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 2021: number 5 !

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.

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.

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

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

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.

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) to perform intent recognition. 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 our system 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 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 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 user's voices. This could be mitigated through moving the microphones further apart, or calibrating them. More on this issue is 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 millionnaire" 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:

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

4.2.1 Evaluation of NLU

An initial model trained on only four annotated transcripts, very little training data was highly 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 manu-

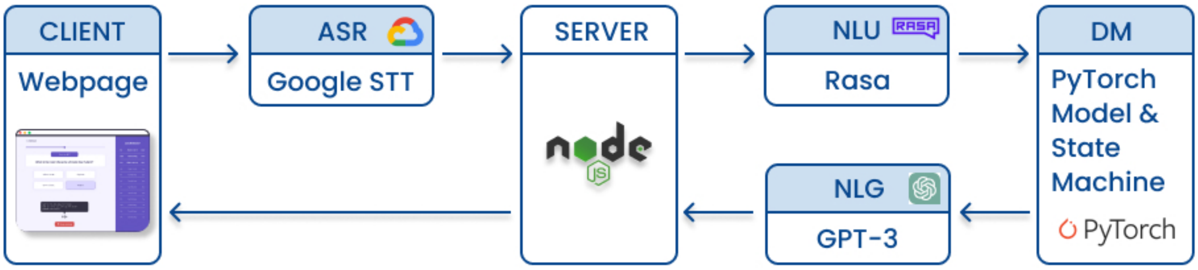

Figure 1: Architecture of the conversational system

ally 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 perfect. Figure 3 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 DM unit.

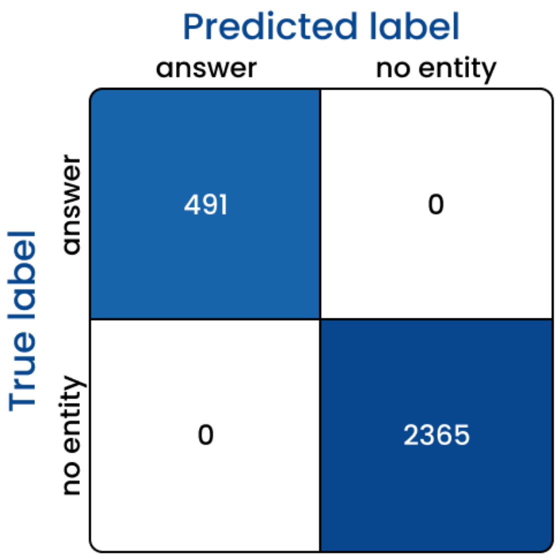

Figure 2: Confusion Matrix of NLU answer extraction

4.3 Dialogue Management

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.

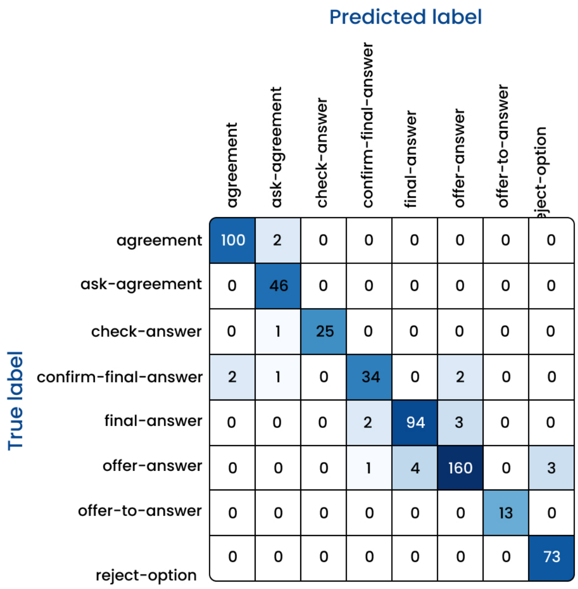

Figure 3: Confusion Matrix of NLU intent recognition

- The game must end at a specified time.

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

4.3.1 Neural Network - Cale

The neural network model is needed to be designed to manage dialogues of three-person conversations. To do this, it has two inputs, the intent (passed on 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

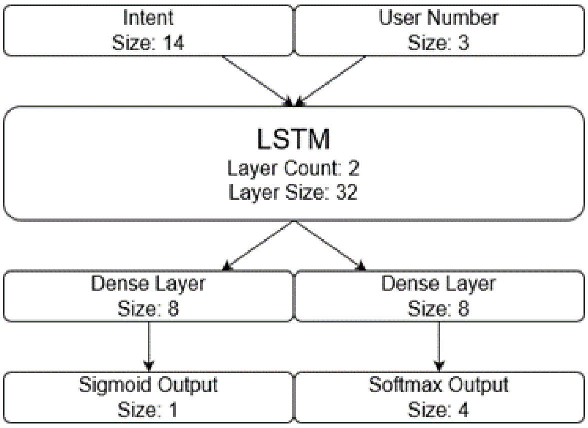

Figure 4: Architecture of PyTorch Model

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 fig. 4. 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.

Neural Network Architectures

- difference between LSTM & RNN
- why not RASA, but own DM

4.3.2 Dialogue Manager - Jack

An intent received from the NLU unit is first input into the PyTorch model. The model then decides on an appropriate 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 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. Using a state machine allowed us to program explicitly how the chatbot is meant to respond to user intents, thus eliminating the need for more transcript data.

```
{
    "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 5: State Machine behaviour

The state machine's configuration is defined inside a JSON file, as shown in Figure 5. 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 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. 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.

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

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

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.

[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.

[OpenAI2023b] OpenAI. 2023b. Gpt-4 technical report. *OpenAI*, Mar.

[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