RESTAURANT FINDER

A Chatbot for Restaurant Recommendations

Development of a text-based restaurant recommendation system using a state-based dialog flow based on machine learning classification of user utterances.

DAAN, D.M, VAN DIJK

5070791 , Universiteit Utrecht, d.m.vandijk1@students.uu.nl

HARSHAVARDHANI SURESH KUMAR

189447, Universiteit Utrecht, h.sureshkumar@students.uu.nl

ECENAZ, E.R, rizeli

2822393, Universiteit Utrecht, e.rizeli@students.uu.nl

Lorenzo, L.P, pASQUALOTTO

9283943, Universiteit Utrecht, l.pasqualotto@students.uu.nl

ABSTRACT

The project aims at developing a text-based dialog system for restaurant recommendations. We implemented a state-based dialog management system exploring different Machine Learning techniques. The system processes user utterances to determine various dialog acts, which result in a structured interaction system and suggests suitable restaurants from a predefined database. First, we developed two baseline models: a majority class baseline and a rule-based keyword matcher. In addition to this, we trained three machine learning classifiers using support vectoral machines, logistic regression and a neural network. After evaluating the three approaches, we chose the best performing one (neural network) to be the state transition function of our dialog system. Furthermore, a reasoning component is added with inference rules to limit restaurant suggestions based on additional user preferences.

Our results demonstrate how a solid dialog flow design and simple machine learning techniques can be used for a limited context chatbot. Future improvements could include a more dynamic approach and expanded reasoning mechanisms.

1 INTRODUCTION

The research into natural language processing has increased significantly in recent years due to the growing usage of dialogs system in everyday life. Natural language processing encompasses both natural language understanding and natural language generation. This process involves both understanding user utterances and generating new utterances that can be understood in natural language. Dialog systems must be able to understand the meaning of the user input in order to give an appropriate reply. This conversation should mimic a human dialog. Dialog systems belong to goal-oriented systems in which a specific goal must be achieved at the end of the dialog. An example of goal-oriented systems is the restaurant recommendation system [1].

In this project a dialog management system for restaurant recommendation is developed. The system is designed to provide restaurant suggestions based on the user preferences. A dataset of restaurants was used consisting of attributes such as area, food type, price range. Three machine learning classifiers were trained using existing dialogs in order to classify new utterances as dialog acts such as greetings and requests. User preferences were extracted using Levenshtein edit distance and keyword matching. The system retrieved suitable restaurants from the database that matched the preferences.

This report is structured as follows: Section 2 describes the data including the dialogs and the state transition diagram. Section 3 discusses data preprocessing, machine learning and evaluation. Section 4 details the dialog manager and Section 5 explains the reasoning and the handling of inconsistencies. Section 6 presents configurability features, followed by the conclusion in Section 7. Finally, in Section 8 an overview of the team contributions will be given.

2 DATA

2.1 Data Description

The dataset consists of 3235 dialogs about recommending restaurants in Cambridge. Each dialog represents an interaction between the user and the system. The user communicates with the system to find a suitable restaurant based on their preferences in food type, price range and area. After the suggestion of a restaurant by the systems, users can request information about the recommended restaurant such as the phone number, postal code or the address. Each dialog consists of a task that needs to be performed by the user. The task specifies the conditions that the recommended restaurant must meet or the information that must be requested. Further, the dataset contains 19.480 number of duplicate utterances. The dialogs were collected using automated speech recognition, which resulted in low quality transcripts with incomplete sentences, repetitions within the same sentences, incorrectly recognized words and informal language such as abbreviations. This language usage affected the performance of the system, demanding iteration of system utterances, and making it difficult to deal with vagueness and replying efficiently.

2.2 Dialog Format

All dialogs have the same structure consisting of the following components:

- **Session ID:** Each dialog starts with a specific identifier such as voip-9f989824fd-20130324_075152
- **Task**: A particular goal that the user should achieve during the interaction with the system
- **Turn Index:** Each dialog has a turn index that starts at 0 and can go up to 29. The index monitors the alternation of turns between the system and the user
- **System turn**: The reply of the system
- **User turn:** The input of the user
- **Speech act**: Each user input is classified as a dialog act that represents a specific purpose of the user. There are 15 dialog acts in total which are shown in Table A.1 in Appendix A.

2.3 State Transition Diagram

The dialog flow between the system and the user is presented using a state transition diagram. The system generates responses according to the user's input and the current dialog state. The states shown as rectangles in the diagram represent a specific purpose, such as suggesting a restaurant. The dialog snippet shows the transition from state 3 to state 5. However, this can change because the current state depends on the user input. So, if the user specifies the area and the food type, then the next state will be state 4 asking the price range.

Example:

System:	Hello! welcome to restaurant search engine how can I help you?	
User:	I want a spanish restaurant in the east	State 2,3
System:	In what price range are you looking?	State 4
User:	Expensive	
System:	So you are looking for a restaurant in *east* with spanish food in the price range expensive right?	State 5
User: yes		

The user utterances were implicitly implemented in a state transition diagram. This ensured that the diagram remained clear without the user's statement being explicitly repeated each time. The first 50 dialogs of all dialogs were analyzed to find dialog states and transitions. The examination showed that the three most common transitions were request area, request food type and request price range. In some dialogs, the system asked for confirmation. This meant that the choices already made by the user, such as area, food type and price range were repeated until they were confirmed by the user. Then the system suggested a restaurant. The user may request an alternative to the recommended restaurant. Subsequently, the user could request information about the restaurant, such as the phone number, postal code or address. Finally, the system ends with "Enjoy your meal." Figure 1 shows the first version of the diagram. This diagram has been expanded with two new statuses: suggest suitable restaurants and ask for additional requirements. This has led to changes in the order of statuses. The system now recommends multiple restaurants and filters for one restaurant that meets the additional requirements. The final diagram is shown in Figure 2.

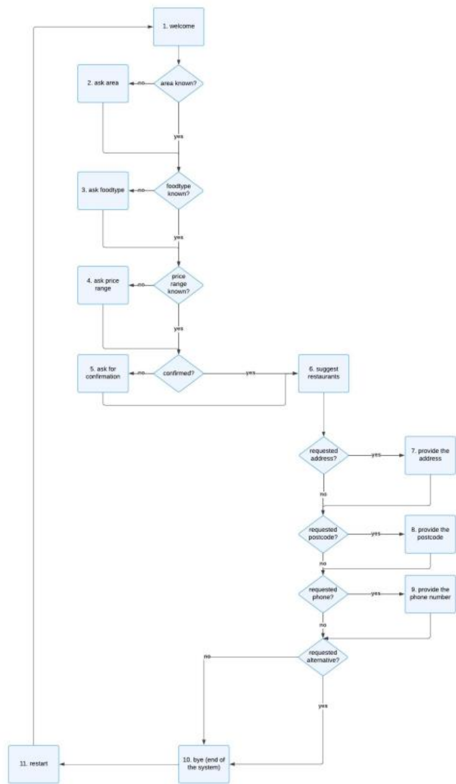

Figure 1: Initial State Transition Diagram

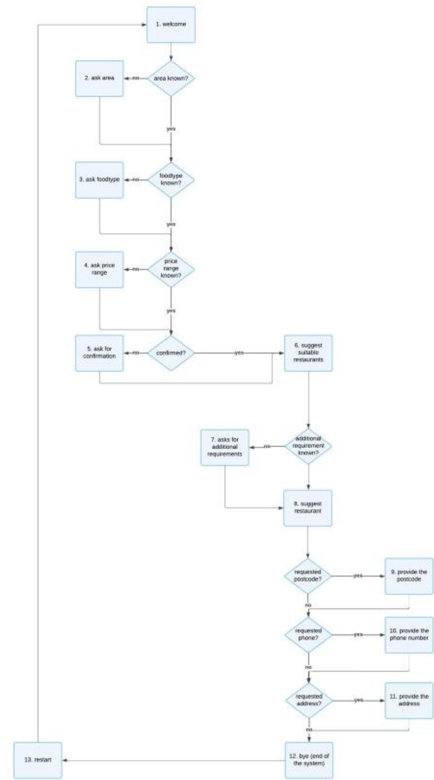

Figure 2: Final State Transition Diagram

3 MACHINE LEARNING

3.1 Data preprocessing

We trained five models in total. Two of these were simple baselines: A Most Frequent Class model, and a Rule-Based model. These gave reference points to compare against. On top of that, we built three machine learning models: a support vector machine (SVM), logistic regression and a multilayer perceptron (MLP), which allowed us to test more advanced approaches. Before training, we needed to get the data into the right shape. We loaded everything into a Pandas DataFrame with two columns: dialog_act and utterance. The dialog_act is the label for each utterance and utterance is the text itself. To make things consistent we converted all the text to lowercase. Next, we split the dataset into 85% training data and 15% test data. This gave us enough material to train the models. Finally, we saved both the training and test sets so we could reuse them whenever we needed, without having to repeat the preprocessing steps.

3.2 Baseline systems

After pre-processing the data, we moved on to training the baseline models. For both baselines, we started by loading the training data. The first baseline was the Most Frequent Class model. To build it, we looked for the most frequent label in the training data and checked what percentage of the dataset the most frequent term had. This gave a simple but useful reference point. Because if a model cannot perform better than guessing the most frequent term, then it is not useable. The second baseline was the Rule-Based model. For this approach, we constructed a dictionary for predicting the dialog act. Each dialog act maps to a set of keywords. If a keyword is in the utterances then that corresponding dialog act is returned. If no keywords are detected, the utterance is labelled as not_covered. This provides a simple but even more accurate baseline to classify utterances.

3.3 Machine learning model

For the machine learning classifiers, we first added a preprocessing step to handle missing values in the dataset. We created input features X and target labels Y for both training and testing data. To convert the text data into numerical form suitable for machine learning, we used count vectorizer, which transforms utterances into feature vectors based on word counts. Then we trained the three different machine learning classifiers and selected the one with the highest accuracy for further development. We also removed duplicate utterances and tested if that would increase accuracy for each of the three classifiers. The results showed that the MLP classifier achieved the highest accuracy, even with duplicates included. The MLP is a feedforward neural network for supervised classification tasks, using the standard ReLu activation function, which enables the classifier to learn complex, non-linear patterns in the data.

3.4 Quantative evaluation

We chose to evaluate the models using a classification report that shows the precision, recall, F1 scores, and accuracy for the classes of each model. It was already clear that the inform class was the dominant class in each model. This meant that we were dealing with an unbalanced dataset. Because of this we looked at the accuracy and weighted average of the F1 scores to assess how the classifiers perform relative to each other.

Model	**Accuracy**	**F1-score**
Baseline 1	0.39	0.22
Baseline 2	0.80	0.80
Logistic regression (non-depuplicated)	0.987	0.99
Logistic regression (deduplicated)	0.95	0.94
SVM (non-deduplicated)	0.986	0.99
SVM (deduplicated)	0.95	0.94
MLP (non-deduplicated)	0.99	0.99
MLP (deduplicated)	0.97	0.97

Table 1: Evaluation

Observations

Baseline 1 shows a poor performance as it gives the lowest scores in the classification report. Baseline 2 shows a better performance than Baseline 1, when looking at the accuracy and F1-scores shown in Table 1. The machine learning classifiers perform roughly the same with the non-duplicated data, when we look at the accuracy and weighted average of F1 scores. The MLP classifier performs slightly better with a very small difference. This does not change when we remove duplicates from the dataset; we only observe that the accuracy and weighted average of F1 scores decrease slightly, with MLP still performing best.

3.5 Error analysis

We analysed the misclassifications of each classifier to identify which utterances and dialog acts were more difficult to classify, and to look at which potential improvements in preprocessing we could do.

3.5.1Most frequent Misclassified Dialog Acts

Across all the machine learning classifiers, certain dialog acts were consistently more difficult to predict. These are shown in Table 2.

Logistic regression		**SVM**		**MLP**		**Rule-Based**	
Dialog act	*Count*	*Dialog act*	*Count*	*Dialog act*	*Count*	*Dialog act*	*Count*
inform	27	inform	20	inform	19	inform	546
request	12	request	15	negate	4	inform	243
affirm	9	affirm	13	thankyou	4	inform	137

Table 2: Error analysis

Observations

The inform dialog act is the most frequently misclassified across all models, likely due to its high frequency and semantic variability. Requests (requests, reqalts) are also common sources of error, particularly for Logistic Regression and SVM. The rule-based approach struggles significantly more than the machine learning classifiers. Correcting spelling errors or normalizing text could reduce misclassifications in the preprocessing part.

Some utterances were misclassified by all models, highlighting ambiguity or insufficient context:

- "thank you goodbye"
- "phone number"
- "moderately priced restaurant"

There were 18 such utterances in the dataset. These often involve:

- Short or incomplete sentences
- Ambiguous or multiple intents
- Negations
- Spellings variations

3.6 Difficult instances

3.6.1Negations

The first difficult instance focuses on the presence of a negation which can be classified as negate by some systems while others classified it as a inform label. The results are shown in Table 3.

Test instance	**Logistic regression**	**SVM**	**MLP**
No, I do not want Turkish food, I want Italian food	Negate	Inform	Negate

No I do not want a cheap restaurant, I want an restaurant	expensive	Negate	Inform	Negate

Table 3: Difficult instances - Negations

3.6.2Spelling errors

The second difficult instance is about spelling errors. Minor spelling errors cause incorrect classification as shown in Table 4, where all three models label these words as inform when they belong under hello, thankyou and bye.

Test instance	Logistic regression	SVM	MLP
Thankyoi	inform	inform	inform
by	inform	inform	inform
hllo	inform	inform	Inform

Table 4: Difficult instances – Spelling errors

4 DIALOG MANAGER

We Implemented the state diagram function based on the state transaction diagram shown in Table A.2. To predict the next state, we used the best-performing machine learning classifier (MLP model). The state transition defines how the dialog system moves from one state to another. It includes several parameters: current state, user input, preferences, output capitalization, and delay. The function returns the next state and the message that the system should display. The possible next states are listed in Table A.3. Below are some examples of the paths that the system can take:

1. User: "I'm looking for a restaurant"
   States: start → introduction → ask area

2. System: "In what area are you looking for a restaurant?"
   User: "west"
   States: ask area → ask food type

3. System: "For what food type are you looking?"
   User: "Italian"
   States: ask food type → ask price

4. System: "In what price range are you looking?"
   User: "expensive"
   States: ask price → confirmation

4.1 Additional features

We also implemented additional features for a better user experience. At the beginning of each dialog, the user is asked to configurate their settings. These settings include whether they want the system to restart after the dialog ends, whether all messages should be displayed in uppercase, and whether a short delay (1 second) should be added between each output message. Users can modify these configurations at any time, as illustrated in the example diagram (Table A.3). They can also specify 'any' as a preference, allowing the dialog manager to consider all elements within the domain. Additionally, the user can change their preferences at any point during the interaction. After the configurations have been set, the user can also define all three preferences at once. For example: "I'm looking for a cheap restaurant in the north with Korean food". The dialog manager will correctly interpret and process this request.

5 REASONING

After the system finds suitable restaurants, the user can proceed to enter an additional preference:
Does he prefer the restaurant to be **touristic**, **romantic**, suitable for **children** or one where **assigned seats** can be reserved? If the system finds at least one restaurant which possess the additional specified characteristic, it will recommend it to the user, otherwise it will ask to specify another one.

Example:

User preferences: area: south, food type: chinese, price: cheap
Suitable restaurants: the missing sock, rice house, the lucky star

System: Do you have any additional requirement? State 7
Type 1 for the touristic restaurants
Type 2 for the romantic ones
Type 3 if you would like them to be for children
Type 4 if you want those who provide assigned seats
User: 1
System: We found this restaurant for you: State 8
===
rice house
===

These additional properties are inferred from the original attributes of the restaurant, together with three others: **food quality** (good or bad), **crowdedness** (busy or not_busy), and **length of stay** (long or short). While the values of these three properties are randomly assigned, the ones proposed to the user are determined using the following inference rules shown in Table 5.

id	*antecedent*	*consequent*	*t/f*	*description*
1	cheap AND good food	touristic	True	a cheap restaurant with good food attracts tourists
2	romanian	touristic	False	Romanian cuisine is unknown for most tourists and they prefer familiar food
3	busy	assigned seats	True	in a busy restaurant the waiter decides where you sit

4	long stay	children	False	spending a long time in a restaurant is not advised when taking children
5	busy	romantic	False	a busy restaurant is not romantic
6	long stay	romantic	True	spending a long time in a restaurant is romantic

Table 5: Inference rules

5.1 Inference Handling

As shown in Table 6, certain combinations of rules (e.g., rules 1 and 2, or rules 5 and 6) may lead to inconsistencies. For instance, a restaurant could be simultaneously Romanian, cheap, and good, or both busy and associated with a long stay. In such cases, the inferred properties are not uniquely determined.

To address this issue, we check the restaurants obtained in State 6 and run the rules to infer the additional requirement specified by the user in State 7. If a restaurant is found to satisfy the user's requirement, it is presented to the user. If the application of the rules results in an inconsistency, the restaurant is flagged. If all restaurants lead to inconsistencies, the user is notified and prompted to revise the requirement. Finally, if no restaurant satisfies the specified property, the user is coherently informed and asked to provide a new requirement.

Example:

User preferences: area: south, food type: chinese, price: cheap
Suitable restaurants: the gardenia
Other properties: crowdness: busy, length of stay: long
Additional requirement: touristic

System: The additional requirement cannot be determined for the selected restaurants, please specify another one.

6 CONFIGURABILITY

The dialog system has the following five configurability options:

- **Restarts:** This feature ensures that the dialog system restarts from the beginning
- **CAP:** This feature ensures that the system utterances are expressed in capital letters
- **Delayed Response:** This feature causes the system utterances to be delayed
- **Modifications of all modifications:** This feature allows the user to modify the options: restart, all letters in caps lock and delayed response during the dialog
- **Modifications of the preferences:** This feature allows the user to modify their preferences

At the beginning of each dialog, these options are presented to the user. This gives the user the option to enable or disable these features. In the following subsections, the features will be explained in detail with regard to their implementations.

6.1 Restart, Capital Letters and Delayed Response

The set configurability function contains the variables restart, caps lock and delay which are all set to False. However, the user could turn these variables to True, by answering the questions related to these options with yes or by confirming them positively. The following question are related to the variables: restart, caps lock and delay:

(1) Do you want to allow restarts?
(2) Do you want a small delay in the response?
(3) Do you want the output in CAP?

The user input on these questions is processed by the utterance classifier which determines if the user utterance belongs to affirm or not. If it belongs to affirm then the variables restart, caps lock and delay are set to True. In addition to allow restarts at the beginning of the dialog, the user can also restart during the conversation by pressing 'r' or 'R'. Furthermore, for a delayed response, the time module and the time sleep function were used, causing system responses to be displayed one second later.

6.1 Modification of the modifications

Before the dialog starts, the user is informed about the possibility of modifying the choices concerning restart, all text in caps lock and delayed response. This allows the user to re-enter their choices during the interaction with the system. This is implemented in the main agent loop by equating the user input to the letter 'c' or 'C'. Once the user presses the letter 'c', the set configuration function is called to update the restart, caps lock and delay options. The dialog is paused and the questions concerning restart,caps lock and delay option are asked again. Once the questions have been answered, the conversation continues with the modified options.

6.2 Modification of the preferences

The option to modify the preferences has been implemented in a similar way. The user is informed prior to the dialog about how the preferences could be adjusted. In this case, the user input is set equal to the letter 'm' or 'M', which ensures that the function modify preferences is called. This function makes it possible to change the area, food type or the price range. First, the user has to enter the preference that will be modified and then enter the new input for that preference. Once the user has modified a preference, the interaction continues.

7 CONCLUSION

This project successfully developed a dialog system for restaurant recommendations, combining a rule-based dialog manager with a machine learning classifier for understanding user intents. The system effectively guided conversations through a predefined flow, collected user preferences, and provided personalized suggestions.

7.1 Dialog System Considerations

A major limitation of the system is its dependency on the dataset's structure and content. The dataset's restricted size and scope often resulted in situations where no restaurants matched the user's specified characteristics, leading to dialog dead-ends. Furthermore, the strict, predefined state transition diagram lacks flexibility to handle highly complex or unconventional user interactions that deviate from the expected flow. To address the empty result problem, the system could be improved by providing more proactive feedback. For instance, after each user-specified characteristic, the system could output the number of restaurants that match the criteria so far. This would manage user expectations and allow them to broaden their preferences early in the dialog. Otherwise, instead of relying on a small dataset, the system can integrate web scraping techniques.

7.2 Dialog Acts Classification Considerations

The experimental methodology for dialog act classification was systematic, progressing from simple baselines to more complex machine learning models. All machine learning models significantly outperformed the baselines, with the MLP classifier achieving the highest accuracy (0.99) on the non-deduplicated data. This demonstrates the superiority of statistical models in capturing the complex patterns in natural language utterances over simple keyword matching. The primary limitation for the classification task was the size and quality of the dataset. The class distribution was highly imbalanced, with the inform class dominating, which posed a challenge for the classifiers. Furthermore, the presence of spelling errors, informal language, and ambiguous short utterances (e.g., "phone number") created difficult instances that were consistently misclassified across all models. Improvements could focus on the research for a quality dataset or on advanced text preprocessing. To handle class imbalance, techniques such as SMOTE (Synthetic Minority Over-sampling Technique) or using class weights during model training could be employed. Incorporating contextual information from previous dialog turns, rather than classifying utterances in isolation, could also resolve ambiguities and improve accuracy. As an alternative approach, one could explore transformer-based models like BERT, which are pre-trained on vast amounts of text and are highly effective for classification tasks. However, given the small size of the dataset, fine-tuning a large model like BERT might lead to overfitting without careful regularization, making the simpler MLP a robust and efficient choice for this specific project context.

8 OVERVIEW

The following table shows how each member of the group contributed to the project.

TASK	DAAN	HARSHA	LORENZO	ECENAZ
Baseline – Majority class			0.5	
Baseline – Rule Based			0.5	
Baseline – Machine Learning	2			
Baseline - Evaluation			1	1.5
Dialog Flow Design				2
Dialog Flow Implementation	3			1
Lookup function			2	
Other utilities functions			1	
Inconsistency Handling				1
Configurability	2			
Report - Abstract		1		
Report - Introduction				0.5
Report - Data				1
Report - Machine Learning	1			

Report - Dialog Manager	1			
Report - Reasoning				1
Report - Configurability				1
Report - Conclusion			0.5	
TOTAL HOURS DEDICATED	9	1	9	9

Table 6: Contribution Table

REFERENCES

[1] A. Algherairy and M. Ahmed. 2024. A review of dialogue systems: current trends and future directions. Neural Computing and Applications 36, 12 (2024), 6325–6351. https://doi.org/10.1007/s00521-023-09322-1

A APPENDICES

A.1 Table of Dialog Acts

Dialog act	*Description*	*Example sentence*
ack	acknowledgement	okay um
affirm	positive confirmation	yes right
bye	greeting at dialog end	see you goodbye
confirm	double check given information	is it in the center of the town

deny	reject system suggestion	I don't want vietnamese food
hello	greeting at dialog start	hi I want a restaurant
inform	state a preference	I'm looking for a restaurant that serves seafood
negate	negation	no in any area
null	noise or not meaningful	cough
repeat	ask for repetition	can you repeat that
reqalts	request alternative suggestion	how about korean food
reqmore	request more suggestions	more
request	ask for information	what is the post code
restart	ask to restart the dialog	okay start over
thankyou	express thanks	thank you goodbye

Table A.1: Dialog acts

State	*Diagram state*	*Message*
start	**welcome**	Hello! Welcome to restaurant search engine how can I help you?
ask_area	**Ask area**	In what area are you looking for a restaurant?
ask_foodtype	**Ask food type**	For what food type are you looking?
ask_price	**Ask price range**	In what price range are you looking?
confirmation	**Ask for confirmation**	Do you have any additional requirements? Type 1 for the touristic restaurants, Type 2 for the romantic ones, Type 3 for children, Type 4 for assigned seats.
ask_add	**Additional requirements**	Would you like any information about the restaurant? Type 1 for phone number, Type 2 for address, Type 3 for postcode, Type 4 for all of them.
provide_info	**Provide info (postcode, phone number etc)**	Do you want any other information?
end	**End of system**	Thank you for using our services, goodbye!

Table A.2: Dialog states and Diagram states

State	*Next state(s)*
start	introduction
introduction	ask_area, ask_foodtype, ask_price, introduction
ask_area	ask_foodtype, ask_price, confirmation
ask_foodtype	ask_area, ask_price, confirmation

ask_price	ask_area, ask_foodtype, confirmation
confirmation	ask_add, ask_foodtype, confirmation
ask_add	provide_info, ask_add, end
provide_info	provide_info, end
end	start, —

Table A.3: State Transition table

==

CONFIGURE YOUR PREFERENCES

==

Do you want to allow restarts?yes
Do you want a small delay in the response?yes
Do you want the output in CAP?yes
ANSWER 'C' IF YOU WANT TO CHANGE THE MODIFICATIONS, 'R' IF YOU WANT TO RESTART AND 'M' IF YOU WANT TO MODIFY YOUR PREFERENCES

==

DIALOG

==

HELLO! WELCOME TO RESTAURANT SEARCH ENGINE HOW CAN I HELP YOU? looking for a restaurant
IN WHAT AREA ARE YOU LOOKING FOR A RESTAURANT? c
Do you want to allow restarts?no
Do you want a small delay in the response?yes
Do you want the output in CAP?no
In what area are you looking for a restaurant? nort
For what food type are you looking? italian
In what price range are you looking? moderate
So you are looking for a restaurant in north with italian food in the price range moderate right? m
Which preference do you want to change (area / food type / price range)?area
Please enter the new area: east
area updated to: east
So you are looking for a restaurant in east with italian food in the price range moderate right? yes
We found these restaurants for you:

==

pizza hut fen ditton

==

Do you have any additional requirement?
Type 1 for the touristic restaurants
Type 2 for the romantic ones
Type 3 if you would like them to be for children
Type 4 if you want those who provide assigned seats

3
We could not find any restaurant with the specified requirement, please insert a new one.

Table A.3: Example dialog