., 2022). The endeavour to create conversational systems becomes considerably more difficult when dealing with multi-party interactions (Addlesee et al., 2023). Compared to handling dyadic interactions, handling multi-party conversations includes more complex challenges, such as coordination of turn-taking and Speaker Recognition, Addressee Recognition, Response Selection (summarised by the phrase "Who says what to whom") (Addlesee et al., 2023; Johansson and Skantze, 2015).

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).

Most existing multi-party conversational research has focussed on the sub-tasks of "Who says what to whom". Mahajan et al. identified 15 papers in the last decade focussing on one or more of these sub-tasks within an english-speaking context (Mahajan et al., 2022). Centralised evaluation criteria are relatively new, and most existing systems are not consistently evaluated. One previous study, conducted by Skantze et al, required users to collaborate with a robot in a multi-party setting. The users' task was to play a card game with the Furhat robot where they had to sort the cards in a specific order. However, the research focussed on turn-taking cues rather than the system's performance in terms of aiding users to collaborate or detecting agreement between users (Skantze et al., 2015; Robotics, 2023a).

3 Data Collection

An important point to consider in multi-party training data is that most are human-human. This can pose a problem as systems trained on human-human conversations may lack the incentive to help users reach their goals (Addlesee et al., 2023). However, in our scenario, the system has the clear goal of guiding the participants to finding an answer to a question. To our knowledge, there are no multi-party corpora available for this use case, therefore data collection was performed by our 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 set of annotations shown in Table 1. This list allows us to capture as much information as possible, without saturating the data.

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 were re-annotated by a team member, which amounts to approximately 15% of the total transcripts.

$$\begin{aligned} KappaScore &= \frac{Agree - ChanceAgree}{1 - ChanceAgree} \\ KappaScore &= \underline{\underline{0.9601}} \end{aligned} \tag{1}$$

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 unit and describes how our system manages the process from the users' utterances to selecting its response.

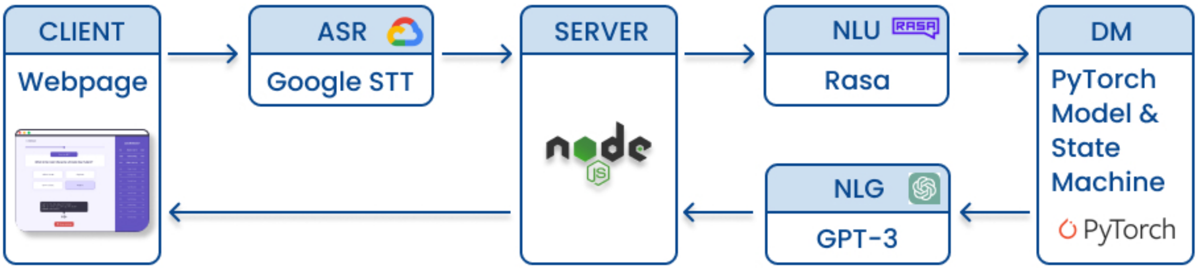


Figure 1: Architecture of the conversational system

Intent	Description
Host (System) Intents	
question	The system presents the question
options	The system presents the options
confirm-agreement	The system tries to confirm the final answer with participants
accept-answer	System considers answer the final answer
User Intents	
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
final-answer()	Players give final answer
confirm-final-answer	Participants confirm their answer is final

Table 1: Intents used for Data Annotation

4.1 Automatic Speech Recognition

The first step is to transform user's speech into text, which can be passed onto Natural Language Understanding (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 features from the ASR: (1) real-time transcription and (2) diarization. Given that it was designed for usage on a robot, it must transcribe what the user is saying in real-time to avoid response delays and ensure natural sounding conversation. Therefore, the time taken for a system to respond can only be marginally longer than the delay a human would leave before responding (Miller, 1968).

The second feature, diarization, is the process of determining the speaker in multi-party conversations. To handle a "Who wants to be millionaire?" style game, our system must be able to diarise to track the intents of each individual user to determine when users agree on a final answer. We tried several STT systems including Amazon's Transcribe, IBM's Watson and locally running Pyannote. Our findings were that these are all suitable for transcription but lack real-time diarization. We settled with Google's Cloud Speech-to-Text API due to its high accuracy 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 statement that ASR systems are "not yet adequate to reliably handle natural spontaneous conversations in real-time" (Addlesee et al., 2020).

This made Google's diarization unusable for our use case. We therefore moved to a set-up with two microphones (one for each user) and integrated them with the real time Google transcription. By removing diarization we were able to use the most up-to-date Google model `latest_long`, which is trained on long-form conversation. This allowed us to not only avoid the issue of diarization, but also to improve the quality of the transcriptions, which had a cascading effect on the rest of the system, leading to better intent recognition, entity extraction, etc.

A drawback of this set-up is that the microphones may pick up both users' voices. This could be mitigated through moving the microphones further apart, or calibrating them. The issue became apparent during our evaluation, results of which are discussed in Section 5.

The real-time transcription of the users' utterances are then passed onto NLU.

4.2 Natural Language Understanding

Natural Language Understanding takes in utterances of natural human language and classifies them into intents. RASA is an open-source framework for building conversational systems, which offers NLU tools. We used its NLU tool to train on our pre-processed data. RASA NLU successfully created a model that can accurately label new, unseen inputs from the "Who wants to be millionaire" game domain with an intent from the list given in Table 1.

The model is also capable of entity extraction, which means it can identify words of interest. In our case, we extract a user's answer to a question or a user's rejected answer given from an utterance. For example, a user's utterance could be "I'm thinking Teaspoon", which would be classified by the NLU in the following way:

```
- intent: offer-answer
  entities:
    - answer: "Teaspoon"
```

4.2.1 Evaluation of NLU

An initial model trained on only four annotated transcripts was innacurate, mislabelled intents, and extracted wrong answers or no answer at all. The accuracy for this model was 47% with a macro-averaged F1-Score of 0.43.

Using a larger volume of training data significantly improved NLU performance. The model used in our system was trained on 25 transcripts with 566 training examples for each intent. Additional improvement could be obtained by manually cleaning the training data of any misplaced or wrongly extracted examples. As can be seen in Figure 2, the improved model's confusion matrix is diagonal, meaning there have been no misclassifications. There was a small amount of misclassifications in intent recognition, this can be seen in Appendix A shows the intent confusion matrix. This model has an accuracy of 96.3% with a macro-averaged F1-Score of 0.96.

The intents identified by the NLU are then passed onto the Dialogue Manager (DM) unit.

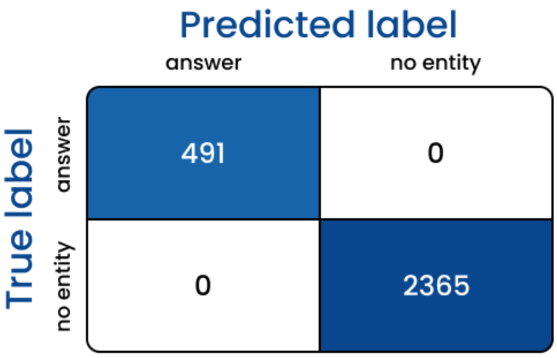


Figure 2: Confusion Matrix of NLU answer extraction

4.3 Dialogue Management

The dialogue manager is responsible for selecting an adequate response at any given moment. It is also tasked with asking questions, which it obtains from the OpenTrivia Database, a user-contributed trivia question database under the Creative Commons License (LLC, 2023). The dialogue manager is composed of two components: a state machine and a Pytorch model. This approach was chosen because, despite involving a conversation which requires intelligent behaviour from the dialogue manager, the game itself is defined by a set of rigid rules:

- The users are always asked ten questions.
- The system must drive the conversation by passing from one question to the next.
- The game must end at a specified time.

Occasionally overwriting the model's output with the hard-coded logic of the state machine ensures these rules are followed and that the movement from state to state is observed.

4.3.1 Neural Network

The neural network model is designed to manage dialogues of three-person conversations. To do this, it has two inputs, the intent from 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` (see Table 1).

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 fig. 3. 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.

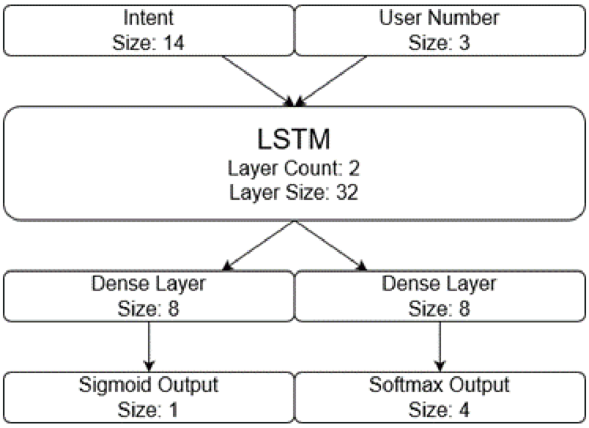


Figure 3: Architecture of PyTorch Model

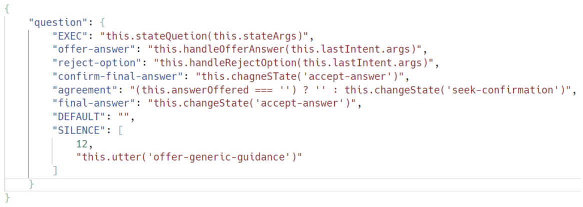
In future, this architecture could be used for handling multi-person dialogues with more intents or more users by simply scaling the inputs, outputs, and hidden layers accordingly. The model could also be altered to allow for more inputs such as pauses or other non-verbal cues.

4.3.2 State Machine

An intent received from the NLU unit is first input into the PyTorch model, which outputs an action. Afterwards, the state machine checks if the output of 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 their final answer, but no answer has yet been given, the PyTorch model may nonetheless decide on `accept-answer`. As no answer has been given yet, the state machine will overwrite this output.

Another reason for using a state machine was the limited amount of transcription data. The PyTorch model learned off of the transcript data, therefore it depended on a sufficient amount of transcript data to function adequately. The state machine extends the set of actions by ten additional intends, which are listed in Table 2. This allowed us to program explicitly how the chatbot responds to user intents, thus eliminating the need for more transcript data.

The state machine's configuration is defined inside a JSON file, as shown in Figure 4. The figure shows the behaviour of the state machine when the state is `question`. An example of previously recorded values influencing the flow can be seen inside `agreement`. When the action de-



```
{
    "question": {
        "EXEC": "this.stateQuetion(this.stateArgs)",
        "offer-answer": "this.handleOfferAnswer(this.lastIntent.args)",
        "reject-option": "this.handleRejectOption(this.lastIntent.args)",
        "confirm-final-answer": "this.chagneSTate('accept-answer')",
        "agreement": "(this.answerOffered === '') ? '' : this.changeState('seek-confirmation')",
        "final-answer": "this.changeState('accept-answer')",
        "DEFAULT": "",
        "SILENCE": [
            12,
            "this.utter('offer-generic-guidance')"
        ]
    }
}
```

Figure 4: State Machine behaviour

cided 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. These values influence the behaviour of the state machine. An example of this can be observed inside the `seek-confirmation` state when the host asks if the user would like to lock in an answer suggested by them. If a user declines, the host checks if the rejected answer is the same as the one just offered by the user. If it matches, then the host assumes that the user does not want to lock in that answer. However, if the answer 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.

Once the DM has identified an appropriate response, this intent is passed onto the Natural Language Generation (NLG) unit.

4.4 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.

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, as GPT-4 works more effectively to avoid inappropriate content and has

Intent	Description
acknowledge-reject-option	The system acknowledges that a player has dismissed an option and may ask them for a reason.
end-of-game	The system announces the end of the game and informs the players of how they did.
offer-generic-guidance	The system offers advice or encouragement to the players.
question-brief	The system restates the current question number and question prize.
repeat-answer	The system repeats a player's answer back to them.
return-to-question	The system returns to the question state, becoming less insistent on locking in answers suggested.
say-correct	The system states that the players's final answer was correct.
say-incorrect	The system states that the players's final answer was incorrect.
seek-confirmation	The system asks the players if they are wishing to lock in a suggested answer as final.
seek-direct-answer	The system asks the players to state their answer. This is done by the system when it suspects that it may have missed an answer offered.

Table 2: Additional State Machine Intents

greater problem solving abilities, however, at this current time it is not publicly available (OpenAI, 2023b; OpenAI, 2023a).

5 Evaluation

5.1 Experiment Layout

We evaluated 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 focussed on the agreement rate. The agreement rate indicates the frequency of the system correctly detecting users' intent to submit a final answer.

The evaluation took place in two iterations in a between-subjects design, both iterations with three pairs of participants. Each participant sat in front of a laptop, where they could see the webpage with the question and answer options on the screen (see Appendix B). Each pair of participants played one round of the quiz (10 questions). Afterwards, they were asked to complete a questionnaire about their experience. They rated the system in the following aspects using a five-point Likert scale:

- Ease of understanding the rules of the game
- Ease of navigating the UI
- Enjoyment
- Difficulty of the questions
- Naturalness of the system
- Overall satisfaction

In the first iteration, we followed a strategy of selecting ten questions at random from a set of 34 and included all possible answers as entities to make sure the system was able to recognise them. For example, the answer option "Bushwhackers" had the following variations:

```
"The Bushwhackers",
"the bushwhackers",
"bushwhackers",
"bush whackers",
"bush wackers",
"bushwackers",
"whackers"
```

However, this approach restricted generalization beyond the specific set of questions and answers. We also came across additional issues, such as the microphones picking up both users' voices and questions being repeated. Therefore, we improved the system for the second iteration with the following features:

1. The set of questions was expanded to 3490.
2. Fuzzy String Matching was implemented.
3. Microphones were moved further apart.

By incorporating a fuzzy matching algorithm (discussed in Section 5.2.2), entity extraction was improved across all questions in our database. This technique enabled us to implement a more robust system that could accommodate to a wide range of questions and answers. As discussed in Section 4.1, too little of a distance between microphones meant that both users' voices would get picked up by one. This led to the system detecting agreement as both users' seemingly agreed on the same answer, while it was in fact a duplicated utterance. In addition to adding questions, we also implemented logic to make sure the same question would not show up twice in one round of the game, which was an issue in the first iteration.

5.2 Results

5.2.1 User Satisfaction

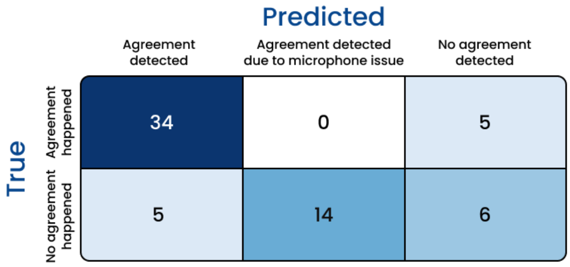
The enhancements between the two iterations yielded improvements, with 83% of participants expressing satisfaction with the overall system in the first evaluation, and 100% in the subsequent evaluation. Regarding the system's behaviour, 50% of participants in both evaluations indicated that the system appeared natural. Furthermore, 83% of participants reported enjoying the experience in both evaluations. This represents a notable advancement in the performance of system 2, as participants reported encountering more challenging questions during the second evaluation, resulting in a lower rate of correct answers. Prior to this improvement, the difficulty of the questions and the lower rate of correct answers had resulted in reduced enjoyment for the participants.

5.2.2 Fuzzy String Matching

Fuzzy string matching is a computational technique to approximate string matches. It does so by tokenizing both strings and comparing the similarity of common tokens of both strings. In our system, we employed fuzzy string matching to identify potential answers based on the participants' speech. The selection of an optimal similarity threshold was critical for performance. We performed a within-subject analysis, comparing a 0.5 and 0.7 threshold. The 0.5 threshold yielded the best performance, as all users' answers were matched with an answer option. With a 0.7 threshold, 33% of users' answers were not matched due to imperfect transcriptions. A 0.5 threshold is more effective mitigating transcription errors compared to a 0.7 threshold.

5.3 Agreement Detection

This confusion matrix shows the predictions made by the system for 6 evaluations. The True Positive value is when the system accurately identifies agreement, the False Positive value is when it identifies agreement that hasn't happened (with additional column to separate the microphone issue), False Negative is when the system misses an agreement and the True Negative value is when users do not wish to lock in an answer and the system recognises that.



True \ Predicted	Agreement detected	Agreement detected due to microphone issue	No agreement detected
Agreement happened	34	0	5
No agreement happened	5	14	6

Figure 5: Agreement Detection Confusion Matrix

By adding the "Agreement detected due to microphone issue" column, issues relating to multiple microphones lead to a more accurate agreement and error rate.

$$\begin{aligned} Agreement\ Rate &= \frac{34+6}{50} = 80\% \\ Error\ Rate &= \frac{5+5}{50} = 20\% \end{aligned} \tag{2}$$

Without the microphone issue, our system would have an agreement rate of 80% with an error rate of 20%.

From the matrix, it can be seen that the main issue with the system is false positive predictions. The reason for this is the microphones of both laptops picking up the same sentence and the system classing this as agreement when only one user had spoken. The false negatives tended to occur when the answer names are very obscure and the system struggled to pick them up.

6 Conclusion and Future Work

In this report, we have presented a conversational system to play a game of "Who wants to be millionaire?" in a multi-party setting. The system is able to detect users agreeing on a final answer, or will continue asking for an answer to encourage users to reach a conclusion. The question and answer options are displayed on a website, which is intended to mimic the ARI robot's screen. To evaluate the system, we conducted an experiment assessing users' perception of the system, as well as its performance. If the microphone issue can be avoided, the system reaches an agreement rate of 80%.

There are several small-scale problems, which we only detected in the second iteration of our evaluation. Users were reading out all possible answers, which we did not account for. The system recognised this utterance as an answer entity, and assumed the users were offering or finalising an answer. The system's output was also sometimes overlooked, which made correct state tracking difficult. As an evaluation participant stated, text-to-speech output would make it feel more like a conversational system and remove the need for users to keep reading the host's responses.

The main challenge for our system will be to transition from the set-up with two microphones to using speech diarization in real-time. This transition will make the microphone issue of picking up both users' voices redundant. As the ARI robot is fitted with directional microphones, deploying our system on the robot can pose a solution to this problem. Furthermore, the robot's hardware could be used to detect visual turn-taking cues. This would allow moving away from simple silence time-out cues.

To make our system better comparable to other multi-party conversational systems, it should be evaluted against the error reporting taxonomy by Higashinaka et al (Higashinaka et al., 2021), extended for multi-party dialogue systems by Mahajan et al (Mahajan et al., 2022).

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. Data collection for multi-party task-based dialogue in social robotics. *IWSDS*, Feb.

[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.

[Higashinaka et al.2021] Ryuichiro Higashinaka, Masahiro Araki, Hiroshi Tsukahara, and Masahiro Mizukami. 2021. Integrated taxonomy of errors in chat-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 89–98, Singapore and Online, Jul. 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.

[LLC2023] Pixeltail Games LLC. 2023. Open trivia db.

[Mahajan et al.2022] Khyati Mahajan, Sashank Santhanam, and Samira Shaikh. 2022. Towards evaluation of multi-party dialogue systems. In *Proceedings of the 15th International Conference on Natural Language Generation*, pages 278–287, Waterville, Maine, USA and virtual meeting, Jul. Association for Computational Linguistics.

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

[Miller1968] Robert B. Miller. 1968. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS 1968 (Fall, part I)*, page 267, San Francisco, California. ACM Press.

[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.

[OpenAI2023a] OpenAI. 2023a. Gpt-4 technical report. *OpenAI*, Mar.

[OpenAI2023b] OpenAI. 2023b. Openai gpt-4 introduction. Accessed on: 2023-03-30.

[Robotics2023a] Furhat Robotics. 2023a. Furhat - the world's most advanced social robot. Accessed on: 2023-04-05.

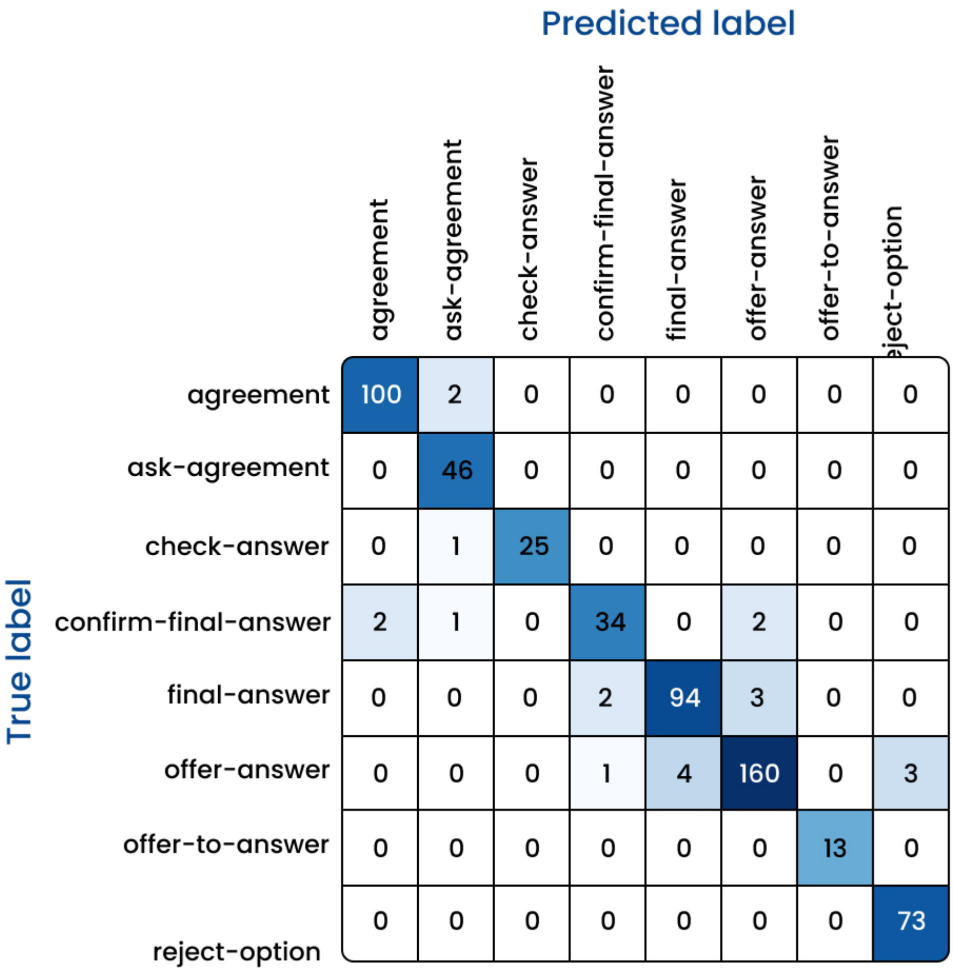
[Robotics2023b] Pal Robotics. 2023b. Ari - the social and collaborative robot. Accessed on: 2023-02-08.

[Skantze et al.2015] Gabriel Skantze, Martin Johansson, and Jonas Beskow. 2015. Exploring turn-taking cues in multi-party human-robot discussions about objects. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 67–74, Seattle Washington USA, Nov. ACM.

[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 NLU Intent Recognition Confusion Matrix



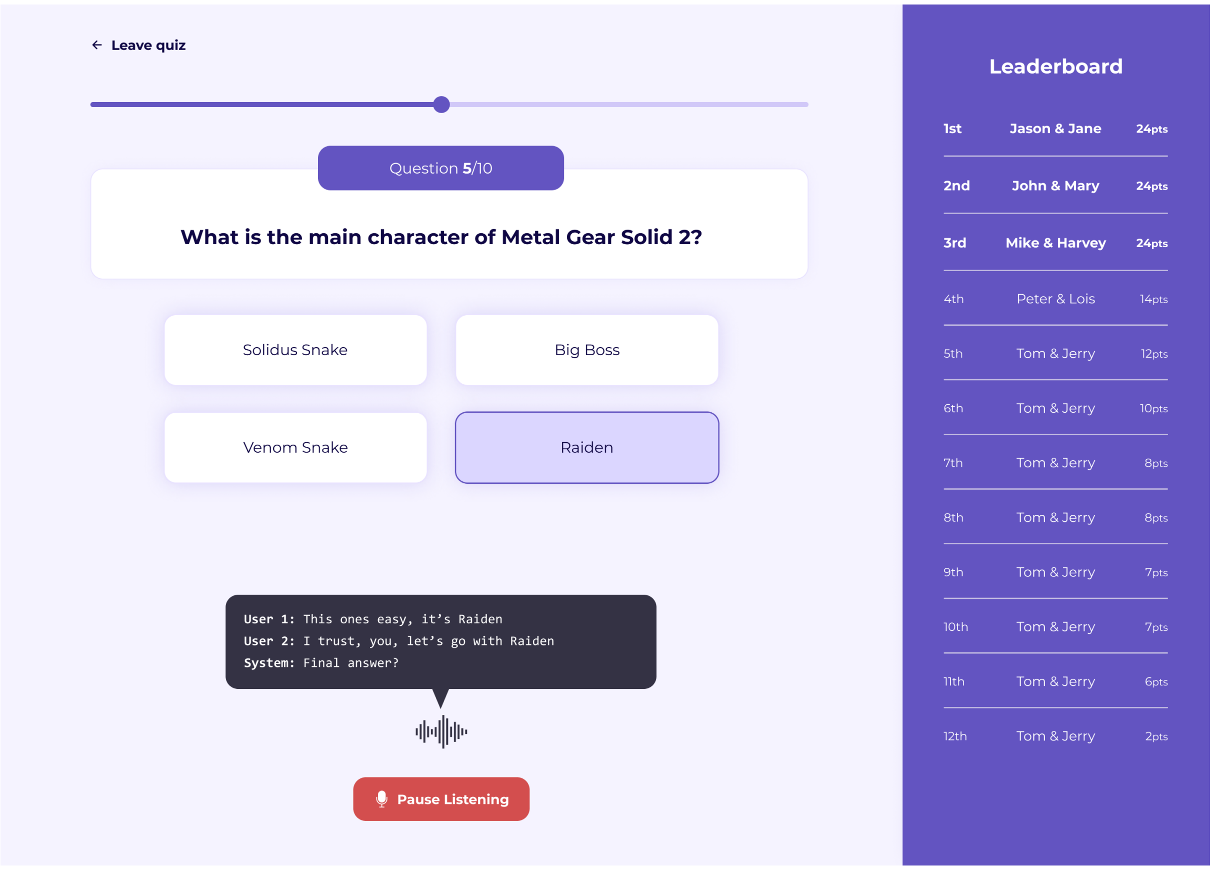
Predicted label

True label

	agreement	ask-agreement	check-answer	confirm-final-answer	final-answer	offer-answer	offer-to-answer	eject-option
agreement	100	2	0	0	0	0	0	0
ask-agreement	0	46	0	0	0	0	0	0
check-answer	0	1	25	0	0	0	0	0
confirm-final-answer	2	1	0	34	0	2	0	0
final-answer	0	0	0	2	94	3	0	0
offer-answer	0	0	0	1	4	160	0	3
offer-to-answer	0	0	0	0	0	0	13	0
reject-option	0	0	0	0	0	0	0	73

Figure 6: Confusion Matrix of NLU intent recognition

B Screenshot of System’s Webpage



What is the main character of Metal Gear Solid 2?

```
User 1: This ones easy, it’s Raiden
User 2: I trust, you, let’s go with Raiden
System: Final answer?
```

Leaderboard

1st	Jason & Jane	24pts
2nd	John & Mary	24pts
3rd	Mike & Harvey	24pts
4th	Peter & Lois	14pts
5th	Tom & Jerry	12pts
6th	Tom & Jerry	10pts
7th	Tom & Jerry	8pts
8th	Tom & Jerry	8pts
9th	Tom & Jerry	7pts
10th	Tom & Jerry	7pts
11th	Tom & Jerry	6pts
12th	Tom & Jerry	2pts

Figure 7: Screenshot of System’s Webpage