Bart the Bartender

1st Danyal Mirza
Department of Computer Science
Uppsala University (UU)
Uppsala, Sweden
danyal.mirza.7311@student.uu.se

2nd Linnéa Soto Carlsson
Department of Computer Science
Uppsala University (UU)
Uppsala, Sweden
Linnea.sotocarlsson.7666@student.uu.se

3rd Walter Levens
Department of Computer Science
Uppsala University (UU)
Uppsala, Sweden
walter.levens.8542@student.uu.se

4th Adrià Irigaray Pola
Department of Computer Science
Uppsala University (UU)
Uppsala, Sweden
email address or ORCID

***Abstract*—Linnea + Danyal**
***Index Terms*—component, formatting, style, styling, insert.**

I. Introduction

Linnea + Danyal
The objective of the project was to design and develop "Bart", a bartender capable of reading the user's emotional state and providing drink recommendations aligning with the detected mood while engaging in a pleasant conversation. In addition, the system takes into account the individual drink preferences of the users to ensure a satisfying experience. These specific objectives were chosen in the beginning of the project, as the group members believed they were essential for creating a meaningful project connected to the course work. The work for this project was divided into two primary subsystems: the Perception Subsystem and the Interaction Subsystem.

- **Linnéa** and **Danyal** were responsible for developing the perception subsystem. They built a video server together with a client to connect to the interaction system. The video server is capable of detecting faces and recognizing emotions, with the help of a Random Forest (RF) machine learning model to classify mood based on facial expressions.
- **Adrià** and **Walter** focused on the interaction subsystem. Their work involved setting up the Furhat robot, integrating a Large Language Model (LLM) into Furhat, and designing the system to translate detected emotions and user preferences into prompts for generating personalized drink recommendations.

Teamwork within the group was collaborative and efficient, where meaningful discussions helped solving problems encountered throughout the project. The writing of the report was divided equally among the members, and each person primarily contributed to their specific area of work.

II. Methodology

Linnea + Danyal
This section describes the methodology and system architecture used in the project.

A. Overall system design

Linnea + Danyal
The scenario chosen for this project is as said in previous sections a bartender. The primary task of Bart, our bartender, is to analyze the customer's emotional state and offer drink recommendations based on their mood. Additionally, the system personalizes the recommendations by taking into account the customer's individual drink preferences from engaging in a pleasant conversation with the customer. This ensures a more enjoyable experience.
The system consists of multiple parts combined into one functional, smooth system accomplishing the below stated high level goals.

HIGH-LEVEL GOALS:

1) **Emotion Recognition:** Detect the client's emotional state accurately based on facial expressions.
2) **Personalized Drink Recommendations:** Suggest a drink that fits both the emotional state and the individual preferences of the client.
3) **Engaging Conversation:** Maintain a pleasant and natural conversation with the client while offering drink recommendations.
4) **Real-Time Interaction:** Provide real-time responses to emotions and preferences without significant delays, ensuring smooth and interactive engagement.

To achieve these goals, the system consists of two subsystems: the **perception subsystem** and the **interaction subsystem**. The perception sub-system consists of a video socket server recognizing faces combined with a machine

learning model classifying emotions of recognized faces. The interaction sub-system consists of a client establishing contact with the video server, the client is integrated with Furhat, a social robot platform which is designed to enable human-like interaction through multimodal communication. Furhat have advanced speech synthesis, facial animations and gesture recognition. Furhat provides a remote API which makes it possible for us to control, design and interact with the robot over a network. Together with the client and Furhat, a Large Language Model (LLM), Gemini is integrated to the interaction subsystem to enable a response to the user speech input.

B. User perception sub-system

Linnea + Danyal

1) Server design: To allow emotion detection, a video server had to be created that captured video of user interacting with Furhat. The video server begins with establishing a socket connection with the Furhat client to allow communication between the two subsystems. The server then begins capturing video and for each current frame uses py-feat to detect faces and AU-values are then sent to our trained machine learning model to predict emotion. A message containing the following information is then sent to the client via the socket connections:

- Number of faces detected
- Values representing location of the face
- Predicted emotion

2) Machine Learning Design: The random forest machine learning model for emotion detection was developed through several steps with the goal to optimize performance. Before deciding on random forests as the primary model, several other machine learning approaches were explored, including Support Vector Machines (SVM), boosting, and neural networks. This process is discussed more thoroughly in the discussion headline of this report.

The first step in building the model was identifying and preparing a dataset suitable for the emotion detection task.

(Adrià, please provide details about the dataset, such as its source, size, and any preprocessing steps applied)

Once the dataset was prepared, it was split into an 80% training set and a 20% test set using stratified sampling to keep the original class distribution. An initial analysis of the class distribution revealed significant imbalances in the emotion categories:

- Neutral: 20.97%
- Happy: 20.95%
- Surprise: 16.86%
- Sad: 11.95%
- Fear: 11.74%
- Angry: 10.42%
- Disgust: 7.11%

To address this imbalance, the Synthetic Minority Oversampling Technique (SMOTE) was applied to the training set, resulting in a more even distribution across classes. SMOTE is a technique to balance uneven distribution of classes by synthesizing samples in minority classes. The selection of synthesized value is decided through k-nearest neighbors within the class in question. This was done to avoid overfitting and bias in the training dataset.

Following this, a baseline random forest model was trained. This baseline model was then optimized with the help of a thorough grid search in the process of finding a better accuracy and stability. A random search was initially included in the process, which later was removed from the code to make it more concise, as its performance did not match that of the grid search using GridSearchCV.

The hyperparameters tuned during this process included:

- Number of estimators (n_estimators)
- Maximum tree depth (max_depth)
- Minimum samples for node splits and leaves (min_samples_split, min_samples_leaf)
- Number of features considered for splitting (max_features)
- Use of bootstrapping (bootstrap)

The optimization process had 5-fold cross-validation. Model performance was assessed using a classification report that included precision, recall, F1-score, and accuracy.

The above mentioned process was repeated but this time without the SMOTE to compare the performances of the resulting models. This process proved the model without the SMOTE to be the best performing model. This model had the Best Hyperparameters: `(bootstrap: True, max_depth: 30, max_features: sqrt, min_samples_leaf: 1, min_samples_split: 2, n_estimators: 100)`. The model has the accuracy of 46.46%. To asses the generalizability of the model a out-of-bag (OOB) score was also calculated. The model has an OOB-score of 0.45 which is close to the accuracy score of the model, indicating that the model is good at generalizing to unseen data [2].

C. Interaction sub-system

This section discusses the second subsystem of Bart the bartender, which handles the interaction with the user. It goes over the high-level design and its implementation. Afterwards it evaluates the obtained results.

1) Design: The interaction sub-system in this project consists of the virtual Furhat, as described in section II-A, Gemini and a connection to the video server, which has been described in section II-B1. Furhat is used as the interface between the system and the user and Gemini is used to generate a response to the input of the user. The response is based on what the user says and what emotion is detected in the face of the user at the beginning of the conversation. This response is then sent to Furhat so that it can reply to the user. This process is repeated until the right drink is found for the user.

2) Considering a group of people: When there is a group of people, the bartender considers the person who is closest to Furhat which matches the logic of how a bartender would work. If there is a queue of people waiting to order, it does not make sense to interact with a person who is not first in the queue. The logic of deciding who is closest to Furhat is decided by who has the biggest face box. If there are two people equally close Furhat, the choice of who to consider becomes undecidable and both faces may be considered. However, we deem this to be a reasonable interaction.

3) Implementation: The typical flow of this subsystem is as follows: A connection is set up between this subsystem and the video server that was described in the previous section. Once this connection is set up, Bart greets the user and asks what kind of cocktail the user wants. From this moment, the subsystem gathers the detected emotions of the user, until the user is done with speaking. From these emotions, we determine the dominant emotion of the user. When the user finishes their reply to Bart, we combine the dominant emotion with the reply of the user and send it to Gemini. Gemini is a LLM that is instructed to act like a bartender that recommends cocktails that fit the mood of the customer. We decided to use Gemini because it is free for people with a google account up to a rate limit. Our application does not use up a lot of tokens, so we should not reach this rate limit. Once Gemini has generated a response, it is forwarded to Bart. Then, Bart is instructed to say the response. After Bart is done with talking, it listens again for input of the user. The input of the user is again combined with the dominant emotion and sent to Gemini. This cycle continues until the user decides on a cocktail. After this the whole cycle repeats from the beginning, assuming there is a new customer.

4) Result: In the project proposal, we described several objectives that we expect our bartender to fullfill. We can test our product by confirming it adheres to the set objectives.

One of the objectives is that our product gives recommendations to the user, based on the user's emotional state. We add the mood of the user to the prompt that we send to Gemini. Because of this, Gemini will take both the emotion of the user as well as the user's speech into account when generating a response.

Another objective is that Bart should take the user's preferences into account. We achieve this by letting Gemini generate questions that explores the needs and preferences of the user. Gemini incorporates current and previous answers of this user into the recommendation.

The last objective is that the conversation with the system should be pleasant. This is a more subjective goal, but we try to make the conversation as pleasant as possible by continuously adjusting the prompts given to Gemini. By doing this, we hope to make the conversation as pleasant as possible.

III. General Discussion

Adria

A. Overall Pipeline

Adria

B. Challenges

Adria

The path to selecting the Random Forest model involved several steps and a lot of experimentation with different approaches. First we explored building our own neural networks, specifically multilayer feedforward neural networks using a gradient backpropagation algorithm. We tested various architectures to improve the accuracy of emotion classification. However, this approach was highly time-consuming in terms of training and optimizing the networks. The results were also often unstable, and fine-tuning the neural network models required more resources and time than the scope of our project and expertise would allow.

Next we found a pre-trained model for facial recognition from a competition on Kaggle [1]. We experimented with replication of the model adjusted to our own expertise. This did approach did however not lead to desired outcomes. Even though pre-trained models were allowed, we decided against using them unmodified because we wanted the work to reflect our own understanding and contributions. Given that the machine learning model was a major component of the project, relying on someone else's model did not feel right.

Due to this we shifted our focus to other machine learning methods. We experimented with Support Vector Machines (SVM), Boosting models, and finally, Random Forest models. After thorough testing, we found that the Random Forest model had best balance of accuracy and stability even though we would preferably have wanted to have more time to improve its results, this was however not possible considering our work load combined with other courses. The RF-model performance was consistent across test runs, and due to this it ultimately became the chosen model for our system.

C. Use of ChatGPT and other tools

ChatGPT has been used to assist in resolving error messages in the code and to discuss different ways to optimize machine learning techniques. The information provided by ChatGPT has been cross-verified with other sources, including GeeksforGeeks and the official Python documentation for specific packages. Using ChatGPT helped streamline our workflow by providing quick solutions to technical challenges and offering alternative approaches to improve model performance. Additionally, it saved significant time during debugging and allowed us to focus on implementing and testing new ideas, ultimately enhancing our understanding of machine learning concepts.

AI has also been used as a proof reader of our project report and gave us valuable help with formulation of sentences and handling of grammatical errors. This significantly improved the flow and readability of our report.

D. Ethical Issues

Adria

Conclusion

Adria

References

[1] Sanskar Hasija, “Kaggle” Available: https://www.kaggle.com/. Accessed: 2024-11-27.

[2] H. L. Smith, P. J. Biggs, N. P. French, A. N. H. Smith, J. C. Marshall, “Out of (the) bag—encoding categorical predictors impacts out-of-bag samples,” November 18, 2024. DOI: 10.7717/peerj-cs.2445.