

# Indecisive? Let Chatbot help you with your dinner plans!

AMY OEY, 7082770

KATHLEEN DE BOER, 6439608

SUNNY HSIEH, 6534856

ANDREA ROIJEN, 5726042

Group 25

## Abstract

The goal of this project was to create a restaurant recommendation chatbot that provides the user restaurant suggestions for their stated preferences. To do this, a state machine was created to coordinate through a conversation with the user. Different types of baseline and machine learning models are used to extract the dialog acts from the user's utterances. These dialog acts are used by the state machine system to move the conversation correctly through different states and recommend a restaurant at the end. The specified user preferences for location, price and food type are extracted using a keyword matching method. The chatbot also uses a reasoning component to check if restaurants satisfy the user's additional requirements. It is possible to use the chatbot in a formal setting or informal setting. Finally, the user will be asked to rate the chatbot at the end of each conversation. All models were trained on a set of labelled dialogue examples and evaluated on a labelled test set and by error analyses for real tested dialogues. The baseline model clearly performed less well than the machine learning models, that all had a similar accuracy of around 98 %. Although there are many possible improvements for the chatbot, it can be concluded from the evaluation of the system that a relatively small system can still result in a well-functioning chatbot that is able to recommend existing restaurants based on its conversation with another human and a database of restaurant and dialogue information.

**Keywords:** machine learning, natural language processing, chatbot, dialog acts, reasoning

## ACM Reference Format:

Amy Oey, Kathleen de Boer, Sunny Hsieh, and Andrea Roijen. 2021. Indecisive? Let Chatbot help you with your dinner plans!. 1, 1 (October 2021), 15 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

Some days you are just indecisive and you do not feel like doing the groceries, even though you are really hungry and desire an extensive meal after a hard day of work. You could call someone or stroll the internet endlessly to find the restaurant you are looking for. However, would it not be easier to just state your preferences to a chatbot that is always available and get a carefully selected recommendation in return?

---

Authors' addresses: Amy Oey, [a.oey@uu.students.nl](mailto:a.oey@uu.students.nl), 7082770; Kathleen de Boer, [k.k.deboer@students.uu.nl](mailto:k.k.deboer@students.uu.nl), 6439608; Sunny Hsieh, [s.hsieh@students.uu.nl](mailto:s.hsieh@students.uu.nl), 6534856; Andrea Roijen, [a.roijen@students.uu.nl](mailto:a.roijen@students.uu.nl), 5726042  
Group 25.

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2021 Association for Computing Machinery.

Manuscript submitted to ACM

Manuscript submitted to ACM

The goal of this project is to create a restaurant recommendation chatbot that can provide the user with appropriate recommendations given their stated preferences. For the implementation of such a chatbot, a state machine and language processing models are used to comprehend a user's preferences during a conversation. Based on this conversation, the chatbot will match a suitable restaurant to recommend to the user, based on a dataset of restaurant information.

The current paper reports the description and evaluation of different machine learning models that are used for the extraction of dialog acts within a conversation, as well as a dialog manager system that coordinates the conversation. Dialog acts can be seen as utterance classification labels that give information about the purpose of a user's utterance (for example, request or negate something). Based on the dialog acts and the content of the user's responses, the state machine can guide the dialog manager to returning appropriate responses (chosen from a fixed dictionary of responses).

In the following sections, a description is given for the data (section 2) and language models that are used to classify the dialog acts of user utterances (section 3). For these models, data pre-processing steps, accuracy evaluations and error analyses are described as well. Section 4 and section 5 describe the dialog manager system and the reasoning component of the chatbot. This will give insight into how different dialog acts and the content of responses are processed in different states of the machine, in order to reach the final state and return restaurant recommendation if this is possible for the given preferences of the user.

## 2 DATA

For the training of the models, a dataset called "dialog\_acts.dat" was used. This set contains 25501 English sentences. These are mostly short sentences that were annotated from 3235 dialogues in the restaurant domain from the second Dialog State Tracking Challenge (DSTC 2) [1].

Each dialogue in DSTC 2 represents an interaction between a user and a system. In these sessions the users were given different tasks. Based on these tasks, the user will start a conversation with the system by communicating the preferences as described in the task. The system responds with questions, communicative statements and at the end an appropriate recommendation. In Example 1 one example of the dialog format is given [1]. All user utterances are collected using automatic speech recognition, which limits the quality of the transcriptions. Secondly, people tend to write more formally than when they speak [3]. This may impact the system's performance when it will be used for more serious inquiries or written text.

The used set for the chatbot, "dialog\_acts.dat", only consists of user utterances from these dialogues and their dialog act label. The set was simplified such that whenever an utterance was labeled with two or more dialog acts, only the first dialog act was used. In table ?? a slice of example sentences from this dataset is shown. Many utterances are very short and informal, and may be similar to each other. Only a limited amount of complex syntactic structures can be found. However, this will not lead to problems for the chatbot system. Finally, sentences like "cough" (1) will never appear in written conversations, and are therefore not useful for the chatbot system.

We divided the data in a training set (85%) and a test set (15%) and we transformed every sentence to lower casing to avoid problems when comparing keys.

The second dataset that was used for the chatbot is restaurant\_info.csv. This is a relatively small set, containing 110 London restaurants entries. For every restaurant the name, price range, area, food type, phone number, address and postcode is specified. This dataset is limited in some perspectives. For some restaurants, information is missing, such as the restaurants' phone number, address and postcode. Restaurant areas are classified by either north, east, south, west or centre, which only refers to parts of London. Furthermore, the dataset is so little that there is a high chance of not

**Example 1: An example of a dialog from the DSTC 2 dataset**

session id: voip-cfd5fa34d9-20130402\_074254  
 Task 00891: You want to find and it should serve mexican food. You don't care about the price range. Make sure you get the address and phone number of the venue.  
 turn index: 0  
 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: mexican food  
 speech act: inform(food=mexican)  
 turn index: 1  
 system: chiquito restaurant bar serves mexican food  
 user: whats the address  
 speech act: request(addr)  
 turn index: 2  
 system: Sure , chiquito restaurant bar is on 2G Cambridge Leisure Park Cherry Hinton Road Cherry Hinton  
 user: whats the phone number  
 speech act: request(phone)  
 turn index: 3  
 system: The phone number of chiquito restaurant bar is 01223 400170 .  
 user: thank you good bye  
 speech act: thankyou()|bye()

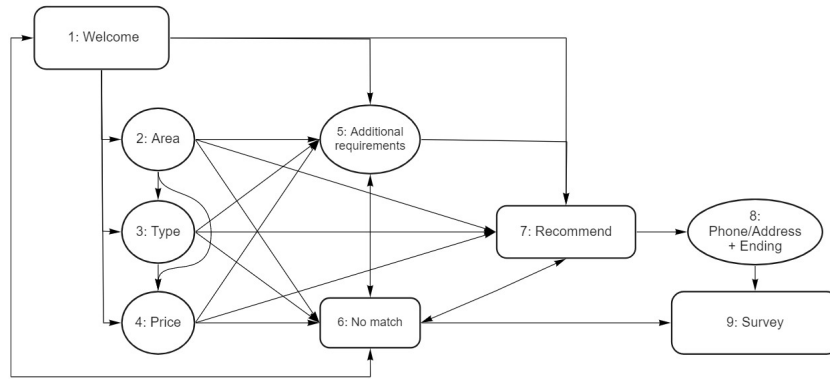
inform	im looking for a moderately priced restaurant that serves
inform	bistro food
confirm	is there a moderately priced restaurant that serves British food
affirm	yes
request	could i get their phone number
thankyou	thank you good bye
null	cough

Table 1. A sample of the sentences used to train and test the model

finding a restaurant that matches a user's preferences. To extend the capabilities of the chatbot system with a reasoning component, the dataset is extended for this project. The columns *quality*, *staytime* and *crowdedness* were added. For each restaurant a random Boolean value is indicated for each of these new columns.

**2.1 Dialog state transition diagram with example dialog snippet**

Below, the Dialog state transitioner and an example dialog snippet are shown. These will be further explained throughout this report.



miro

Fig. 1. State transition diagram

## Example 2: Example dialog snippet

system: Hey there!!! Welcome to the Group 25 restaurant system! You can ask for restaurants by area, price range, or food type. Let's go, enter your request! :)

Comment(won't be shown in the chatbot): Welcome state.

user: Hello I want an italian restaurant.

system: Hi, In which part of town do you want to dine???

Comment: Area unknown so going to state 2.

user: Let's do somewhere in the south.

system: What's your budget? Cheap, moderate or expensive???

Comment: Going from state 2 to state 4 because food type is already known.

user: cheap, because I'm not very rich.

system: Okay, I came up with a recommendation: la margherita. Sounds good?

Comment: There is only one option found by the system, so it won't ask for additional requirements, which means it goes to state 7 directly, recommend restaurant.

user: No, not really.

system: Sorry, there's no restaurant matching your tastes :( . Wanna try something else??? I'm now looking for cheap, italian restaurant in south region.

Comment: As mentioned above, there was only 1 match, so when the user is not satisfied in this case there won't be any other suggestions left which leads the system goes to state 6.

user: no thank you. system: I hope you liked our conversation, I sure did! How would you rate my help on a scale from 1 to 5? (1 being very bad, 5 being very good).

Comment: At the end of the conversation we will ask the user to rate the chatbot.

user: exit, I don't want to rate you.

Comment: Ending of the conversation.

### 3 MACHINE LEARNING

The models described in this section are used in this system to classify the dialog acts from the dataset "dialog\_acts.dat". This dataset consists of a simplified user system dialog, in which each utterance is labeled as an action. These 15 actions abbreviations are "ack", "affirm", "bye", "confirm", "deny", "hello", "inform", "negate", "null", "repeat", "reqalts", "reqmore", "request", "restart", "thankyou". The goal of the machine learning models is to give the correct dialog action label to a sentence. The classification of the utterances is used in the dialog system to follow the right dialog track through different states.

#### 3.1 Baseline Systems

To begin with, two baseline systems were created. These systems are relatively simple methods that do not use any machine learning. For the baseline systems, the only pre-processing steps that needed to be taken was the separation of the data set (of dialog acts) into a training and test set. For this, the data was split into a training to test ratio of 85:15.

The first baseline counts the total number of labels and counts for each label how often it occurs in the training set. For all labels, a probability distribution in the training set is calculated. The baseline always returns the same label, which is the one that has the highest probability (the one that occurs most frequently compared to all other labels). For this set, this was the label 'inform', which occurs in about 40 % of the set. This also means that the accuracy of the first baseline model on the test set was around 40%. No content of utterances are taken into account for this baseline.

The second baseline that was created for this system, is based on the use of keywords to recognize dialog acts. A keyword dictionary was manually created to match specific keywords to dialog act labels. These keyword decisions are based on the exploration of the training data. For each label, one or more keywords are added as value if they occurred frequently in sentences with this label. All decisions that are made, are based on whether the baseline's accuracy on the training set did improve by adding a certain keyword to the dictionary. After the creation of this dictionary, no further training was necessary. When using this second baseline, it can be fed a user utterance. The baseline was constructed such that it looks at each individual word of the utterance and returns the label from first match that it finds with a keyword from the dictionary. This is done because the simplification of the dialogue data was also based on only using the first dialogue act label if there were more than one labels for an utterance. If the baseline does not find a match, then it will always return the label 'inform'. This is because 'inform' is considered the default label, since it occurred most frequently according to the data exploration. The accuracy of the second baseline is around 90% on the test set.

#### 3.2 Design Machine Learning Model

For the main chatbot's system three different Machine Learning Models are implemented. The user can choose out of a Decision Tree, a Neural Network and a Logistic Regression model. To implement the models the sklearn python library is used.<sup>1</sup> The three used models will briefly be discussed: A decision tree is a predictive model with the structure of a tree in which the leaves represent class labels and branches represent features that lead to those class labels. The tree is trained by growing decision nodes using features that best split the dataset. Decision trees are relatively easy to understand. A logistic regression starts with an input layer where the input values are summed up the input is then passed to a layer with the sigmoid function. At last the model uses a decision threshold layer for classification. A Neural network is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. The hidden layers perform nonlinear transformations and can not be observed

<sup>1</sup><https://scikit-learn.org/stable/>

from the input and output. The neural network is trained using backpropagation, which alters the weights between the layers. All build models have a accuracy of 98%. This proves that all models are very precise in predicting the dialog actions. For time saving purposes the models are pre-trained for usage in the system (the training does not happen at each classification task).

### 3.3 Data Preprocessing

The raw input for the models consist of the examples  $x$  from the column "Act" and the corresponding labels  $y$  from the column "Sentence". Both the sentences  $x$  and the labels  $y$  are then encoded to contain only numbers. The preprocessing of the data is done with the sklearn packages "LabelEncoder" for the  $y$  values (labels) and "CountVectorizer" for the sentences. LabelEncoder is used to transform the non-numerical labels to numerical labels. CountVectorizer is used to tokenize the words and count the occurrences of the input words. It tokenizes the string by extracting words of at least 2 letters. Each token found by the analyzer during the fitting is assigned a unique integer index corresponding to a column in the resulting matrix which will be the vocabulary. Because CountVectorizer fits on the dataset, words not occurring in the dataset will be ignored. No stoplist was used for the preprocessing of our data since the sentences are short and thus training time is limited. The distribution of the labels from the data is as follows: inform = 0.398, request = 0.254, thanjyou = 0.127, reqalts = 0.069, null = 0.063, affirm = 0.045, negate = 0.017, bye = 0.017, confirm = 0.0067, hello = 0.003647, repeat = 0.0012, ack = 0.001, deny = 0.001, restart = 0.0005, reqmore = 0.0001. The label "inform" is the most occurring label at almost 40%. Next up is "request" occurring 25,5% of the time. This is not suprising since these particular dialog acts are of most importance for the system. The average amount of words in a utterance of the dataset is 3,73 words. The dataset contains a lot of utterances consisting of only one word, such as the food type for example "thai".

### 3.4 Parameters

The default parameters from sklearn resulted in the highest accuracy for all three models. It is because of this that these parameters are used in the system. For train-test splitting a ratio of 85:15 was used. The decision tree uses the gini impurity score for determining the best split on which it grows a decision node. No parameter such as max-depth or min-leaf is used. The neural network uses the ReLu activation function and the Adam stochastic gradient-based optimizer. The Logistic Regression model uses the lbfgs solver, which stands for "Limited-memory Broyden-Fletcher-Goldfarb-Shanno". This solver stores only the last few updates, so it saves memory.

### 3.5 Error analysis

For the first baseline, some problematic issues occur. All sentences that should be labelled with anything else than 'inform' (the most frequent label) will be classified incorrectly. The use of the first baseline has negative consequences for the dialog state transitions. Based on the speech acts of the utterances, the system moves through different states (as will be explained in the next section). However, this baseline always returns the same type of act. This leads to the system getting stuck towards the end of the conversation, where the class of 'inform' can no longer lead the system through the conversation. For example, when the system wants to know whether the user agrees with its suggestion it only understand an 'affirm' of 'negate'. However, responses like "yes" or "no" will be classified as 'inform'. Because of this, the system will never reach a next state as can be seen in Example 3.

## Example 3: An example of getting stuck while using the first baseline

```

user: i want chinese food for an expensive price
system: In what part of town do you want to dine???
speech act:(inform)
user: north
system: Okay, I came up with a recommendation: the hotpot. Sounds good?
speech act:(inform)
user: no
system: Let me know if the restaurant sounds good or not by responding "yes" or "no"!
speech act:(inform)
user: yes
system: Let me know if the restaurant sounds good or not by responding "yes" or "no"!
speech act:(inform)

```

Although the second baseline has a higher accuracy and overcomes the error of the first baseline, it is in practice far from perfect. Error analysis on the test set utterances revealed some classification errors that have no major consequences on the conversations. For example, incomplete sentences could not be labelled correctly with 'null' by the baseline. However, as will be explained, the dialog system will use other methods to detect whether certain information is incomplete or unclear. The baseline is constructed such that the first match between a word in the sentence will return the appropriate label. However, sometimes it would make more sense to return the label of the second keyword match. One example is the utterance *"thank you and good bye"*. In this case, the user would expect the system to respond to its dialog act 'bye'. However, the baseline only picks up on the first act 'thankyou'. This limitation could be overcome, if a less simplified dataset is used and the system can label more than one dialog act to a user utterance. A second issue is that the dictionary is only based on the language variation in the data set. Natural language use consists probably of even more language variation than captured by the current dictionary of keywords. Its accuracy of 90% is thus not promised for all possible conversations with the chatbot, only for the conversations in the test set. Although the keyword dictionary could be anticipated on more variation of the keywords, it is impossible to include all options. To overcome this limitation, the chatbot system could be extended with more instructions for the user. These instructions could inform the user of not using relatively *'usual'* words for the best results.

The machine learning models are trained on a simplified version of the dataset. As for when an utterance is labeled with two different dialog acts, only the first dialog act is used as a label. This could result in some state transition problems, for when a "wrong" action is assigned by one of the classifiers. For example if you ask the chatbot's system "May I have an Indian restaurant", this is classified as "Inform", while "Request" would also be correct. This does not cause any problems, since in this example the information is the most important and the request is fulfilled when the information is saved and then used to select a restaurant to the users requests. Some instances that our system has difficulty with are for example when the system saves the preference "world" when the user types "I would like some indian food". The same issue arises when the user types the word "eat", which the system saves as "east". Nevertheless these problems are not classifier related, since the classifier classifies "would", and "eat" as "null" and not as "inform" like "world" and "east".

## 4 DIALOG MANAGER

### 4.1 Final diagram

The final diagram is represented in the figure State Transition Diagram. We designed our chatbot in a way that multiple state transitioning are allowed to make it as flexible as possible. For example from state 1 you can transition to state 1, 2, 3, 4, 5, 6 and 7 (only 8 and 9 not). Also, we included the option for the user to choose "any", when there is a question like in which area do you want to eat. In the figure some arrows were not drawn because they were not special to mention and would make the figure look more complicated. However, keep in mind that it's always allowed to go to state 1 by reset, using repeat to stay in the same state and using exit to leave. How the state transition works will be explained in the next section. At last there were of course some limitations which will be discussed in the discussion .

### 4.2 State transition model

In the state transition table here below you can see which transitions are allowed. It might be not clear yet under which conditions one state can transition into another state, so that will be elaborated one by one.

State	1	2	3	4	5	6	7	8	9
1	O	O	O	O	O	O	O	X	X
2	X	O	O	O	O	O	O	X	X
3	X	X	O	O	O	O	O	X	X
4	X	X	X	X	O	O	O	X	X
5	X	X	X	X	O	O	O	X	X
6	O	X	X	X	O	O	O	X	O
7	O	X	X	X	X	O	O	O	X
8	X	X	X	X	X	X	X	X	O
9	X	X	X	X	X	X	X	X	X

Table 2. State transition table

"Exit" and "reset" are not included in this table. These two are considered special acts. If the act is "exit", you can always leave immediately no matter in which state you are. The same for reset, you can always reset and the system will start over again, starting in state 1. Besides, there's also a repeat act, such that a state will remain in the same state. These cases won't be included in the following description of the states, but as you can see every state can stay in the same state.

**4.2.1 State 1.** State 1 is the welcome state where the system greets the user and introduces itself. Depending on the utterance of the user, the system can move to different states.

- Transitioning to state 2 Area -When a user haven't expressed any preferences,
- A user expressed the food type or price. (inclusive disjunction)
- Greets back.
- Any other utterance that has nothing to do with filling the slots of the preferences.



Transitioning to state 3 Type -The user expressed the area. If the user expressed the area and price range, it will also go to state 3.

Transitioning to state 4 Price -The user expressed the area and food type.

Transitioning to state 5 Additional requirements -The user expressed the area, food type and price. And the system found more than 1 option.

Transitioning to state 6 No match -The user expressed the area, food type and price. But the system found no options.

Transitioning to state 7 Recommend -The user expressed the area, food type and price. And the system found exactly 1 option.

**4.2.2 State 2.** State 2 is the area state where the system asks for the area where the user would like to eat. Depending on the utterance of the user, the system can move to different states.

Transitioning to state 3 Type -When the type is still unknown

Transitioning to state 4 Price -The user expressed the area, and the food type was already known.

Transitioning to state 5 Additional requirements -The user expressed the area, but the food type and price were known. And the system found more than 1 option. It would be not efficient if the system found only 1 option and asks for more additional requirements.

Transitioning to state 6 No match -The user expressed the area, and the food type and price were known, but the system found no options.

Transitioning to state 7 Recommend -The user expressed the area and the food type and price were known. And the system found exactly 1 option. (Note that state 5 is skipped here because the system found only 1 option based on the first three preferences. Going to state 7 and skipping state 5 would be the same for the cases state 3 -> 7 and state 4-> 7.)

**4.2.3 State 3.** State 3 is the type state where the system asks for the food type the user would like to eat. Depending on the utterance of the user, the system can move to different states.

Transitioning to state 4 Price -The user expressed the area and the food type was already known.

Transitioning to state 5 Additional requirements -The user expressed the area, but the food type and price were known. And the system found more than 1 option.

Transitioning to state 6 No match -The user expressed the area, and the food type and price were known. But the system found no options.

Transitioning to state 7 Recommend -The user expressed the area and the food type and price were known. And the system found exactly 1 option.

**4.2.4 State 4.** State 4 is the state where the system asks for the price range. The possible transitions are the following:

Transitioning to state 5 Additional requirements -The area, food type and price are known. Only if the system finds more than 1 option it goes to state 5.

Transitioning to state 6 No match -The area, food type and price are known, but the system finds no options.

Transitioning to state 7 Recommend -The area, food type and price are known, and the system finds exactly 1 option.

**4.2.5 State 5.** State 5 is the state where the system asks for the additional requirements.

Transitioning to state 5 Additional requirements -Normally all states can stay in the same state by repeat, but here it's worth to mention that if the user has chosen a requirement that's not included in the list (such as busy romantic or children), it stays in state 5 and remind the user of which additional requirements the user can choose from.

Transitioning to state 6 No match -The user expressed an additional requirement. If the system finds no options together with the previous preferences it goes to state 6.

Transitioning to state 7 Recommend -The user expressed the area and the food type and price were known. And the system found at least 1 option and not exactly 1 option anymore.

**4.2.6 State 6.** State 6 is the state where the system can't find any options (anymore), but it's still possible to go to state 5 or 7 (and 1 and 9).

Transitioning to state 1 Welcome -When the user's utterance is classified as affirm, it will assume that the answer on the question whether the user want to restart or not is yes. In this case it will go to state 1.

Transitioning to state 5 Additional requirements -If the act is inform it can go to state 5 when the inform act was an utterance where the user edited his preferences and there were more than 1 restaurants found by the system.

Transitioning to state 7 Recommend -The same as before, if the act is inform where the user edited his preferences the system can go to state 7 when the system only found 1 recommendation.

Transitioning to state 9 Survey -When the user doesn't need any help anymore (when the utterance of the user is classified as negate) it will go to state 9. There is a survey (not compulsory of course), that the user can participate in.

**4.2.7 State 7.** State 7 is the state that recommend restaurants. It can go to state 6, 7 and 8.

Transitioning to state 6 No match -When the user is not satisfied with the current recommendation, the system can answer no. But when there are no recommendations left it can go to state 6, no matches left. An example snippet is following:

**Example 4:** An example of going from state 7 to state 6

system: "Okay, I came up with a recommendation: nandos. Sounds good?"

user: "No, It's not."

In this case it will go to state 6 when there are no options left anymore.

Transitioning to state 7 Recommend -Here again, when the user is not satisfied with the current situation he or she can say no. But in this case when there are other options left the system will stay in state 7 and recommend a new restaurant.

Transitioning to state 8 Phone address and ending -When the user is satisfied with the recommendation the system will go to state 8. Here it will give the phone number and address of the recommended restaurant.

**4.2.8 State 8.** State 8 sort of the last state of the chatbot. After state 8 the chatbot has done her job, and state 9 is for the survey. But before going to any state the system confirms if the user gets the information of the phone number and address or not, and greet the user with an ending sentence. Here the system can also go to state 1 of course by reset (not included in the graph since reset works in every state).

Transitioning to state 9 Survey -When the user gets the information the system can move to state 9. Example snippet is the following:

**Example 5: An example of going from state 8 to state 9**

```

system: "Okay, I came up with a recommendation: la margharita. Sounds good?"
user: "Yes it sounds good!"
system: "The address is 15 magdalene street city centre. The phone number is: 01223315232. Okay?"
user: "Okay"
system: "I sure hope I was of help for you, It would be really awesome if you say bye to me before you leave, but if
you need me to recommend more restaurants, respond with reset."
user: "Bye"
here it goes to state 9
system: "I hope you liked our conversation, I sure did! How would you rate my help on a scale from 1 to 5? (1 being
very bad, 5 being very good)"

```

4.2.9 *State 9.* The last state, survey. The user can rate the chatbot and after rating it, (or the user can say "exit" and the user doesn't rate it but just leave) it will exit the conversation. This state is created on purpose for later analysis.

### 4.3 System utterances

For the system utterance we made 2 templates, one template containing formal responses and one containing informal responses. The responses were generated by four function but mainly generated by the `giveresponse` function.

### 4.4 `giveresponse` function

Here we made two dictionaries (formal and informal), where the key was a word and the value the utterance. An example of how this could work (simplified version):

```
if (state == 1):
```

```
  giveresponse("welcome", formal)
```

This will return the utterance that corresponds with the key "hello" which is: "Hello welcome to this restaurant recommendation guide. You may ask for restaurants by area, price range, or food type. To start, please enter your request. "

Something worth noting is that for the formal responses, if `giveresponse` receives parameter "hello" it can greet the user based on the current time. So it's either good morning, good afternoon or good evening.

Next to this some keys has a list as value containing different responses. The chatbot will randomly choose one of the responses.

### 4.5 `giverecommendation` function

Next to the main utterance generator function `giveresponses`, we have two functions which is responsible for the utterance where our chatbot needs to recommend restaurants. This is generated by calling the function and give it the argument of the lists of the restaurants matching the preferences and the boolean formal.

An example would be: `response = giverecommendation(restaurants, formal = true)`, would lead to a response of: "Thank you, I came up with the following recommendation: {}. Are you satisfied? "

where {} is filled with a randomly chosen restaurant from the restaurants list that is gives as argument.

#### 4.6 checkrule function

The other function responsible for generating utterances for recommending restaurant is the checkrule function. Here it does the same as the function giverecommendation but it will bind a reason utterance to it. The difference between using giverecommendation or checkrule function is based on whether there were additional requirements or not. If there were it will use checkrule function to recommend restaurants. For example if an user had an additional requirement of "romantic", using checkrule it will return the following response:

"Okay, I found a restaurant for you: la margherita. this one is perfect for some romance, because you can spend a long time in this restaurant. Sounds good??"

As you might have noticed, for each additional requirements there are templates for the reason why this restaurant satisfies the additional requirement.

#### 4.7 givemoreinformation function

At last there is a givemoreinformation function that returns the phone number and address by using a fixed template. The informal response is

"The address is: ... The phone number is: ... okay?"

### 5 REASONING

If the conversation reached the state in which the preference list is complete and there is more than one restaurant suggestion, the user will be asked if there are any additional requirements. If this is the case, a rule-based system is used to check if any of the restaurant that match the user's preferences also matches the additional requirement(s). The system uses six rules for three options of additional requirements (consequences): "busy", "romantic" or "children". All rules consist of one or more antecedents and one consequent. To check if the restaurants in the list match a given requirement, the system checks whether the antecedents of the rule are true or false for the restaurant. For example, for a restaurant to be "romantic" (consequent) the antecedent "long stay" has to be true. The restaurant is then returned as suggestion, together with a description of the reasoning that is used (for each rule, one reason is written out to return). The list of restaurant options is returned as well, from which the suggested restaurant is removed. This is done to allow the user to ask for any other restaurant in the list that matches the user's preferences and requirement as well. The process will repeat itself for the updated list of restaurants if there are still enough options. The function looks at one consequent at a time and the first restaurant that gives a match with a is returned to the user. If there is any conflict with other rules or restaurant information, the system will just follow one rule (which is the first match with a restaurant and a rule). If there is no match with a restaurant in the list for the given additional requirement, the function will return empty values for the restaurant suggestion and reason, and the dialog manager will communicate to the user that no match could be found. An example of this reasoning component in the code and the dialogue is shown in the snippets below. This is for the earlier mentioned rule: if the consequent is "romantic" then the antecedent "long staytime" must be true.

---

```

elif consequent == "romantic":
    (...)
elif (list(rest_info['staytime'])[indexje] == "true"):

```

---

Fig. 2. Example of the implementation of the reoning aspect

**Example 6: An example of a dialog for the reasoning component**

system: Okay then! Any additional requirements or not???

user: i want the restaurant to be romantic

system: Okay, I found a restaurant for you: yu garden. This one is perfect for some romance, because you can spend a long time in this restaurant. Sounds good??

**6 CONFIGURABILITY COMPONENT**

The systems utterances can be set to formal or informal. This option was chosen to study the change in the ratings of the chatbot by the users. The formal and informal utterances are formed by applying some simple rules also used by Liebrecht et al. in their study. The informal sentences are made "informal" by using abbreviations, emoticons and excessive use of the question mark sign. The word "Would" is replaced by the word "Do". Lastly the informal sentences have more adjectives and adverbs. The chatbot's default chat style is informal. Moreover the system asks after the last state for the user to rate the chatbot. The user can respond with a number from 0 to 5, which gets stored in a file. This additional configurable component is implemented for further research on the system. As mentioned in the dialogue manager, another configurability component was the option for the your to say that 'any' preference may be chosen for the restaurant, if the user has no specific preferences for the price, area or food type.

**7 DISCUSSION**

Although this report aims to show how this relatively simple system for a chatbot works quite well for restaurant recommendation for user preferences, the system still has important limitations that make improvements or alternative approaches desirable for next versions. In this discussion general system limitations and possible improvements will be discussed, as well as for the data labelling classification.

To begin with, one general limitation of the model is that it does not always lead to a very 'natural' conversation. First of all, this can be due to the preference keyword matching function. To look for keywords, a Levenshtein distance of one is used, to compare keywords to words in the utterances. This allows the user to make one typing error and still extract the correct preference. However, this can also lead to some words being matched with a keyword for a specific preference, while this is not correct. For frequent words, the model is ordered to continue, so this will not lead to problems (for example the word 'that' in an utterance leading to 'thai' as food type preference). For less frequent words, this problem cannot be prevented this way. This will lead to the extraction of the wrong preference, without the user understanding why. Two alternative approaches to this problem are: making it an option to use or to not use the Levenshtein distance for preference keyword matching; or an extra response from the system could be added that asks the user to confirm the preferences that the system has extracted from the user's utterances. Another limitation that might make the system seem a bit inflexible and unnatural, is that additional requirements are now only being asked

for at the end of the conversation when the preference list is complete. Therefore, a good improvement to the model would be to allow for additional requirements, such as 'romantic', to be stated and saved earlier in the conversation. Improvements like more response options and more use of patterns could make the system seem a lot more natural as well.

For the classification of the dialog acts of the utterances, several limitations can be noted. Evaluation of the language models has shown that the performance of the machine learning models are the highest and do not differ a lot from each other. This makes these machine learning models most useful to classify dialog acts. However, as explained in this report, the limitations to the data set on which the models are trained, also have implications for the performance on the act classification. The data for the dialog acts is only in spoken language. As explained earlier, written language use can lead to users being more informal to the chatbot when they chat using spoken language. This would create a difference between the set that the models are trained on and the set of utterances that users will feed the chatbot in practice. This difference can lead to misclassifications of dialog acts during the conversations, even if a model has a high performance on its test set. This limitation may be overcome if training is done on larger datasets that contain more dialogues, with more language variation and, preferably, written dialogues.

## 8 TEAM MEMBER CONTRIBUTIONS

Contributions				
Task	Amy Oey	Kathleen de Boer	Andrea Roijen	Sunny Hseih
Report Part 1a	3h	6h	6h	3h
baseline systems	3h		5h	
machine learning model	1h	4h	3h	4h
data loading and preprocessing	1h	1h		1h
quantitative evaluation	4h		1h	1h
error analysis	5h	2h	4h	3h
difficult instances	1h	3h	4h	3h
Report Part 1b	5h	5h	7h	5h
diagram	2h	1h		4h
state transition function	24h	7h	4h	18h
identifying user preferences	4h	5h	4h	3h
Levenshtein distanced		4h	5h	
CSV lookup function		1h	1h	
Report Part 1c	2h	4h	5h	5h
implication rules		5h	8h	
configurability	6h	3h	2h	6h
integration with dialog management	3h		1h	2h

## REFERENCES

- [1] Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*. 263–272.
- [2] Christine Liebrecht, Lena Sander, and Charlotte Van Hooijdonk. 2020. Too informal? How a chatbot's communication style affects brand attitude and quality of interaction. In *International Workshop on Chatbot Research and Design*. Springer, 16–31.

- [3] D.R. Olson, N. Torrance, A. Hildyard, and Cambridge University Press. 1985. *Literacy, Language and Learning: The Nature and Consequences of Reading and Writing*. Cambridge University Press. <https://books.google.nl/books?id=kq48AAAAIAAJ>