A Dialog System for Intelligent Restaurant Recommendations: Design, Implementation and Evaluation

YOUSSEF BEN MANSOUR, 1135554, y.benmansour@students.uu.nl, Utrecht University, The Netherlands
MATS BUIS, 6205135, m.p.buis@students.uu.nl, Utrecht University, The Netherlands
LOTTE LAAN, 6125573, l.m.m.laan@students.uu.nl, Utrecht University, The Netherlands
CERINE TOUBER, 6823823, c.c.touber@students.uu.nl, Utrecht University, The Netherlands

Additional Key Words and Phrases: dialog system, machine learning, classification, reasoning

1 ABSTRACT

An AI-driven restaurant recommendation dialog system is designed and implemented based on techniques like domain modeling, ML-driven text classification and a text-based dialog system application. Trained on dialogs from the Dialog State Tracking Challenge (DSTC 2) the dialog system finds the restaurant that best suits the users (culinary) preferences. The user utterance classification system for dialog acts works on a majority class baseline, keyword matching baseline, random forest and a support vector machine (SVM) classifier. While both baseline systems have accuracies of approximately 80% the ML-models both have an accuracy of 98%, thereby outperforming the baselines significantly. Besides, the ML models outperform the baselines also on f1-score, precision and recall. The dialog system evaluates the current knowledge in each state in the state transition system and assesses based on that knowledge which utterance is posed next to the user. Based on a keyword-matching algorithm the system moves from state to state until it has found a suitable restaurant for the user. The system also uses reasoning to deduce supplementary restaurant properties. Additionally, possible improvements in the system are the use of advanced ML-models for error handling and noisy inputs. Also, the implementation of dynamic state transitions that take into account the context and avoid unnecessary questions would be beneficial. Finally, the reasoning part can be improved with probabilistic logic (like Bayesian networks) and could handle contradictions more intelligently.

2 INTRODUCTION

The dialog management system discussed here is developed to recommend restaurants by intelligently interacting with users based on natural language. To gain insight in the dataset used for the development of the dialog management system, this part is discussed in detail in section three. Here, the dialog state diagram, the choices for certain states and the language and grammar used in the data are discussed. After that the dialog system is illustrated. The dialog system roughly consists of three parts: dialog act classification, dialog management and a reasoning part to offer improved recommendations. The dialog act classification part makes use of baseline systems that work on majority class and keyword matching. Also, two machine learning models are used to maximize the accuracy of the dialog act classification. The system aims to build an interactive and robust recommendation system that is able to guide users through the process of choosing a restaurant by specifying preferences for location, price, food type and other characteristics. In addition, reasoning and rule-based inference are used to refine the recommendation and infer new information from

Authors' addresses: Youssef Ben Mansour, 1135554, y.benmansour@students.uu.nl, Utrecht University, Utrecht, The Netherlands; Mats Buis, 6205135, m.p.buis@students.uu.nl, Utrecht University, Utrecht, The Netherlands, m.p.buis@students.uu.nl; Lotte Laan, 6125573, l.m.m.laan@students.uu.nl, Utrecht University, Utrecht, The Netherlands, l.m.m.laan@students.uu.nl; Cerine Touber, c.c.touber@students.uu.nl, 6823823, c.c.touber@students.uu.nl, Utrecht University, Utrecht, The Netherlands.

existing knowledge. For example, spending a long time in a restaurant is romantic. Users interact with the dialog system entirely via natural language by inputting text. The system parses the user utterances and extracts preferences (e.g Italian food) from them to obtain knowledge. Based on matching preferences, the system offers the user a suitable restaurant. The state-transition system is flexible and handles user input in many orders. Also, the system can handle additional user preferences and contradictions.

The first component of the system is the dialog act classifier as further discussed in section four, the Machine Learning section. In the system there is the option to choose from one of the two baseline models. The majority class baseline which always selects the dialog act that occurs most often in the dataset. While this classifier generally achieves a remarkably high score, it is not the most intelligent classifier. Besides, a keyword matching baseline is available that classifies the dialog act based on the selected keywords for a specific dialog act. For example, if bye is in an utterance, the utterance is automatically classified as goodbye. In addition, two machine learning models are used: a random forest classifier and a support vector machine. The random forest has multiple "trees" that are trained on a subset of the training data and classify the dialog act independently. Consequently the classifications are aggregated and a decision is made which dialog act has a majority vote. The support vector machine classifies the dialog acts by transforming the utterances into vectors. Then, the SVM tries to find a hyperplane that separates the dialog act categories in this feature space. A new utterance is then classified based on the side of the hyperplane it is on.

In component two of the dialog system, the dialog manager, the conversation flow is managed by extracting the user preferences through state transitions in the state-transition system. On this part is elaborated further in section five, the dialog manager. For every prompt information is extracted from the utterance such as food type, location and price range. Based on this user input the system transitions from state to state. For example, when the user inputs north of town, the location is known and system consequently transitions to food type. This feature is strengthened by the use of the Levenshtein edit distance for handling spelling errors.

The third component of the dialog management system consists of the reasoning and configurability part. This will be discussed in further detail in sections six and seven respectively. The inference rules to infer additional properties of a restaurant from already known information. For example, a busy restaurant is not considered romantic hence this is not suggested when the user searches for a romantic restaurant while the user has not directly entered to avoid busy restaurants. Also, the system can handle contradictions. Hence, if a user enters it wants a romantic and a busy restaurant then the systen knows how to choose from these two user preferences. Furthermore, several configurations are available to the user such as the Levenshtein distance for preference extraction, allowance of dialog restarts and the number of recommendations offered to the user.

3 DATA

The dataset used in this analysis is called *dialog_acts.dat*. It is stored in a .dat file that contains only plain text representing user and system utterances from a dialog management database designed to provide restaurant recommendations based on user preferences. An example of a line in the database is: *"inform I need a restaurant in the west part of town that serves Europ."* The first word in this sentence, *"inform,"* represents the dialog act, while the remainder of the sentence, *"I need a restaurant in the west part of town that serves Europ,"* is the example sentence. The sentences in the dataset are derived from both user and system utterances. The system begins the dialog by asking for the user"s preferences. The user responds, and the system then asks for confirmation or provides suggestions based on the user"s utterance. Some example sentences contain typographical errors, such as *"Europ"* or *"addrss,"* while others include informal words, such as *"sil"* or *"um."* These errors can complicate the system"s ability to recognize and classify specific dialog acts.

The dataset contains 15 dialog act, an example of the description of the dialog acts is as followed: "ack" stands for "acknowledgement". The most frequently reported dialog act in the dataset is *inform*, with a frequency of 10,160 occurrences, while the least frequent is *reqmore*, which appears only 5 times. For all the descriptions and frequencies, see (*appendix: table 1*).

The total number of dialogs in the database is 22,501. To understand how the dialog acts are distributed, see *figure 1* in the appendix. However, many sentences, also referred to as utterances, are duplicated. The dataset contains a duplication rate of 78.99%. After removing duplicated utterances, the cleaned database has a total of 5,359 utterances. The example sentences in the database, before deduplication, have an average length of 19.27 characters, with a standard deviation of 15.27 characters. The average word count per example sentence is 3.73, with a standard deviation of 2.94. In contrast, the example sentences in the deduplicated database have an average length of 34.29 characters and a standard deviation of 19.47 characters. The average word count for these sentences is 6.50, with a standard deviation of 3.72. This indicates that many short words were duplicated in the original database, as the average character and word counts are higher in the deduplicated dataset.

In the figure (*appendix: figure 2*), we see the transitions from state to state. The user provides information based on what the system says and asks. In turn, the system requests information or gives a restaurant recommendation based on the user"s input. To give a good overview of how the dialog state diagram works, here are some snippets:

State 1: Welcome State
System: "Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range, or food type. How may I help you?"
User: "expensive"
Speech Act: inform(pricerange=expensive)
Next state: State 3: AskUserForClarification

This dialog represents the Welcome state (1) where the system begins the interaction by greeting the user and asking how it can help. The user"s input provides information about the price range, so the system transitions to the next step where it asks for more information. In this case the system wants to know more information about the area, and the type of food preferences. So the system goes to AskUserForClarification state (3). Another snippet example is:

State 2: AskForMissingInfo
Dialog:
System: "What part of town do you have in mind?"
User: "east"
Speech Act: inform(area=east)

Next state: State 3: AskUserForClarification or State 4: Ask for confirmation of preferences

In this snippet, the system is in the AskForMissingInfo state (2), where it is collecting essential information to assist the user effectively. The system prompts the user for their preferred area, encouraging a response. When the user replies with "east," they provide specific information regarding their area preference. The associated speech act indicates that the user is informing the system of this preference. Based on this interaction, the system can transition to either the AskUserForClarification state (3) or the AskForConfirmationOfPreferences state (4). It will move to the AskUserForClarification state if further details are required to fulfill the user's request—such as their food type or budget preferences. The system will continue to request this information until it has gathered user preferences across all three categories. If the information provided is adequate to confirm the user's preferences, the system will transition to the AskForConfirmationOfPreferences state (4).

4 MACHINE LEARNING

During the project, we trained four different models: two baseline models and two machine learning models. Before training the models, we preprocessed the data by converting it to lowercase and dividing it into two columns: one for the "dialog act" and one for the "utterance content".

4.1 Baseline Models

The baseline models are designed to provide a performance comparison with the more complex machine learning models discussed later in this article. Two baseline models have been created: Baseline 1, which is based on the most common dialog act, and Baseline 2, which relies on keyword identification.

4.1.1 Baseline 1. The first baseline model is based on classification that uses the majority class. It predicts the most common class in the training dataset for all instances. In this case, the majority class of the training dataset is the "inform" dialog act. Because it relies on the majority class, the model is not very sensitive and performs poorly compared to the other models.

The performance of the the Baseline 1 model (*appendix: table 2*) applied to duplicated data, the accuracy was 0.40. The precision was low for most classes, for example in “ack"", “affirm"", and “bye"", which had a precision of 0.00. The recall showed the class “inform"" achieved a perfect recall of 1.00. The F1-score for this model was low, therefor it was a model with poor performance, mainly for underrepresented classes.

The Baseline 1 with deduplicated data, (*appendix: table 3*) the accuracy increased to 0.56. The trends in precision and recall remained similar, with “inform"" performing better than other classes, but most with a precision of 0.00.

4.1.2 Baseline 2. The second model is a keyword-based classification model. This model maps the user"s input sentences to dialog act categories by relying on specific words or phrases present in the user"s sentence. It provides an easy way to understand what the model is doing and how it classifies the user"s input. However, it can be challenging to manually identify keywords for classification, as the number of keywords can continuously increase. Additionally, since the model is rule-based, it does not have the capability to learn or adapt over time.

The accuracy of Baseline 2 with duplicated data (*appendix: table 4*) was measured at 0.90. The results for some classes had a good performance. The classes "affirm"" and "inform"" scored with f1-scores of 0.97 for "affirm"" and 0.91 for "inform"", indicating good classification performance.

The performance of the Baseline 2 model with deduplicated data (*appendix: table 5*) increased to 0.70. Classes, such as "affirm"" and "inform"", had significant precision, indicating that the performance was better than in Baseline1. The recall was also improved. However, the model still performed poorly for smaller classes.

4.2 Machine learning models

In this section, two machine learning models used for dialog act classification are explained in further detail: the Random Forest and Support Vector Machines (SVM) classifiers. Both models use supervised learning techniques to categorize user utterances into dialog acts. Both machine learning models are trained on a subset of the dialog acts dataset.

4.2.1 RandomForest Models. A Random Forest is a supervised learning model that can be used for tasks like regression or classification. The intuition behind a Random Forest is that training multiple unrelated models is more accurate than training one model would do alone. The multiple models are referred to as trees within a forest, hence the name Random Forest. A Random Forest consists of multiple trees that make a decision in a classification task. The decision returned by a Random Forest is consequently the mode or the mean of the trees" decisions. In case of the dialog act classification RF-classifier the trees predict for each utterance which dialog act should be assigned to it. Since the dialog act refers to a categorical variable the RF-classifier returns the mode of the trees as predicted dialog act. The model is consequently evaluated with cross-validation.

The accuracy of the RandomForest model with duplicated data was 0.98 (*appendix: table 6*). The precision was high across all classes, with good results for "affirm"", "bye"", and "inform"". The recall was good, was means that the model was good for almost all the classes. Additionally, the F1-score was also high, which means that there was a good balance between precision and recall.

The accuracy of the RandomForest model on deduplicated data was 0.89 (*appendix: table 7*), a bit less then the accuracy of the RandomForest model with duplicated data. While some classes, such as "ack"", had a precision of 0.00, the class "inform"" had a high precision and recall.

The advantage of a Random Forest classifier is that it handles high-dimensional data and non-linear relationships well. Also, a Random Forest does not overfit fast and it can provide the importance of features which increases interpretability.

4.2.2 SVM Models. Support vector machines (SVM) are supervised learning machine learning models that predict the chance of a datapoint belonging to a certain category. This is done by maximizing the distance between the division, a line or hyperplane, and the datapoints belonging to different categories. Whether this is a line on a 2-dimensional plane or a plane in a hyperdimensional plane, is determined by the number of categories in a dataset. In this case the classes define the dimensionality of the hyperplane. The features are the utterances and the labels are the dialog acts. The datapoints that lie closest to the line or hyperplane are and define the margin are called the support vectors. After training the model the SVM can predict to which category a new datapoint belongs. The performance metrics for the

SVM model trained on the dialog acts dataset is given in 8. Several parameters that can be tuned within the SVM are the kernel type, regularization strength, gamma and class weights.

The accuracy of the SVM model with duplicated data was 0.98. Which is the same as the RandomForest model. The precision was high for “affirm"", “bye"", “inform"", and “thankyou"". The recall was overall good, but could be better with some classes. generally good but exhibited slight decreases for certain classes. The F1-score was also good.

The accuracy of the SVM model with duplicated data was 0.96. Precision is high. While precision remained high overall. Additionally, certain classes, such as “null"" and “repeat"", had low recall and precision. For a summary of inccrectly classified utterances, see *appendix: table 10*

4.3 Difficult cases

The difficult cases section dives deeper into challenging utterances for the classification systems to unravel the robustness of the models. Based on the utterances that were wrongly classified, three difficult cases are distinguished: utterances starting with and, utterances with a negation and utterances with a negation. To further test the ability of the systems to classify difficult cases, two test sets of utterances are composed for each difficult case: one set with utterances that occur in the dataset or show strong similarity with utterances in the dataset and one set of utterances that do not occur in the dataset.

As can be observed in appendix *table 12* and appendix *table 11*, the precision, recall, f1-score and accuracy is lower for every set of utterance set that does not occur in the dataset. This outcome is expected, given that the models have been trained on a subset of the utterances. Additionally, the SVM demonstrates better performance compared to the Random Forest model for "and" utterances not found in the dataset, which means it has a better performance when implementing new utterances. The unaggregated data can be found in tables 10 up until and including table 21 in the appendix.

For the negation utterances, both models show improvement when the utterances are part of the training data. However, the SVM"s performance on negation utterances not present in the dataset has a lower performance than that of Random Forest. While Random Forest is better in classifying negation utterances that are not in the training dataset, SVM maintains a higher F1-score for the negation utterances included in the dataset. Both models have a low performance on utterances that include typo erros. Nevertheless, SVM achieves a little better precision and F1-score for typo errors not present in the dataset, while both models perform almost the same for those within the dataset.

In summary, the SVM model generally has a better performance compared to the Random Forest model across various metrics for difficult cases, particularly when handling utterances that are not included in the training data.

5 DIALOG MANAGER

The state transition function determines how the system changes states dependent on the user input. The state transition function is located in the *SystemDialog* class. This class has a member variable called *State()* which keeps track of the current state by means of the following attributes:

- Current_state (string)

- User preferences dictionary
- Still required information (list of 0 to 3 of the following: area, food, pricerange)
- Dialog acts instance that contains all the functions that are used to determine the new user utterance and the new state
- Helpers instance that contains all the helper functions
- Last system utterance (string)
- Ambiguity dictionary
- Found restaurants list: all the restaurants that meet the user preferences (but not the user's additional requirements)
- Filtered restaurant list: all the restaurants that meet the user preferences and the user's additional requirements.
- Currently selected restaurant (Restaurant type)
- File path to the restaurant .csv file
- Additional requirements dictionary.

When `state_transition` is called in the `SystemDialog` class, `state_transition` gets the self-object containing that state that was described above, the current user input and the predicted class.

Based on the predicted class, `state_transition` then calls the appropriate function and always passes the state as an argument. In most cases, it passed the user input as an argument too, but in cases where it is not necessary to determine the system utterance it will not do that.

For example, when the predicted class is "bye", the system must return “Goodbye! Enjoy your meal!” and there is no need to know what the content of the user input was. There is no need to know the content of the user input. Since the reference to the state is passed along in `state_transition`, each time `state.current_state` is changed in the functions themselves, without explicitly returning `current_state`. The new system utterance is always returned explicitly.

The system"s state transitions are demonstrated through example dialogues in the table 25. The system starts in the Welcome state (1). It can move out of the Welcome state as soon as the user gives a certain input. From here it can go to one of these 3 states: AskForMissingInformation1 (2), AskForClarification (3) or AskForConfirmation1 (4).

All the state-transitions can be found in the Appendix section B and C, and table 25

6 REASONING

The dialog agent will begin by asking for the user"s primary preferences, such as price range, food type, and restaurant location. Once these preferences are gathered and confirmed, the system will inquire about additional preferences, like whether the user prefers a restaurant to be touristic, romantic, child-friendly, or offer assigned seats. These additional requirements will influence the selection of restaurants by applying a set of predefined logical rules based on the information in the dataset. It must be noted that three new columns have been added to the dataset:

- *Crowdedness*: Indicates how busy the restaurant is, with values "busy" or "empty."
- *length_of_stay*: Represents how long customers typically stay, with values "short" or "long".
- *Food qualit**: Describes the quality of the food, with values "normal," "good," or "excellent."

6.1 Rules

The following rules are formulated to classify restaurants based on previously known information:

- If a restaurant is *cheap* and has good or excellent food, it will be classified as *touristic*.
- If a restaurant serves *Romanian* cuisine, it will be classified as *non-touristic*.
- If a restaurant is *busy*, it will be classified as having assigned *seats*.
- If a restaurant is *busy*, it will also be classified as *non-romantic*.
- If a restaurant has *long* stays, it will be classified as *non-child-friendly*.
- If a restaurant has *long* stays, it will be classified as *romantic*.

After gathering this additional information from the user, the system will apply these rules to the restaurant dataset and recommend a restaurant based on the user"s preferences.

Example:

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: i want to eat cheap Chinese food in the south
System: Let me confirm, you are looking for a Chinese restaurant in the south of town in the cheap price range, right?
System: Are there any additional preferences you"d like to specify such as romantic atmosphere,requiring a reservation, or being child-friendly?
Me: I want a romantic and touristic restaurant
System: To clarify, you prefer a restaurant with these qualities: Touristic, Romantic.
Me: Yes
System: I recommend the missing sock in the south area, serving Chinese cuisine, with cheap prices. It is a touristic hot-spot due to the cheap prices and the good food. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.

6.2 Contradictions

During the interaction with the dialog agent, users may express preferences that contradict each other. For instance, a user might request a restaurant that is both romantic and has assigned seats. These two preferences can create a conflict because a romantic atmosphere typically implies a quieter, while assigned seats implies a busy establishment. When the dialog agent detects such contradictions, it will respond in a considerate and informative manner. Specifically, the agent will first acknowledge the user"s request and then highlight the contradictory of the preferences. The system will not let the user continue unless it corects the contradictory preferences.

Example:

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: i want to eat cheap Chinese in the south of town
System: Let me confirm, you are looking for a Chinese restaurant in the south of town in the cheap price range, right?

Me: Yes
System: Are there any additional preferences you"d like to specify such as romantic atmosphere,requiring a reservation, or being child-friendly?
Me: romantic and assigned seats
System: To clarify, you prefer a restaurant with these qualities: Romantic, with assigned seats.
Me: Yes
System: Sorry but you can not choose romantic and a assigned seat restaurant, can you change your requirement please
Me: Romantic
System: To clarify, you prefer a restaurant with these qualities: Romantic.
Me: Yes
System: I recommend the missing sock in the south area, serving Chinese cuisine, with cheap prices. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.

7 CONFIGURABILITY

The dialog system is more than a vanilla dialog system in which user features are already determined on forehand. User-friendliness and adaptability to user needs is important and has been taken into account during the system"s design and modelling phase. This resulted in the addition of four configurability options:

- Allowing for dialog restarts
- User delay
- Output in caps
- Usage of baseline classifiers

These features are defined as global (Boolean) variables so that users can easily turn them on or off by changing the boolean value of these variables.

System configurations in the code are managed by several boolean variables, each controlling a specific behavior:

(1) **Dialog Restart:** The *allow_dialog_restarts* variable determines if the dialog can be restarted. If set to *True*, it reinitializes preferences and starts over. If *False*, the system informs the user that restarting isn"t allowed and displays the message: "I'm sorry. You are not allowed to restart the dialog."" This ensures user preferences remain fixed throughout the conversation when restarts are not permitted.
(2) **User Delay:** When *use_delay* is *True*, the system adds a delay (via the *time* library) before displaying responses. If *False*, the response is immediate.
(3) **Input Capitalization:** The *output_in_caps* variable handles input case sensitivity. If *False*, the system converts input to lowercase, treating "Food"" and "food"" as identical. If *True*, the model will have no case sensitivity.
(4) **Classifier Selection:** The boolean *use_baseline_as_classifier* selects between classifiers. If set to *True*, the system uses the baseline classifier for input classification. If *False*, it switches to the Random Forest classifier, which provides more sophisticated classification.

8 CONCLUSION

The dialog management system developed during this project provides a way to recommend restaurants by engaging users through the use of natural language interactions. The system consists of three main components: dialog act classification, dialog management and reasoning-based recommendations. The first part provides a way to classify the dialog acts with a majority class baseline, a keyword matching baseline, a Random Forest classifier and an SVM-classifier to enhance the accuracy of dialog act classification. The dialog management part handles the input of user preferences by using a state-transition system that accounts for flexibility in user inputs, while the reasoning part refines the recommendations and infers information from existing preferences.

While the system poses a way to dynamically interact with a (partially) AI-driven dialog management system, it has also its limitations. For example, the majority class and keyword matching components of the baselines could not always understand the input of user intents so that the handling of complex inputs can be further improved. Even though machine learning models like Random Forests and SVM"s are effective and show promising results, there may be more advanced algorithms such as deep learning-based approaches that could further improve the classification accuracy.

Furthermore, improvements can be made in the dialog management part by integrating more complex natural language processing techniques to improve contextual understanding and sentiment in user inputs. In this way, user intent and preferences can be more accurately extracted from the user inputs. Future work can also focus on the refinement of handling contradictory inputs with greater precision.

While the current dialog management system offers a foundation for interactive restaurant recommendations, there are opportunities for further improvement. Such as enhancing the machine learning models and increasing the system"s ability to handle more intricate and complex user preferences. Also, further improving the contextual understanding of user inputs through sentiment analysis and more nuanced preference extraction, would make it a more sophisticated dialog management system.

9 DIVISION OF WORK

The team has worked mostly on the coding for a specific section and also wrote that section in the report. Hence, the hourly work is stated per section which thus includes both coding and writing. The coding part usually takes up considerably more time than the writing part. The general rule of thumb is 80-20 (coding-writing).

Task	Cerine	Lotte	Mats	Youssef
Abstract			1hr	
Introduction			2hr	
Data Section		13hr		5hr
Machine Learning Section	3hr	12hr	12hr	2hr
Difficult Cases Section		3hr	7hr	
Error Analysis Section	5hr			
Dialog Manager Section	15hr	8hr	1hr	12hr
Reasoning Section	10hr			12hr
Configurability	2hr			
Conclusion			1hr	
Coding (other)	10hr	7hr	12hr	13hr
Final Report Editing	4hr	7hr	14hr	4hr
Reading and Understanding Data	2hr	2hr	2hr	2hr
Code Optimization	2hr	2hr	2hr	5hr
Total	**57hr**	**54hr**	**54hr**	**55hr**

Table 1. Individual contributions and time spent on tasks

A APPENDIX

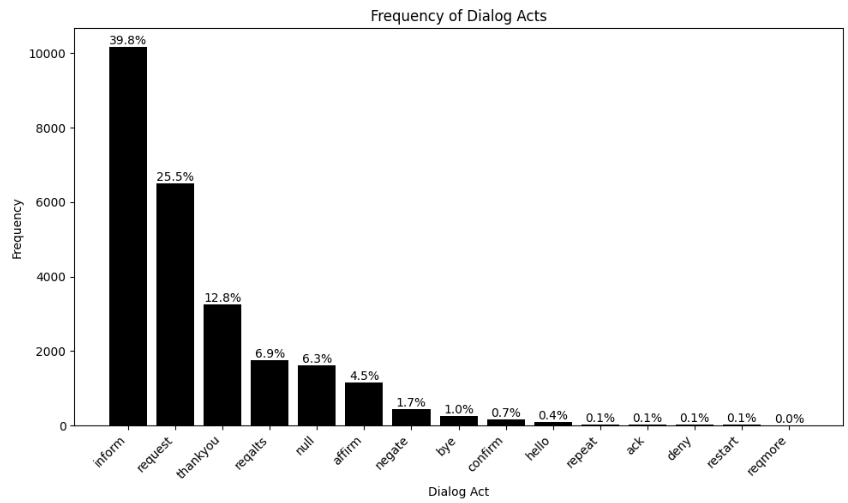

Fig. 1. Shows the frequency and percentage of dialog acts that appear in the duplicated data.

Table 1. Number of Dialogs per Category

Category	Description	Number of Dialogs
inform	state a preference or other information	10160
request	ask for information	6494
thankyou	express thanks	3259
reqalts	request alternative suggestions	1747
null	noise or utterance without content	1612
affirm	positive confirmation	1156
negate	negation	435
bye	greeting at the end of the dialog	266
confirm	check if given information confirms to query	172
hello	greeting at the start of the dialog	93
repeat	ask for repetition	33
ack	acknowledgement	28
deny	reject system suggestion	27
restart	attempt to restart the dialog	14
reqmore	request more suggestions	5
Total Number of Dialogs	-	**25501**

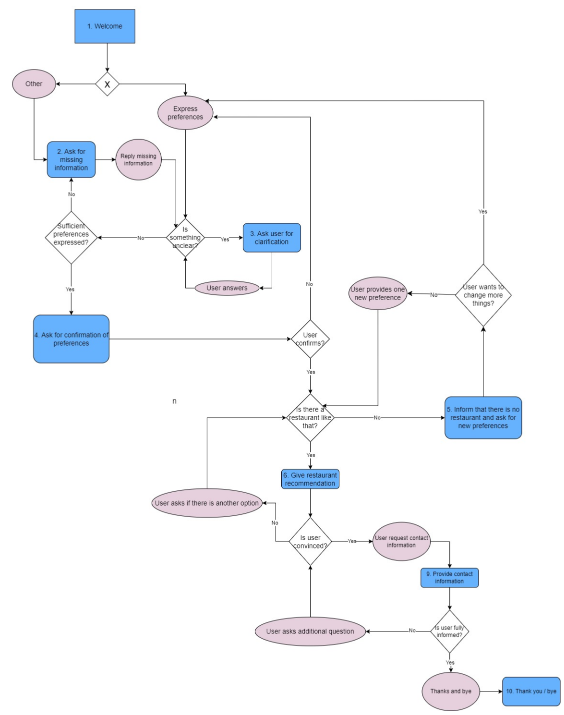

Fig. 2. State dialog diagram: First version

Table 25. Full dialogue

Full dialogue
System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: Hi!
System: Could you provide me more information about your preferred area, price range and food type?
Me: yess I would like chinese
System: Did you say you are looking for a restaurant that serves chinese food?
Me: yes
System: Okay. Your food preference is now set to chinese. What part of town do you have in mind?
Me: centre
System: Would you like something in the cheap, moderate, or expensive price range?
Me: cheap
System: Let me confirm, you are looking for a chinese restaurant in the centre of town in the cheap price range, right?

Me: no wait i want in the south
System: I'm sorry. What is your preference for food, price range and area?
Me: chinese restaurant in the south
System: Would you like something in the cheap, moderate, or expensive price range?
Me: cheap
System: Let me confirm, you are looking for a chinese restaurant in the south of town in the cheap price range, right?
Me: yes
System: Are there any additional preferences you'd like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: yes, romantic please
System: Are there any additional preferences you'd like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: romantic
System: To clarify, you prefer a restaurant with these qualities: Romantic.
Me: no
System: Which additional preferences would you like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: reservation
System: To clarify, you prefer a restaurant with these qualities: with assigned seats.
Me: yes
System: I recommend the missing sock in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: is there anything else?
System: I recommend rice house in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: what about british food?
System: I recommend the lucky star in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: does it serve british food?
System: the lucky star is a nice place serving chinese food.
Me: i don't want that. I want british food.
System: For the food preference, would you like chinese or british?
Me: british
System: Let me confirm, you are looking for a british restaurant in the south of town in the cheap price range, right?
Me: yes
System: Which additional preferences would you like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: romantic

System: To clarify, you prefer a restaurant with these qualities: Romantic. Me: yes System: Sorry, I couldn't find a cheap british restaurant in the south. Please change your requirements. Me: what about the centre System: I recommend the cambridge chop house in the centre area, serving british cuisine, with cheap prices. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.
Me: thank you
System: You're welcome.
Me: can i have the address and the phone number
System: The address of the cambridge chop house is on 1 kings parade. The phone number of the cambridge chop house is 01223 359506.
Me: thanks, bye!
System: You're welcome. Good bye.

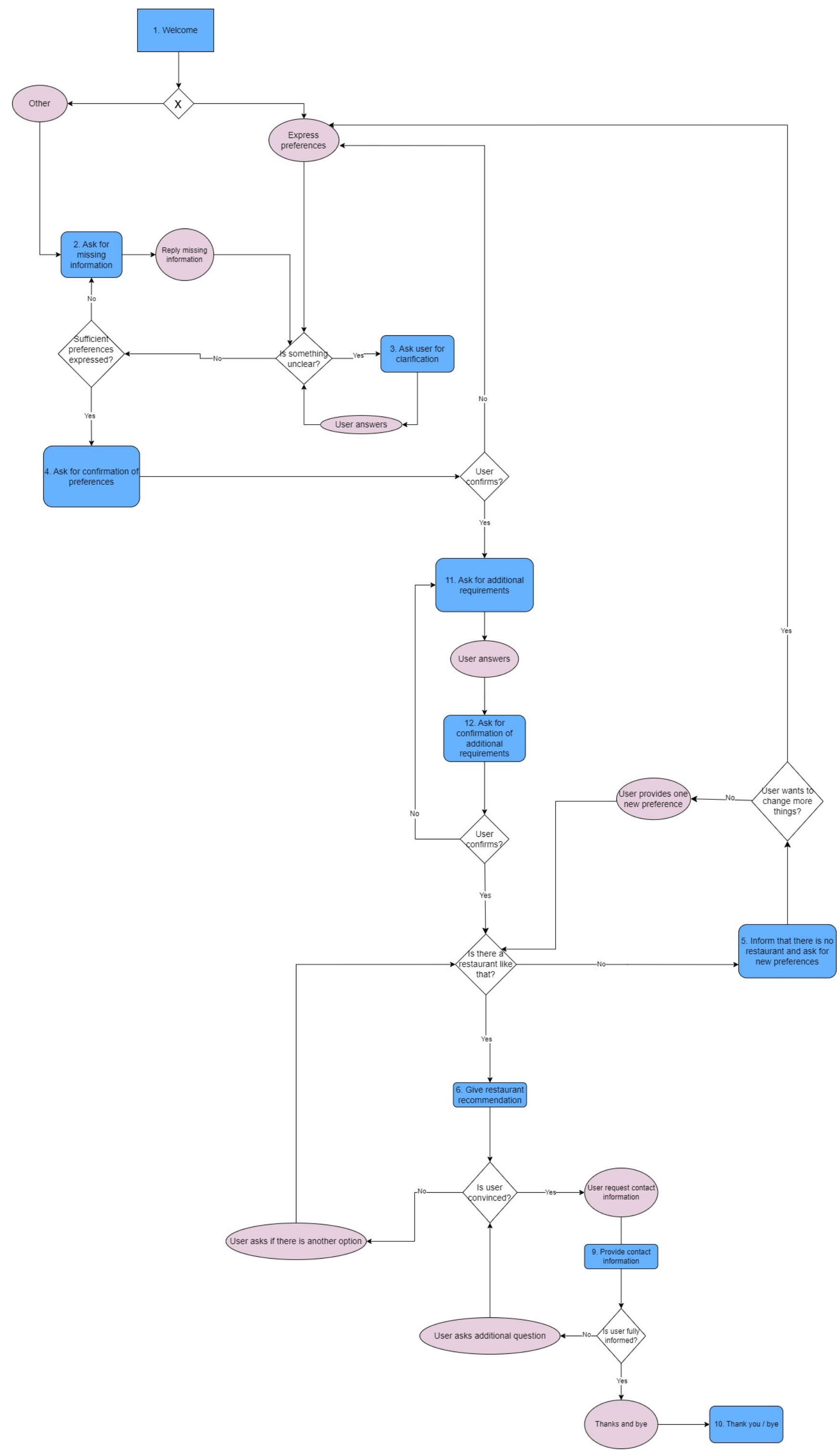

Fig. 3. State dialog diagram: Second version

Table 2. Baseline1: Report of duplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	5
affirm	0.00	0.00	0.00	180
bye	0.00	0.00	0.00	35
confirm	0.00	0.00	0.00	22
deny	0.00	0.00	0.00	6
hello	0.00	0.00	0.00	14
inform	0.40	1.00	0.57	1532
negate	0.00	0.00	0.00	69
null	0.00	0.00	0.00	232
repeat	0.00	0.00	0.00	3
reqalts	0.00	0.00	0.00	279
reqmore	0.00	0.00	0.00	1
request	0.00	0.00	0.00	972
restart	0.00	0.00	0.00	2
thankyou	0.00	0.00	0.00	474
Accuracy	**0.40**	-	-	**3826**
Macro avg	**0.03**	**0.07**	**0.04**	**3826**
Weighted avg	**0.16**	**0.40**	**0.23**	**3826**

Table 3. Baseline1: Report of deduplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	3
affirm	0.00	0.00	0.00	24
bye	0.00	0.00	0.00	9
confirm	0.00	0.00	0.00	17
deny	0.00	0.00	0.00	1
hello	0.00	0.00	0.00	5
inform	0.56	1.00	0.71	447
negate	0.00	0.00	0.00	25
null	0.00	0.00	0.00	46
repeat	0.00	0.00	0.00	1
reqalts	0.00	0.00	0.00	77
request	0.00	0.00	0.00	135
restart	0.00	0.00	0.00	1
thankyou	0.00	0.00	0.00	13
Accuracy	**0.56**	-	-	**804**
Macro avg	**0.04**	**0.07**	**0.05**	**804**
Weighted avg	**0.31**	**0.56**	**0.40**	**804**

Table 4. Baseline2: Report of duplicated data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.06	0.20	0.10	5
affirm	0.99	0.96	0.97	180
bye	1.00	0.74	0.85	35
confirm	0.00	0.00	0.00	22
deny	0.62	0.83	0.71	6
hello	0.83	0.36	0.50	14
inform	0.95	0.88	0.91	1532
negate	0.79	0.61	0.69	69
null	0.90	0.72	0.80	232
repeat	0.67	0.67	0.67	3
reqalts	0.96	0.84	0.90	279
reqmore	1.00	1.00	1.00	1
request	0.92	0.98	0.95	972
restart	1.00	0.50	0.67	2
thankyou	0.94	1.00	0.97	474
unknown	0.00	0.00	0.00	0
Accuracy	**0.90**	-	-	**3826**
Macro avg	**0.73**	**0.64**	**0.67**	**3826**
Weighted avg	**0.93**	**0.90**	**0.91**	**3826**

Table 5. Baseline2: Report of deduplicated data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	3
affirm	0.91	0.83	0.87	24
bye	0.83	0.56	0.67	9
confirm	0.00	0.00	0.00	17
deny	0.00	0.00	0.00	1
hello	0.33	0.20	0.25	5
inform	0.86	0.81	0.84	447
negate	0.58	0.28	0.38	25
null	0.08	0.02	0.03	46
repeat	1.00	1.00	1.00	1
reqalts	0.81	0.51	0.62	77
request	0.71	0.87	0.78	135
restart	1.00	1.00	1.00	1
thankyou	0.37	0.77	0.50	13
unknown	0.00	0.00	0.00	0
Accuracy	**0.70**	-	-	**804**
Macro avg	**0.50**	**0.46**	**0.46**	**804**
Weighted avg	**0.75**	**0.70**	**0.71**	**804**

Table 6. RandomForest: Report of duplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	1.00	0.20	0.33	5
affirm	1.00	0.97	0.98	247
bye	0.98	1.00	0.99	46
confirm	0.94	0.88	0.91	33
deny	1.00	1.00	1.00	7
hello	1.00	0.75	0.86	16
inform	0.98	0.99	0.99	2041
negate	1.00	0.90	0.95	81
null	0.95	0.97	0.96	309
repeat	1.00	0.75	0.86	4
reqalts	0.96	0.96	0.96	368
reqmore	1.00	1.00	1.00	2
request	0.99	0.99	0.99	1305
restart	1.00	1.00	1.00	3
thankyou	1.00	1.00	1.00	634
Accuracy	**0.98**	-	-	**5101**
Macro avg	**0.99**	**0.89**	**0.92**	**5101**
Weighted avg	**0.98**	**0.98**	**0.98**	**5101**

Table 7. RandomForest: Report of deduplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	5
affirm	1.00	0.71	0.83	34
bye	0.75	0.75	0.75	12
confirm	0.72	0.82	0.77	22
deny	0.00	0.00	0.00	1
hello	1.00	0.33	0.50	9
inform	0.93	0.95	0.94	604
negate	0.97	0.94	0.95	31
null	0.53	0.58	0.55	59
repeat	1.00	1.00	1.00	1
reqalts	0.88	0.84	0.86	99
request	0.90	0.95	0.93	176
restart	1.00	1.00	1.00	1
thankyou	0.94	0.83	0.88	18
Accuracy	**0.89**	-	-	**1072**
Macro avg	**0.76**	**0.69**	**0.71**	**1072**
Weighted avg	**0.89**	**0.89**	**0.89**	**1072**

Table 8. SVM: Report of Duplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	1.00	0.20	0.33	5
affirm	0.99	0.97	0.98	180
bye	0.94	0.97	0.96	35
confirm	0.83	0.86	0.84	22
deny	1.00	0.83	0.91	6
hello	1.00	0.86	0.92	14
inform	0.98	0.99	0.98	1532
negate	1.00	0.99	0.99	69
null	0.98	0.92	0.95	232
repeat	1.00	0.67	0.80	3
reqalts	0.96	0.97	0.96	279
reqmore	1.00	1.00	1.00	1
request	0.99	0.99	0.99	972
restart	1.00	1.00	1.00	2
thankyou	1.00	0.99	1.00	474
Accuracy	**0.98**	-	-	**3826**
Macro avg	**0.98**	**0.88**	**0.91**	**3826**
Weighted avg	**0.98**	**0.98**	**0.98**	**3826**

Table 9. SVM: Report of Deduplicated Data. Precision, Recall, F1-score, and Support for Each Class

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	3
affirm	1.00	0.88	0.93	24
bye	1.00	0.67	0.80	9
confirm	0.83	0.88	0.86	17
deny	1.00	1.00	1.00	1
hello	1.00	1.00	1.00	5
inform	0.89	0.99	0.94	447
negate	0.96	0.92	0.94	25
null	0.80	0.17	0.29	46
repeat	0.00	0.00	0.00	1
reqalts	0.88	0.88	0.88	77
request	0.96	0.94	0.95	135
restart	1.00	1.00	1.00	1
thankyou	0.93	1.00	0.96	13
Accuracy	**0.91**	-	-	**804**
Macro avg	**0.80**	**0.74**	**0.75**	**804**
Weighted avg	**0.90**	**0.91**	**0.89**	**804**

Table 10. Number of incorrectly classified utterances

Model	With Duplicates	Without Duplicates
Baseline 1	2294	357
Baseline 2	389	239
RandomForest	79	117
SVM	68	75

Table 11. Classification outcomes for difficult cases (Randomforest)

Difficult case	**Precision**	**Recall**	**F1-Score**	**Accuracy**
And utterances not in data	0.31	0.16	0.17	0.17
And utterances in data	0.80	0.74	0.73	0.74
Negation utterances not in data	0.34	0.24	0.22	0.26
Negation utterances in data	0.80	0.70	0.69	0.70
Typo utterances in data	0.38	0.16	0.16	0.17
Typo utterances in data	0.64	0.45	0.47	0.45

Table 12. Classification outcomes for difficult cases (SVM)

Difficult case	**Precision**	**Recall**	**F1-Score**	**Accuracy**
And utterances not in data	0.45	0.20	0.23	0.21
And utterances in data	0.87	0.76	0.75	0.76
Negation utterances not in data	0.32	0.14	0.14	0.15
Negation utterances in data	0.74	0.61	0.61	0.61
Typo utterances in data	0.50	0.23	0.24	0.25
Typo utterances in data	0.64	0.39	0.40	0.39

Table 13. SVM: and utterances not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	0.80	0.40	0.53	10
bye	1.00	0.10	0.18	10
confirm	0.29	0.20	0.24	10
deny	0.00	0.00	0.00	10
hello	1.00	0.10	0.18	10
inform	0.12	0.50	0.19	10
negate	0.14	0.10	0.12	10
null	0.05	0.30	0.09	10
repeat	1.00	0.30	0.46	10
reqalts	0.00	0.00	0.00	0
reqmore	0.75	0.30	0.43	10
requalts	0.00	0.00	0.00	10
request	0.20	0.10	0.13	10
restart	1.00	0.40	0.57	10
thankyou	0.80	0.40	0.53	10
Accuracy	**0.21**	-	-	**150**
Macro avg	**0.45**	**0.20**	**0.23**	**150**
Weighted avg	**0.48**	**0.21**	**0.24**	**150**

Table 14. SVM: and utterances in data

Class	Precision	Recall	F1-score	Support
ack	1.00	0.20	0.33	10
affirm	1.00	0.80	0.89	10
bye	0.77	1.00	0.87	10
confirm	1.00	0.80	0.89	10
deny	1.00	0.20	0.33	10
hello	1.00	1.00	1.00	10
inform	0.43	1.00	0.61	10
negate	0.91	1.00	0.95	10
null	0.41	0.90	0.56	10
repeat	1.00	0.80	0.89	10
reqalts	0.69	0.90	0.78	10
reqmore	1.00	0.50	0.67	10
request	0.82	0.90	0.86	10
restart	1.00	0.90	0.95	10
thankyou	1.00	0.50	0.67	10
Accuracy	**0.76**	-	-	**150**
Macro avg	**0.87**	**0.76**	**0.75**	**150**
Weighted avg	**0.87**	**0.76**	**0.75**	**150**

Table 15. SVM: utterances with negation not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	0.00	0.00	0.00	10
bye	1.00	0.10	0.18	10
confirm	0.00	0.00	0.00	10
deny	0.00	0.00	0.00	10
hello	1.00	0.10	0.18	10
inform	0.11	0.80	0.19	10
negate	0.00	0.00	0.00	10
null	0.12	0.30	0.17	10
repeat	0.67	0.20	0.31	10
reqalts	0.00	0.00	0.00	0
reqmore	0.67	0.20	0.31	10
requalts	0.00	0.00	0.00	10
request	0.00	0.00	0.00	10
restart	1.00	0.40	0.57	10
thankyou	0.50	0.20	0.29	10
Accuracy	**0.15**	-	-	**150**
Macro avg	**0.32**	**0.14**	**0.14**	**150**
Weighted avg	**0.34**	**0.15**	**0.15**	**150**

Table 16. SVM: utterances with negation in data

Class	Precision	Recall	F1-score	Support
ack	1.00	0.20	0.33	10
affirm	1.00	0.30	0.46	10
bye	0.83	0.50	0.62	10
confirm	1.00	0.70	0.82	10
deny	0.00	0.00	0.00	10
hello	1.00	1.00	1.00	10
inform	0.23	1.00	0.38	10
negate	0.53	1.00	0.69	10
null	0.31	0.50	0.38	10
repeat	1.00	0.70	0.82	10
reqalts	0.60	0.30	0.40	10
reqmore	1.00	0.60	0.75	10
request	0.80	0.80	0.80	10
restart	1.00	0.90	0.95	10
thankyou	0.86	0.60	0.71	10
Accuracy	**0.61**	-	-	**150**
Macro avg	**0.74**	**0.61**	**0.61**	**150**
Weighted avg	**0.74**	**0.61**	**0.61**	**150**

Table 17. SVM: Utterances with typos not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	1.00	0.20	0.33	10
bye	1.00	0.10	0.18	10
confirm	0.36	0.40	0.38	10
deny	0.00	0.00	0.00	10
hello	1.00	0.10	0.18	10
inform	0.08	0.40	0.13	10
negate	0.14	0.10	0.12	10
null	0.09	0.40	0.15	10
repeat	1.00	0.10	0.18	10
reqalts	0.00	0.00	0.00	0
reqmore	0.80	0.40	0.53	10
requalts	0.00	0.00	0.00	10
request	0.75	0.90	0.82	10
restart	1.00	0.20	0.33	10
thankyou	0.80	0.40	0.53	10
Accuracy	**0.25**	-	-	**150**
Macro avg	**0.50**	**0.23**	**0.24**	**150**
Weighted avg	**0.54**	**0.25**	**0.26**	**150**

Table 18. SVM: utterances with typos in data

Class	Precision	Recall	F1-score	Support
ack	1.00	0.30	0.46	10
affirm	1.00	0.10	0.18	10
bye	0.50	0.20	0.29	10
confirm	1.00	0.60	0.75	10
deny	0.00	0.00	0.00	10
hello	1.00	0.80	0.89	10
inform	0.14	1.00	0.24	10
negate	0.56	0.90	0.69	10
null	0.11	0.20	0.14	10
repeat	0.00	0.00	0.00	10
reqalts	0.60	0.30	0.40	10
reqmore	1.00	0.60	0.75	10
request	0.67	0.20	0.31	10
restart	1.00	0.20	0.33	10
thankyou	1.00	0.40	0.57	10
Accuracy	**0.39**	-	-	**150**
Macro avg	**0.64**	**0.39**	**0.40**	**150**
Weighted avg	**0.64**	**0.39**	**0.40**	**150**

Table 19. RandomForest: and utterances not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	0.57	0.40	0.47	10
bye	0.00	0.00	0.00	10
confirm	0.00	0.00	0.00	10
deny	0.00	0.00	0.00	10
hello	0.00	0.00	0.00	10
inform	0.50	0.30	0.37	10
negate	0.14	0.10	0.12	10
null	0.07	0.50	0.12	10
repeat	1.00	0.30	0.46	10
reqalts	0.00	0.00	0.00	0
reqmore	0.60	0.30	0.40	10
requalts	0.00	0.00	0.00	10
request	0.09	0.30	0.14	10
restart	1.00	0.30	0.46	10
thankyou	1.00	0.10	0.18	10
Accuracy	**0.21**	-	-	**150**
Macro avg	**0.45**	**0.20**	**0.23**	**150**
Weighted avg	**0.48**	**0.21**	**0.24**	**150**

Table 20. RandomForest: and utterances in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	1.00	0.70	0.82	10
bye	0.89	0.80	0.84	10
confirm	1.00	0.80	0.89	10
deny	1.00	0.90	0.95	10
hello	1.00	0.90	0.95	10
inform	0.60	0.90	0.72	10
negate	1.00	0.80	0.89	10
null	0.48	1.00	0.65	10
repeat	1.00	0.70	0.82	10
reqalts	0.82	0.90	0.86	10
reqmore	1.00	0.20	0.33	10
request	0.33	0.90	0.49	10
restart	1.00	0.70	0.82	10
thankyou	0.90	0.90	0.90	10
Accuracy	**0.76**	-	-	**150**
Macro avg	**0.87**	**0.76**	**0.75**	**150**
Weighted avg	**0.87**	**0.76**	**0.75**	**150**

Table 21. RandomForest: utterances with negation not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	0.15	0.20	0.17	10
bye	0.00	0.00	0.00	10
confirm	0.00	0.00	0.00	10
deny	0.00	0.00	0.00	10
hello	1.00	0.10	0.18	10
inform	0.28	0.90	0.43	10
negate	0.00	0.00	0.00	10
null	0.19	0.80	0.30	10
repeat	0.80	0.40	0.53	10
reqalts	0.00	0.00	0.00	0
reqmore	0.83	0.50	0.62	10
requalts	0.00	0.00	0.00	10
request	0.22	0.20	0.21	10
restart	1.00	0.50	0.67	10
thankyou	1.00	0.30	0.46	10
Accuracy	**0.15**	-	-	**150**
Macro avg	**0.32**	**0.14**	**0.14**	**150**
Weighted avg	**0.34**	**0.15**	**0.15**	**150**

Table 22. RandomForest: utterances with negation in data

Class	Precision	Recall	F1-score	Support
ack	0.75	0.30	0.43	10
affirm	0.71	0.50	0.59	10
bye	1.00	0.70	0.82	10
confirm	1.00	0.70	0.82	10
deny	1.00	0.10	0.18	10
hello	1.00	0.90	0.95	10
inform	0.37	1.00	0.54	10
negate	0.41	0.70	0.52	10
null	0.50	0.70	0.58	10
repeat	1.00	0.90	0.95	10
reqalts	0.73	0.80	0.76	10
reqmore	1.00	0.30	0.46	10
request	0.82	0.90	0.86	10
restart	1.00	1.00	1.00	10
thankyou	0.77	1.00	0.87	10
Accuracy	**0.61**	-	-	**150**
Macro avg	**0.74**	**0.61**	**0.61**	**150**
Weighted avg	**0.74**	**0.61**	**0.61**	**150**

Table 23. RandomForest: Utterances with typos not in data

Class	Precision	Recall	F1-score	Support
ack	0.00	0.00	0.00	10
affirm	1.00	0.20	0.33	10
bye	0.00	0.00	0.00	10
confirm	0.00	0.00	0.00	10
deny	0.00	0.00	0.00	10
hello	1.00	0.10	0.18	10
inform	0.15	0.60	0.24	10
negate	0.14	0.10	0.12	10
null	0.06	0.40	0.10	10
repeat	1.00	0.10	0.18	10
reqalts	0.00	0.00	0.00	0
reqmore	0.80	0.40	0.53	10
requalts	0.00	0.00	0.00	10
request	0.00	0.00	0.00	10
restart	1.00	0.30	0.46	10
thankyou	1.00	0.30	0.46	10
Accuracy	**0.25**	-	-	**150**
Macro avg	**0.50**	**0.23**	**0.24**	**150**
Weighted avg	**0.54**	**0.25**	**0.26**	**150**

Table 24. RandomForest: utterances with typos in data

Class	Precision	Recall	F1-score	Support
Class	Precision	Recall	F1-score	Support
ack	0.67	0.40	0.50	10
affirm	0.50	0.10	0.17	10
bye	0.40	0.20	0.27	10
confirm	1.00	0.60	0.75	10
deny	1.00	0.40	0.57	10
hello	1.00	0.50	0.67	10
inform	0.18	1.00	0.30	10
negate	0.60	0.60	0.60	10
null	0.21	0.50	0.29	10
repeat	0.00	0.00	0.00	10
reqalts	0.60	0.30	0.40	10
reqmore	1.00	0.60	0.75	10
request	0.71	0.50	0.59	10
restart	1.00	0.40	0.57	10
thankyou	0.67	0.60	0.63	10
Accuracy	**0.39**	-	-	**150**
Macro avg	**0.64**	**0.39**	**0.40**	**150**
Weighted avg	**0.64**	**0.39**	**0.40**	**150**

B DESCRIPTIN OF SYSTEM UTTERANCE TEMPLATES

The system always starts with the following utterance:
Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?

To ask for missing information:

- "What part of town do you have in mind?"
- "What kind of food would you like?"
- "Would you like something in the cheap, moderate, or expensive price range?"

To ask for additional requirements:

- "Are there any additional preferences you"d like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?"
- "Which additional preferences would you like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?"

To tell that there is no restaurant that meets the user's preferences:

- "Sorry, I couldn"t find a `price_string` `food_string` restaurant `area_string`. Please change your requirements. For example: Sorry, I couldn"t find a cheap italian restaurant in any area.

To sell a restaurant:

- "I recommend `state.currently_selected_restaurant.name`, `state.currently_selected_restaurant.name` in the `state.currently_selected_restaurant.area` area, serving `state.currently_selected_restaurant.food` cuisine, with `state.currently_selected_restaurant.pricerange prices." + second_part + Let me` know if you need the post code, address or phone number.
- Here `second_part` can consist of:
  - " This restaurant is child-friendly because you are served quickly. "
  - " It is a touristic hot-spot due to the cheap prices and the good food. "
  - " This is not a touristic place, since the food quality is very average. "
  - " The waiter will assign you a seat, since it is a very busy restaurant. "
  - " The restaurant is romantic because it allows you to stay for a long time. "
  - For example: I recommend the cambridge chop house in the centre area, serving british cuisine, with cheap prices. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.

To ask for confirmation of the user preferences:

- "Let me confirm, you are looking for `food_text` restaurant in `area_text` texttpricerange_text, right?"
  - Here `food_text` is: "article food"
  - Here `area_text` is: "any area" OR "the {area} of town"
  - Here `pricerange_text` is: "and you don"t care about the price range" OR "in the pricerange price range"
  - For example: Let me confirm, you are looking for an european restaurant in any area in the cheap price range, right?

To ask for confirmation of the additional requirements:

- "To clarify, you prefer a restaurant with these qualities: {", ".join(preferencesss)}."
- "Let me confirm, you have not specified any additional requirements?"
  - For example: To clarify, you prefer a restaurant with these qualities: Romantic, with assignedseats.

To ask user for clarification:

- "For the {key} preference, would you like `possible_values[0]` or `possible_values[1]`?"
  - For example: "For the food preference, would you like british or romanian?"

To fix ambiguity:

- "Your choice for {key} is now set to: {value}. "
  - o For example: You choice for food is now set to Lebanese.
- "I"m sorry, I failed to adjust your choice. "

To provide all contact information:

- "The restaurant restaurant.name is on restaurant.address with post code restaurant.postcode. This is their phone number: restaurant.phone"
  - For example: The address of the cambridge chop house is on 1 kings parade. The post code of the cambridge chop house is nan. The phone number of the cambridge chop house is 01223 359506.
- When the user utterance is classified as ack() and the system does not really know what to say back:
  - "How else can I help you?"

When the user affirms one of the system's confirmations and you have to give them feedback about what happened:

- It consists of 2 parts. This is the feedback that the preference was set:
  - "Okay. The area is now set to `state.user_preferences["area"]`. "
  - "Okay. The price range is now set to `state.user_preferences["pricerange"]`."
  - "Okay. Your food preference is now set to `state.user_preferences["food"]`."
- The next part comes from `ask_for_missing_info1`, or `ask_for_confirmation1`
  - For example: "Okay. The area is now set to south. What kind of food would you like? "
  - For example: "Okay. The price range is now set to expensive. "Let me confirm, you are looking for an italian restaurant in the south of town in the expensive price range, right?

If a user utterance is classified as affirm(), but the system was in the state "InformThatThereIsNoRestaurant":

- "Please provide your new preferences."

When the system ends the dialog:

- "Goodbye! Enjoy your meal!"
- "You"re welcome. Good bye."

When the user said thanks:

- "You"re welcome."

When the user wants to confirm some information:

- "restaurant.name is a nice place in the restaurant.area of town."
  - For example: the cambridge chop house is a nice place in the north of town.
- "restaurant.name is a nice place serving restaurant.food food."
  - For example: the cambridge chop house is a nice place serving british food.

- "restaurant.name is a nice place in the restaurant.pricerange price range."
  - For example: the cambridge chop house is a nice in the expensive price range.

When the user denies a preference:

- "Okay. What is your preference for key?"
  - For example: Okay. What is your preference for food?
- And when it fails: "Let us try again. Could you provide me more information about your preferred area, price range and food type?"

When the user has said nothing useful in the Welcome state:

- "Could you provide me more information about your preferred area, price range and food type?"

When the user negated the first confirmation:

- "I"m sorry. What is your preference for food, price range and area?"

When the system has detected a typo:

- "Did you say you are looking for a restaurant in the word of town?"
  - For example: Did you say you are looking for a restaurant in the south of town?
- "Did you say you are looking for a restaurant in the word price range?"
  - For example: Did you say you are looking for a restaurant in the cheap price range?
- "Did you say you are looking for a restaurant that serves word food?"
  - For example: Did you say you are looking for a restaurant that serves italian food?
- If the system doesn't understand it either:
  - "I"m sorry, I did not get that. Please give your preferences for area, price range and food type. "
  - "I am sorry. Could you please rephrase that?"

When the user wants other restaurant recommendations, but there is no other restaurant:

- "Sorry, I couldn"t find another restaurant that matches your preferences. Can you change your requirements?"
- "The restaurant `state.currently_selected_restaurant.name` is the only restaurant that meets your preferences."
  - For example: The restaurant the cambridge chop house is the only restaurant that meets your preferences."

When the user wants to restart the dialog:

- "Okay. We start over. Welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?"
- "I"m sorry. You are not allowed to restart the dialog."

C DESCRIPTIN OF STATES

Welcome (1) to AskForMissingInformation1 (2)

Explanation: In the example below, the system moves from Welcome (1) to AskForMissingInformation (2) as the user only gave the price range and the food type preference. Therefore, the system asks which part of town the user had in mind.

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: Can I have cheap european food please?
System: What part of town do you have in mind?
Me: any area is fine
System: Let me confirm, you are looking for an european restaurant in any area in the cheap price range, right?
Me: yes

The system never stays in the Welcome state (1), because it will always try to find the preferences, by either confirming, clarifying, or asking for it. So in the example below, where the user responds with 'hi!', the system moves to the AskForMissingInformation state (2) as well. System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type.

How may I help you? Me: hi! System: Could you provide me more information about your preferred area, price range and food type?

Welcome (1) to AskForClarification (3)

Explanation: In the following example, the system moves directly to the AskForClarification state (3), because the user made a typo and the user input could not be classified correctly. When the user input is classified with our Random Forest classifier, it classifies it as null and in here we perform the Levenshtein algorithm in order to find similar words. If a similar word is found, we get the following dialog:

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: chinees
System: Did you say you are looking for a restaurant that serves chinese food?
Me: yes
System: Okay. Your food preference is now set to chinese. What part of town do you have in mind?

If no match is found with the Levenstein algorithm, the dialog goes as follows:

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you? Me: chineeeeeeees System: I"m sorry, I did not get that. Please give your preferences for area, price range and food type.

The system stays then in the AskForClarification state (3) until the user starts informing the system.

Welcome (1) to AskForConfirmation1 (4)

Explanation: In the case that the user provides all preferences at once, the system will go form the Welcome (1) state, directly into the AskForConfirmation1 state (4) like in this dialog:

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: I would like to eat in an european restaurant that is expensive. Any area is fine.
System: Let me confirm, you are looking for an european restaurant in any area in the expensive price range, right?
Me: yes
System: Are there any additional preferences you"d like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?

AskForConfirmation1 (4) to AskForMissingInfo (2)

Explanation: In the case where the user does not confirm the preferences, the system removes all preferences and the user has to provide them all again.

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: Hi! Can I have an expensive british restaurant in any area?
System: Let me confirm, you are looking for a british restaurant in any area in the expensive price range, right?
Me: no
System: I"m sorry. What is your preference for food, price range and area?
Me: Uhh i want cheap british food
System: What part of town do you have in mind?
Me: centre
System: Let me confirm, you are looking for a british restaurant in the centre of town in the cheap price range, right?
Me: yes

AskForConfirmation1 (4) to AskForClarification (3)

Explanation:When the system asked for a confirmation and the user provides another preference, the system will go into the AskForClarification state (3), where it asks which one of the given preferences the user actually wants.

System: Hello, welcome to the UU restaurant system! You can ask for restaurants by area, price range or food type. How may I help you?
Me: I want expensive libanese food in any area
System: Let me confirm, you are looking for any restaurant in any area in the expensive price range, right?
Me: in the centre please
System: For the area preference, would you like any or centre?
Me: centre
System: Let me confirm, you are looking for any restaurant in the centre of town in the expensive price range, right?
Me: yes

AskForAdditionalRequirements (11) to GiveRestaurantRecommendation (6)

Explanation: As we have already seen in some of the dialogs above, when the system is in the AskForConfirmation1 state (4) and the user confirms the preferences and the, the system will move to the AskForAdditionalRequirements state (11).

System: Are there any additional preferences you"d like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: romantic
System: To clarify, you prefer a restaurant with these qualities: Romantic.
Me: yes
System: I recommend charlie chan in the centre area, serving chinese cuisine, with cheap prices. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.

AskForAdditionalRequirements (11) to InformThatThereIsNoRestaurant (5)

Explanation: When there is no restaurant that meets the user preferences and the additional requirements, the system moves to the InformThatThereIsNoRestaurant state (6) where a suitable system utterance is printed. In this case, the user has to start over with the preferences. The dialog will then continue starting from the user action named Express preferences.

System: Let me confirm, you are looking for a lebanese restaurant in the centre of town in the expensive price range, right?
Me: yes
System: Are there any additional preferences you"d like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?
Me: reservation
System: To clarify, you prefer a restaurant with these qualities: with assignedseats.
Me: yes
System: Sorry, I couldn"t find an expensive lebanese restaurant in the centre. Please change your requirements.

GiveRestaurantRecommendation (6) stays in GiveRestaurantRecommendation (6)

Explanation:It can happen that the user doesn't want the first recommendation. In that case, the system will provide another restaurant that meets the requirements. The user can keep asking for other options until the system has no more restaurants left that meet the preferences and the requirements. It will then move to InformThatThereIsNoRestaurant state (5), which we will see in the next section.

System: Let me confirm, you have not specified any additional requirements?
Me: yes

System: I recommend the missing sock in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: Is there anything else?
System: I recommend rice house in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: anything else?
System: I recommend the lucky star in the south area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: anything else?
System: I recommend charlie chan in the centre area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.

GiveRestaurantRecommendation (6) moves to InformThatThereIsNoRestaurant (5) to GiveRestaurantRecommendation (6)

Explanation: When the system does not have any restaurant recommendations left anymore, it tells the user that there are no restaurants that meet the requirements. It is possible that the user just wants to change one thing. In that case, the user utterance starts with "how about …'. In this case, the food preference is then changed from Chinese to British.

System: I recommend charlie chan in the centre area, serving chinese cuisine, with cheap prices. Let me know if you need the post code, address or phone number.
Me: anything else?
System: Sorry, I couldn"t find another restaurant that matches your preferences. Can you change your requirements?
Me: how about british?
System: I recommend the cambridge chop house in the centre area, serving british cuisine, with cheap prices. Let me know if you need the post code, address or phone number.

In case the user wants to specify several preferences over again, the user has to provide the additional requirements again as well:

System: Sorry, I couldn"t find another restaurant that matches your preferences. Can you change your requirements?
Me: british, cheap
System: Let me confirm, you are looking for a british restaurant in any area in the cheap price range, right?
Me: Yes
System: Which additional preferences would you like to specify such as romantic atmosphere, requiring a reservation, or being child-friendly?

GiveRestaurantRecommendation (6) moves to ProvideContactInformation (9)

Explanation: When the user has chosen a restaurant to eat, he can ask for the contact information. The system than provides this contact information. It is possible that the user has an additional question. The system will then answer this additional question.

System: I recommend the cambridge chop house in the centre area, serving british cuisine, with cheap prices. The restaurant is romantic because it allows you to stay for a long time. Let me know if you need the post code, address or phone number.
Me: Can i have the post code, address and phone number please?
System: The address of the cambridge chop house is on 1 kings parade. The post code of the cambridge chop house is nan. The phone number of the cambridge chop house is 01223 359506.
Me: can I have just the phone number please?
System: The phone number of the cambridge chop house is 01223 359506.
Me: does it serve chinese food
System: the cambridge chop house is a nice place serving british food.
Me: how about chinese
System: Sorry, I couldn"t find another restaurant that matches your preferences. Can you change your requirements?
Me: phone number
System: The phone number of the cambridge chop house is 01223 359506.

ProvideContactInformation (9) to End (10)

Explanation: When the user is fully informed, he will close the conversation by saying goodbye. The system will then say goodbye too. That is the end of the dialog."

System: I recommend eraina in the centre area, serving european cuisine, with expensive prices. Let me know if you need the post code, address or phone number.
Me: Thanks! Can I have the address and the phone number?
System: The address of eraina is on free school lane city centre. The phone number of eraina is 01223 368786.
Me: Thanks! Goodbye, system :)
System: You"re welcome. Good bye.