Building A Goal Based Conversational Agent With State Transition Function For Finding Restaurant Recommendations In Cambridge, UK

LUCA LIN 5522146, IVAN KONDYURIN 9086765, CÉSAR DI MEGLIO 2042789, and STAN FRINKING 7007043

In this project, we develop a goal-oriented dialog agent in the restaurant domain by applying Supervised Machine Learning for user utterance Classification. The agent makes use of a Dialog State Diagram to make informed decisions on what to respond to the user based on the classified user utterance and moves in between states to advance the conversation towards the goal of the user, finding a suitable restaurant. In the end, we will discuss the results and outline shortcomings of our approach.

Additional Key Words and Phrases: Logistic Regression, Feedforward Neural Network, Information Retrieval, Keyword Matching, NLP, Bag of Words

ACM Reference Format:
Luca Lin 5522146, Ivan Kondyurin 9086765, César Di Meglio 2042789, and Stan Frinking 7007043 . . Building A Goal Based Conversational Agent With State Transition Function For Finding Restaurant Recommendations In Cambridge, UK. 1, 1 (October), 16 pages.

1 INTRODUCTION

Goal-based conversational agents can provide a quality of life improvement in many domains for users. One example is the restaurants domain. We build a goal-based conversational agent which uses Machine Learning to classify user utterances to understand the goal of the user and jumps between states to help guide the user to find a suitable restaurant in Cambridge, UK.

In section 2 we will show the structure of the data and outline the State Transition Diagram. Next, in section 3, we will explain the preprocessing steps taken to make the data machine-readable, present the Machine Learning classifiers and compare their performance to baseline models. Difficult sentences will be analyzed and we will interpret the errors. In section 4 and 5, the State Transition Function and the Reasoning extension will be explained in detail. Finally, we will discuss our approach and adjustments we can make for future research.

2 DATA

The dataset for this project consists of dialog acts from the Dialog State Tracking Challenge DSTC2. In total, 3235 dialogs were collected in the restaurant domain. Each dialog consists of a number of utterances between the system and the user and the goal is for the user to use a certain set of preferences to obtain a restaurant recommendation. The dialogs are recorded using automatic speech recognition, which influences the quality of the dialogs. Moreover, the original dataset contained multiple labels, but only the first label was used in the final dataset for this project. An example dialog taken from the dataset:

Authors' address: Luca Lin 5522146, l.lin1@students.uu.nl; Ivan Kondyurin 9086765, i.kondyurin@students.uu.nl; César Di Meglio 2042789, c.t.dimeglio@students.uu.nl; Stan Frinking 7007043, s.frinking@students.uu.nl.

This is an assignment that was made for the course Methods in AI Research as part of the Master's degree in Artificial Intelligence.

```
Task 03888: You want to find an expensive restaurant and it should be in the south part of town.
Don't go for the first venue the system offers you, ask if there is anything else.
Make sure you get the address, phone number, and postcode.
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:   I'm looking for an expensive restaurant in the south part of town
system: What kind of food would you like?
user:   I don't care
...
```

The utterances vary greatly in depth, with some sentences only containing the words 'phone number' and others being more complex such as 'may i have the address phone number and price range'. Moreover, the dataset contains noise in the form of 'unintelligible' or 'noise', due to the way the data was collected. Each utterance is labeled according to 15 dialog acts labels. All 15 dialog acts can be found in table 1.

In order to work with the dataset, we first apply NLP preprocessing techniques such as the bag of words representation to make the dataset machine readable.

2.1 Restaurants Dataset

The restaurant dataset stores information about restaurants in Cambridge, UK in a tabular format. For each restaurant, the price range, the area, the type of food, the phone number, the address and the post code are given. Moreover, information was added to the database for extended functionality. More about this in section 5. An example restaurant from the database:

restaurantname	**pricerange**	**area**	**food**	**phone**	**address**	**postcode**
saint johns chop house	moderate	west	british	01223 353110	21 - 24 northampton street	c.b 3

There are 108 restaurants in the file. Certain restaurants in the file are missing values which compromise the recommendation because there is -for example- no phone number or address to recommend.

2.2 System Sentences and Dialog Structures

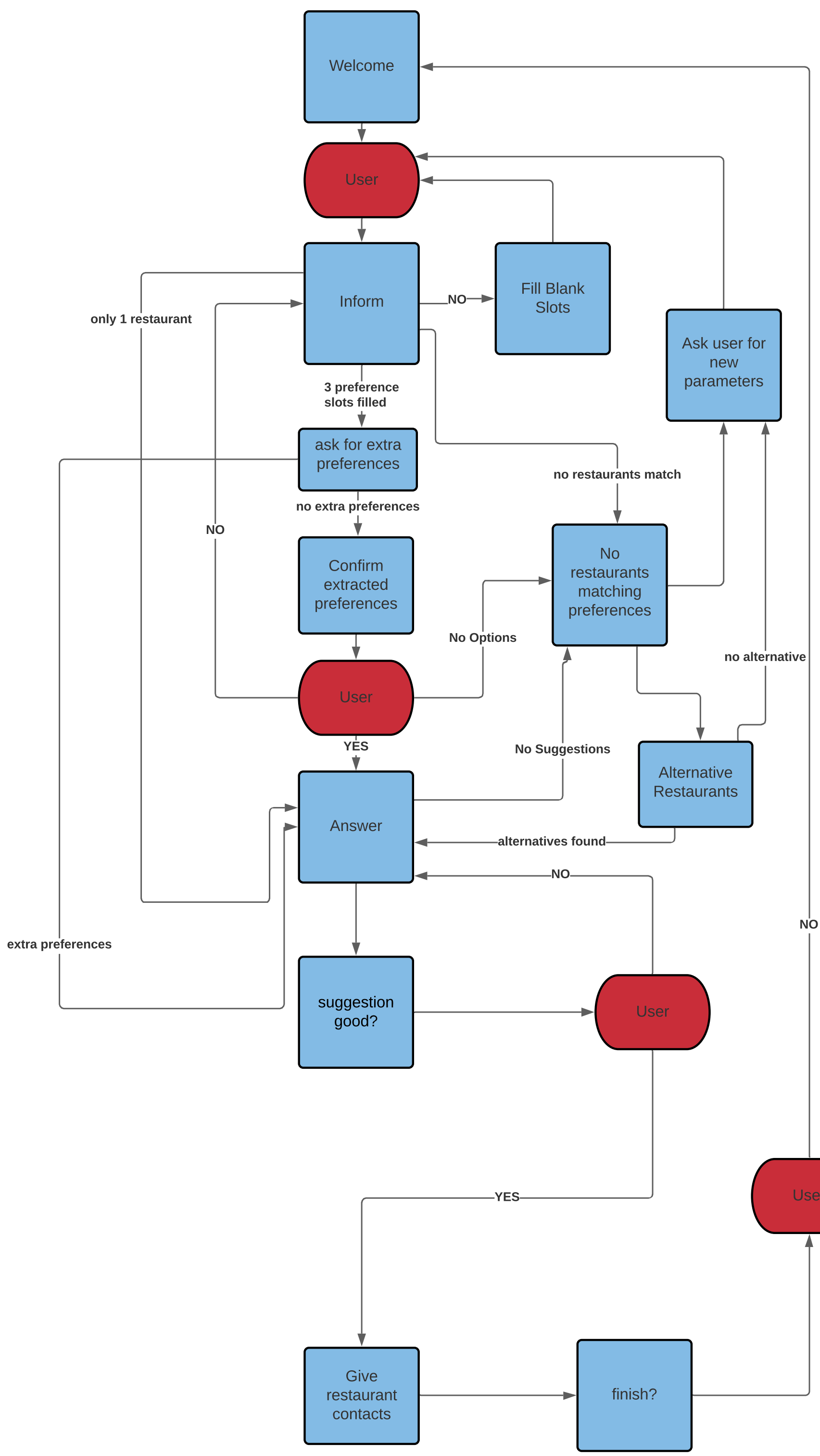
This is a text file which contains examples of restaurant recommendation dialogs between system and user. The file gave us inspiration for the system sentences, which in turn guide the course of the conversion to recommend restaurants. Indeed, the system's responses each refer to a certain state in the state transition diagram (see below) and therefore helped in structuring the overall integrity of the dialog agent. Finally, this file was used to evaluate the average length of a dialog for this purpose, thence the user would not consider the agent as too precipitated nor too time consuming.

2.3 Dialog Agent States

The dialog agent classifies each user utterance and makes use of a State Transition Diagram to enter and exit states. In each state, the goal is to advance the conversation to satisfy the goal of the user, which is to find a suitable restaurant. Below is an example dialog of the system and the State Transition Diagram:

2.4 State Transition Diagram With Example Dialog Snippet

State Transition Diagram



Example Dialog:
S:Hi, let's choose a restaurant! Where do you want to eat? Area, price range, food type? #welcome state
U:hi i would like some spanish food
S:So you would like me to find a restaurant serving spanish cuisine. What part of town do you have in mind?
#after user utterance, the system checks if all preferences are filled. In this case, we go to the 'Fill Blank Slots' state, which tries to fill the 3 preference slots.
U:perhaps something in the center
S:So you would like me to find a restaurant in the centre of town serving spanish cuisine. What's your budget?
Cheap, moderate, or expensive? #preferences slots still not filled
U:any price is fine
S:Any other requirements? You can choose from: #ask for extra preferences
Good food, open kitchen, good hygiene, children friendly, romantic or busy
U:get me something busy
S:So, you want a restaurant in the centre part of town, with any price range, serving spanish food, am I right?
#there were no restaurants with the extra requirements, we go to 'confirm extracted preferences' state
U:yes #user confirm, go to 'Answer' state
S:I have found a cool place matching your preferences: 'la tasca'. You like it?
U:no #suggestion not good, go back to Answer state
S:Okay, I came up with a recommendation: 'la raza'. Sounds good?
U:yes
S:Alright, here are the contacts:'la raza', 01223 464550, 4 - 6 rose crescent. Would you like to finish here?
#give restaurant contacts and go to 'finish?' state
U:yes
Thanks, hope it was useful. See you! #Thank you, goodbye

3 MACHINE LEARNING

3.1 Label statistics and Preprocessing

label	frequency	avg length
'inform'	10160	19.9
'confirm'	172	24.5
'affirm'	1156	8
'request'	6494	19.4
'thankyou'	3259	17.7
'null'	1612	6.1
'bye'	266	11.7
'reqalts'	1747	21.3
'negate'	435	8.6
'hello'	93	27.6
'repeat'	33	11.8
'ack'	28	14
'restart'	14	11.6
'deny'	27	14
'reqmore'	5	4

3.1.1 Imbalanced dataset. The label distribution in table 1 indicates an imbalanced dataset. However, due to both Logistic Regression and Neural Networks assigning weights to the features, undersampling/oversampling was not necessary.

3.1.2 Preprocessing. In the previous section, we explained that the utterances were recorded by making use of speech recognition. The utterances in the dialogs were lower cased and excluded punctuation and special characters, simplifying the preprocessing steps.

To reduce the vocabulary size for the bag of words representation, we applied stemming from the NLTK package to the sentences. Words such as 'restaurant' were stemmed to 'restaur' as an example. Before applying stemming, the bag of words representation has 767 columns. After applying stemming, we have 653 columns.

In the literature, stopword removal is used to reduce the vocabulary and thus the size of the bag of words model. However, upon inspecting NLTK's stopword dictionary, we found that it contained many words which were important for our classification task, such as the word 'any', used to indicate no preference and a family of negation words such as 'no' and 'not', which can indicate a negation label. To further research whether we could use stopwords in our task without having to generate a list manually, we printed the top 10 most frequent words in our dataset (see appendix). However, upon inspecting them, we found that words like 'thank' and 'bye' were present, which are used to identify 'thankyou' and 'goodbye' labels respectively, which were important in identifying two labels in our data. We have thus decided not to remove stopwords in our project.

To split the data into train and test sets, we make use of Sklearn's train_test_split package. For our dataset size, we make a split of 0.85 in training set, 0.15 in test set.

3.1.3 Bag Of Words Representation. A Bag of Words model (BoW) displays a sentence as an unordered set of all the words where each word in BoW is given a unique ID, with no representation on order or context. Since the order of words is not included in the representation, the syntactic differences between two sentences is not represented. For example, in the sentences "the man hit the child" and "the child hit the man", there is a difference in meaning. However, with the BoW representation, they are treated the same. Because we are mainly interested in the content of the sentences for label classification (more about this in section 3), we believe this will perform well enough for the scope of this project.

3.2 Baseline Models

classifier	accuracy
Baseline 1	0.430
Baseline 2	0.896
Logistic Regression	0.978
Neural Network	0.985

Table 1. Trained models and accuracy scores

As a preliminary step, two baseline systems for simple rule-based classification were implemented. The goal was to establish a benchmark for comparison with machine learning classifiers and to analyze the data for potential difficult cases. Besides, a baseline system with sufficient accuracy can be used as an alternative means of classification on the dialogue agent.

3.2.1 Majority Label Model. For the first baseline, the data was split into a training set, used to calculate the majority label, and a test set, used to test the system's accuracy. The baseline achieves an accuracy of about 40%, which is the number of occurrences of the majority label 'inform' in the whole dataset.

3.2.2 Keyword matching. The second baseline system utilised a keyword recognition approach. We defined the keyword dictionary which would include meaningful keywords for each label in the dataset. For each label, the structure and meaning of utterances were analyzed, and a list of words and collocations was created, which covered as many semantically sensible utterances with a given label as possible. It was important to avoid overlapping of keywords, because without a weighting function it would be impossible to differentiate between two labels with the same keyword. Another constraint included avoidance of using extremely frequent words, such as personal pronouns, for the probability of finding them in other utterances is also high. In such cases we had to use word collocations, such as: 'is it', 'is there', as they explicitly account for possible beginnings of a confirmation question. Another difficulty was related with ambiguous classification. Even though the lists of keywords did not overlap, some user utterances included keywords from at least two categories. The most common example of such ambiguity was a typical goodbye sentence 'thank you good bye', where two keywords ('thank you' and 'bye') are found. In order to account for such cases, we had to range the labels with reference to their role in the dialogue structure. Thus, if a sentence contains multiple keywords, the system labels it with the keyword from the more general label, so that such utterance as «Wrong, reset» would be labelled as 'restart' rather than 'deny'. The accuracy of the second baseline was also assessed, and the errors were analyzed in order to improve the performance. After the division of the keywords lists into three dictionaries, the accuracy has reached 89.6%.

3.3 Machine Learning Models

3.3.1 Gridsearch for hyperparameter tuning. In this section, we will explain hyperparameter tuning for the classifiers. For this purpose, we made use of GridSearch from the package SkLearn. GridSearch takes as input a classifier and a set of hyperparameters and makes use of k-fold Cross-Validation to output the accuracy for each combination of hyperparameters. Taking a look at the data, we found that the label 'reqmore' had only 5 examples in the dataset and thus set k to 5. Below are the hyperparameters we used to feed the gridsearch algorithm for each model:

Activation Function	Softmax
Solver	LBGFS
penalty	['l2','none']
n	[300,400,500]

Table 2. GS hyperparameters for Logistic Regression

3.3.2 Gridsearch for Logistic Regression. The best hyperparameter results in terms of accuracy were **n**=400 and **penalty**=none, with an accuracy of 0.978%. We found that for low iterations (100), the optimization algorithm did not converge. For the Logistic Regression classifier, it seems like the model performed better on the test data when it performed no regularization, indicating that it did not overfit on the training data. Moreover, it could be that the dataset is low in noise, making regularization less important for this dataset.

3.3.3 Gridsearch for Neural Network. Adam is a stochastic gradient-based optimizer, proposed by Kingma, Diederik, and Jimmy Ba. The documentation of Sklearn indicates that adam

Activation Function	ReLU
Solver	adam
Hidden Layers	[(50),(50,50),(50,50,50), (100),(100,100), (100,100,100)]
n	[100,200,300]

Table 3. GS for Neural Network

works well on large datasets (thousands of training samples), which was the case for us. We found that the best results of Gridsearch in terms of accuracy given by the hyperparameters above were **H** = (100) and **n** = (200), with an accuracy of 0.984. We at first predicted the classifier to perform better when increasing the number of hidden layers. However, the classifier would overfit on the training data, causing it to perform worse on the test data and not being able to generalize well enough. Dropout regularization is used to prevent a Neural Network model from overfitting, where nodes in the hidden layer would be dropped randomly from the model. Unfortunately, due to limitations in the package of SkLearn, we did not implement this. For future research, we will consider making use of other, more complete Neural Network packages.

3.4 Errors analysis

To analyze the misclassifications from our classifier and find out which sentences were hardest, we saved the misclassifications in the triple form (utterance, y_pred, y_true) and looked at which were most frequent. An example sample of the triple form can be found in table 2. To find out which categories the classifier had most trouble with, we first looked at frequency of each misclassification and then took a look at an accompanying sentence.

LR			**NN**		
y_pred	**y_true**	**freq**	**y_pred**	**y_true**	**freq**
reqalts	null	5	null	inform	5
inform	null	14	null	request	3
inform	reqalts	9	confirm	inform	3
...	...	...	...	...	...

Table 4. Sample of Frequencies of misclassified sentences and their true label

3.4.1 Keyword Matching Baseline. The keyword matching baseline classifier misclassifies most sentences which were noise as 'null'. These were the sentences containing keywords such as 'noise' or 'unintelligible' such as 'im looking for a restaurant that serves unintelligible food'.

Second, sentences which had multiple labels in the original dataset, such as 'okay can i have the phone number and address', were difficult to classify because only one label was used. Labels which were close in terms of semantics, such as 'inform' and 'reqalts', were hard to classify as well because of this. It is clear that this approach has limitations.

stemmed sentence	**y_pred**	**y_true**
unintelligible i dont care	null	inform
unintelligible food	null	inform
okay can i have the address	ack	request

Table 5. Example sentences misclassified by the baseline Keyword Matching model

3.4.2 Logistic Regression and Neural Network. Now, we will discuss the errors made by the chosen classifiers. Both Neural Network and Logistic Regression classifiers perform similarly and have similar errors. However, one difference is that the Logistic Regression classifier has many more instances of misclassification of the majority label: (inform, null, 14), which we will discuss first, as opposed to the Neural Network (inform,null,2).

3.4.3 Classifier bias. Examining the (inform, null) misclassified labels for the Logistic Regression Classifier, these sentences include mostly single word, infrequent sentences, such as ('or',inform,null) or ('child',inform,null), which were manually tagged as null in the dataset. The words in some of these sentences only occur once in the whole dataset, which makes it possible that the words were not trained on and therefore the classifier just assigned it to the highest label, 'inform', since the Bag of Words model ignores the words it does not know. Unlike the Logistic Regression classifier, the results of the Neural Network classifier do not seem to indicate a bias towards the majority label but towards the 'null' label. For example, in sentences such as 's e a f o o d', due to each letter being treated as a word, the Logistic Regression Classifier uses its bias to correctly predict 'inform' label but the Neural Network misclassifies this as 'null'. One explanation is the difference in activation function of each classifier, with Logistic regression using a softmax activation function and the Neural Network having a ReLU activation function. This means that for the Neural Network, when the summed input of a neuron is not strong enough, the neuron does not fire. Because of this, the loss that is backpropagated through the system is either 0, in which case the weights are not updated, or 1. The softmax function returns a probability, which means the weights are always updated depending on the gradient of the loss and because there are many instances of 'inform' label, the loss function would keep updating the weights towards this label.

3.5 Logistic Regression and Neural Network misclassifications

Stemmed Sentence	**y_pred**	**y_true**
how about tha	reqalts	null
or	inform	null
how european food	inform	reqalts
right on good bye peac	bye	affirm
can i get the next one	null	reqalts
derat price	request	inform
yea im look for an expens	affirm	inform
what about unintellig food restaur	reqalts	null
halo	inform	hello
what about other part of town	reqalts	inform
veri good thank you good bye	thankyou	bye
okay can you give me anoth restaur	reqalts	ack
yes i dont care about the price rang	affirm	inform
and is this a moder price restaur	inform	confirm
unintellig hello	null	hello

Table 6. Example of LR misclassified sentences and their true label

In this subsection, we will give examples of misclassified sentences which were found in both the Logistic Regression and the Neural Network classifiers.

3.5.1 Labels Used in Conjunction. Sentences which could have been classified as two labels, were difficult for the classifier. For example, both models misclassified the ('inform', 'reqalts') tuple, with LR 9 times and NN 8 times. Some examples of these sentences are 'are there any restaurants serving turkish food' and 'how european food'. In these cases, it is possible that the sentences could have had two labels in the original dataset. Sentences with 'yes' or 'yeah' were also difficult due to these keywords being present, indicating an affirmation.

Another misclassified tuple was ('thankyou', 'bye'). These labels are sometimes used in conjunction with each other, which makes it hard for the classifiers to predict. In the dataset, we found 3259 'thankyou''s and 266 'bye''s. This could be an explanation as to why the sentences were labelled as 'thankyou' and not the other way around.

3.5.2 Label Specific Words. The tuples ('reqalts', 'null') and ('reqalts', 'inform') were also misclassified tuples. When we closely examined the misclassified sentences, we found that the sentences were either 'how about'/'what about' or contained it within the utterance, such as the sentence 'how about tha' or 'what about turkish'. In the dataset, almost all sentences including 'how about' and 'what about' were labelled as 'reqalts', which indicates that the classifiers overfit on these words. Affirmation keywords such as 'yes' and 'yeah' also fall under this category of misclassifications.

Concluding this section, it seems like the most difficult sentences were the ones where two labels are used in conjunction but only one was used as label. Moreover, label specific words such as 'yes' or 'how about' were more difficult to classify. An example of a difficult to classify sentence is therefore: "yes I would like to find a restaurant" or "what about Indian food". We think adding an extra feature representing the length of the sentence could prove useful.

Table 7. Examples of hard to classify sentences due to errors in labeling or ambiguity of label

Stemmed sentence	**y_pred**	**y_true**
how european food	inform	reqalts
are there ani restaur serv turkish food	inform	reqalts
yeah im look for a restaur in the south part of town that serv intern food	inform	reqalts
yes i dont care about the price rang	affirm	inform
yes and i dont care about the price range	affirm	affirm
veri good thank you good bye	thankyou	bye
how about tha	reqalts	null
what about other part of town	reqalts	inform

4 DIALOG MANAGER

4.1 State transition function

4.1.1 Finite state machine. The dialogue management system was created basing on the concept of a finite state machine (FSM). It is an abstract model of a machine that includes a set of states (one initial state and at least one final state), a state transition function, a set of input events, and a set of output events. For the case of a restaurant recommendation system, inputs are user utterances, and outputs are system responses. The state transition is performed based on the current state and the input. The goal of this FSM is to implement the dialogue structure (formalized in the transition diagram) keeping the architecture of the dialogue management system clear and expandable.

4.1.2 Recursive function for finite states. In order to implement a finite state machine, a class for dialogue agent was created, which allows for storing preferences within an object of this class. The general dialogue structure was represented by means of a recursive function that takes user utterance, current state, and the list of user preferences as input parameters, performs a particular set of actions depending on the state, and calls itself again with new parameters. The only state which does not involve a new call is 'exit.' The states and their names are generally based on the labels of the training dataset, except those that account for the functions that are not implemented in this system (for example, 'request') or represent the states of the transition diagram, such as 'confirmpreferences'.

4.2 Structure of the dialogue management system

4.2.1 Initial stages and informatin extraction. The dialogue management system is launched with an 'init' state, which functions similarly to 'hello,' but can only occur at the beginning of the session. At states' init' and 'hello,' a brief user instruction is provided, and the new input utterance is classified. They correspond with the 'welcome' state of the transition diagram. The 'inform' state is used in several cases when the user utterance contains either a request about a restaurant or some additional parameters for this request. As a user is expected to provide some meaningful information about his request at this state, a preference extraction function is launched, and a list of user preferences is built based on its output. This step corresponds with the «Area AND price AND foodtype in utterance?» state of the diagram. If the preferences enable the system to recommend one particular restaurant, or there are no matches at all, the state switches to 'answer' as the system is ready to provide the final answer. Otherwise, it switches to 'fill blanks.' The 'fill blanks' state matches the 'ask user…' state of the diagram; it includes filling the list of user preferences by asking questions if some preferences are absent. If all slots are filled, but multiple suggestions are possible, it switches to 'ask extra preferences' state, which launches a function of the same name, which extracts additional user preferences, such as 'open kitchen' and provides a recommendation.

4.2.2 States for preference processing. When the preferences are extracted, the system proceeds to 'confirmpreferences' state, where extracted preferences are printed and the user can either approve them (then the system proceeds to restaurant suggestions and the state switches to 'answer') or request a change (in this case, the state returns to 'inform' and the list of preferences is cleaned. At the 'answer' state (corresponding with «Suggest restaurant» state of the diagram), the system tries to use the output of the 'lookup' function that looks for matching restaurants in the database. If the result satisfies the user, the system provides the restaurant's contact details and switches to 'goodbye' state. Otherwise, it removes that suggestion from the list of possible suggestions and remains in the same state. If the lookup function fails, an alternative search with alternative preferences is launched based on the user's initial preferences. Its results can be approved by the user (moving to 'thankyou' state) or declined (in this case, the user preferences are reset, and the system returns to the 'inform' state).

4.2.3 Other states. The 'thankyou' and 'goodbye' states are similar: they include a warning message asking the user if he or she wants to finish the conversation, and either stop the dialogue function by switching to terminal state 'exit', or relaunch the system by returning to 'init' state. The 'repeat' state accounts for situations when the user demands to go back. In this case, the previous state and user input are recovered from the log of dialogue manager. All other states, which can be caused by erroneous classification or unexpected user behavior, result in an output message which suggests the user repeat. The system state remains unchanged since the previous step.

4.3 System utterance templates

The system answers were designed to provide unambiguous information for the user. Two sets of system responses are included: a formal and an informal one. To make conversations more natural and to clarify the information when needed, some system answers have multiple wordings. For example, there are three different options for a welcoming utterance, and some of them include the following explanation: «To start, please enter your request.» Multiple options are also provided for sentences that contain recommendations (as a user may wish to see more than one recommendation; this diversity is essential for the dialogue not to be monotonous and artificial). Whenever possible, the system questions specify the expected options to ensure that a user does not come up with an unexpected answer. For example, the question about price range preference is formulated as follows: «What is your desired price range? Cheap, moderate, or expensive?» To make the user experience more informative, and to amplify it with additional grounding, a confirmation question is provided, where all the extracted preferences are stated: «So, you are looking for a restaurant in the 0 part of town, with 1 price range, serving 2 food, correct?»

5 REASONING

5.1 Adding Data and Implication Rules For Reasoning

In this section, we will cover the extension function which implements a set of rules for extra user preferences. The goal of this extension function is to reason about these extra preferences and derive new knowledge using inference rules. For this purpose, we add extra user preferences to the database 'restaurant.csv':
hygiene: indicating the restaurant has good hygienic standards
good food: indicating the restaurant has good food

open kitchen : indicating the restaurant kitchen is visible to the customer (buffet-style).
Among the 109 listed restaurants, we labeled 55 of them with ‘good food’ (about 50.46%) , 40 of them with an ‘open kitchen’ (36.7%) and 88 of them with the ‘hygiene’ standard (80.7%).

In addition to these extra preferences, a set of implication rules are added. Some of the rules can directly be inferred from the dataset (lv1 rules) and some can only be inferred from other rules (lv2, for example the rule ’long time’ -> ’romantic’. ’long time’ is not a part of the dataset and has to be inferred from other rules such as ’spanish’ -> ’long time’). A list of our added rules can be found in table 8. The full table can be found in the appendix.

id	antecedent	consequent	t/f	level	description
7	bad hygiene, open kitchen	romantic	F	1	A restaurant where you can see inside the kitchen and where the kitchen is not clean is not romantic
8	bad food, bad hygiene	busy	F	1	Restaurants with bad quality and low standards are not busy
9	open kitchen	children	T	1	Restaurants with open kitchens and all you can eats are popular for families
10	long time, not open kitchen	boring	T	2	Restaurants which serve slowly and ’a la carte’ tend to be boring
11	boring, expensive	busy	F	2	A la carte restaurants which are expensive and where you have to wait long for food are not very popular
12	boring	romantic	F	2	Boring restaurants are not a place to impress someone

Table 8. Added Implication Rules

5.2 Reasoning

The agent performs this process in 4 steps:

5.2.1 Extracting User Extra Preferences. When the user is done filling the standard user preferences (area, price, foodtype), the system asks the user for extra preferences. To extract these extra preferences, we look at the user utterance and perform keyword matching. If the preference is a negation, such as ’not romantic’, we include the negation in the slot. Moreover, special cases like ’bad food’ are converted to ’not good food’ for simplicity and as a pre-processing step. Because the extra preferences are limited, the system lists the options in the dialog to inform the user. The result is a list of the user’s extra preferences, which we will call ’UEP’.

5.2.2 Building a Knowledge Base. Next, we look up the list of restaurants given the user’s standard preferences. We create a knowledge base KB, containing all information about a given restaurant, taken from the ’restaurant.csv’ database. Because the implication rules include standard preferences such as the food type and the price range in the antecedents of the rules, we add those to the KB as well. The result is a knowledge base KB which includes the standard preferences (foodtype, area, price range) and the extra preferences (good food, open kitchen, hygiene).

Algorithm 1: Inference Function

```
Result: appliedRules, KB
KB ← Knowledge Base ;
IR ← Implication Rules;
AR ← Applied Rules;
for k in KB do
    for antecedent,consequent in IR do
        if k == antecedent then
            KB ← consequent;
            AR ← antecedent, consequent;
    end
end
```

5.2.3 Inference Function. In this paragraph, we will explain the implementation of the inference function. In the previous section, we stored the restaurant info in a knowledge base KB. For each k in the KB, if for any of the rules it is an antecedent, add the consequent to the KB, repeating this step until no more knowledge can be inferred. For each iteration, we keep a list of applied rules, adding the knowledge k to the list of applied rules as well, (for example 'hygiene' - > ['hygiene']). The result is a set of applied rules, and a knowledge base KB. A pseudocode can be found in algorithm 1 below.

5.2.4 Checking The Viability Given The Used Rules. To present the reasoning to the user (Algorithm 2), we look at the applied rules (AR). If the preference is present in the consequent of the rule, we check if the signs of the preference and the consequent match. If they do, we return true and the rule. If not, we return false and the rule. In case the preference is not present, we return false and no rule. Based on the outcome of this function ('true' or 'false'), we either suggest the restaurant because of rule and ask the user if it is ok, or tell the user we do not suggest the restaurant because of rule. If a restaurant does not contain the defined preferences after applying the inference function, we ignore it. Below is a sample dialog from the system showcasing this process:

Last, we will discuss how our system handles contradictions. Contradictions in our system are handled by returning the first rule that applies. In the example dialog snippet below, the inference function derives that 'nandos' is not a romantic restaurant first, followed by the inference that it is indeed a romantic restaurant. Because the system inferred 'not romantic' first, this is what it tells the user. In the second example, it infers 'romantic' first and 'not romantic' afterwards. This is a shortcoming which we will discuss in section 6.

Algorithm 2: Check Viability

```
Result: suggest or not, rule
AR ← Applied Rules;
UEP ← User Preferences;
for preference in UEP do
    for antecedent,consequent in AR do
        if preference in consequent then
            if preference.sign == consequent.sign
             then
                return true, (antecedent,consequent)
            if preference.sign != consequent.sign then
                return false, (antecedent,consequent)
    end
    return False, ("","")
end
```

```
Example dialog snippet showcasing reasoning:
Any other requirements? You can choose from:
Good food, open kitchen, good hygiene, children
friendly, romantic or busy
romantic
...
step 7 : busy->['long time', 'not romantic']
step 8 : long time->['not children', 'romantic']
nandos, this restaurant is not recommended because of busy -> ['long time', 'not romantic']
...
step 7 : long time->['not children', 'romantic']
step 8 : boring->['not romantic']
royal standard, this restaurant is recommended because of long time->['not children', 'romantic'].
Would you like to choose this restaurant?
```

5.3 Configurability

As extra configurability options, we have added a delay to the system to mimic human interaction. Moreover, we have added formal and informal sentences to the system, which can be configured as well.

6 DISCUSSION

6.1 Approach for dialog act classification

6.1.1 Keyword matching baseline. The keyword-based classifier, despite showing considerably high accuracy, revealed a number of inherent difficulties. The most frequent label, 'inform', was the most difficult to formalise using keywords. Although it was clear that all of the utterances included either such words as 'restaurant' and 'food' or one of the search parameters, the total list of keywords would be potentially unlimited: the system needs to be capable of processing a request even if the mentioned type of food is not the one existing in the database. As an *ad hoc* solution, it was decided to label the input utterance as 'inform' whenever other keywords are not present to make sure that parameter extraction function is triggered for such utterances.

6.1.2 Machine learning classifiers. For future research, we think adding sentence length could provide the classifier with a useful feature to distinguish between labels which are used in conjunction with each other such as 'thankyou' and 'bye'. The average length of 'thankyou' sentences is 17.7 whereas 'bye' is 11.7. It could also prove to be a useful feature in deciding whether an utterance is 'null' or not, as we can observe that 'null' sentences have an average length of 6.1 words, much lower than the other labels.

For representing the sentences, we think using a representation which captures context as well can prove useful. In sentences where noise was present, depending on the surrounding words, the sentence would not necessarily labeled as 'null'. For future work, we would like to try different representations, such as word2vec.

6.2 Approach for the dialog system

6.2.1 Recursive function. As mentioned earlier, the dialogue structure was implemented with a recursive function. The use of a recursive function is motivated by its adaptability to new states and changes in the existing structure, as well as the simplicity of returning to earlier stages. As the length of an average dialogue in the dataset is quite short, it is unlikely that a restaurant recommendation requires more than 1000 transitions, where the maximum recursion depth is reached. However, the system's memory consumption will grow with the increase of dialogue length, and a conversation with more than 1000 system turns may to break the system. These problems clearly need be resolved to make the system scalable and suitable for commercial use.

6.2.2 Reasoning systems. For the reasoning system, we would like to improve upon it by checking how many times it derived a user preference in the applied rules. In our current system, the first applied rule wins. However, for future iterations, we think that a function counting the number of times it satisfies the user requirement by applying a rule vs the number of times it does not satisfy the user requirement would make the system make better decisions.

7 CONTRIBUTIONS

Name	Implementation part	Report part
Luca Lin	Majority class baseline (1), machine learning classifiers (8), error analysis (6), hyperparameter tuning (2), preprocessing (2), reasoning module (3), extensions of dialog system (2), general code adaptation (4), extra configurations (0.5), debugging (4), updating code (4), documentation (4), State Transition Diagram (1)	Introduction (0.5), Abstract (0.5), Data (1), Machine learning paragraphs 2–5 (3), Reasoning (4), Discussion(0.5), formatting (4)
Ivan Kondyurin	Keyword matching baseline (1), dialog interface with state transitions (7), system utterances (1), lookup function (0.5), test and improvements (1)	3.3 Keyword matching baseline (1.5), 4 dialog manager and system answers (2), 6.1.1, 6.2.1 Discussion (0.5), contributions table (0.5)
César Di Meglio	Preference extraction (4.5), alternative set suggestions (4), updating/debugging these functions (2)	Data description (4.5 initially): 2 Data description, 2.1 Restaurant dataset, 2.2 System Sentences and Dialog Structures, Title brainstorming(0.5)
Stan Frinking	Data preparation (2), custom reasoning rules (2.5)	Data analysis and preparation (1), reasoning (1), rereading and rewriting (2.5)

Table 9. Team member contributions

7.1 Reflection

Luca: Planning for the project was good. A lot of time was spent on researching the machine learning part of the project and figuring out error analysis and difficult cases and debugging the code. We at first started the project with Colab, a jupyter notebook based web application. However, due to colab not being flexible enough for multi-class projects, we switched to github for collaboration. One thing I would like to do better next time is use git or a better collaboration tool from the beginning so my teammates do not have to adapt mid-project. I learned a lot during this project about state transition based dialog agents from my project partners.
César: I wish I could have contributed way more for this project yet that was somehow limited regarding my background. I spent a few hours on the Data part of the report but only a little ended up being on the report because of size limitations. It has been an interesting project and I have learned a lot throughout my fellow group members' respective codes and ideas.
Ivan: The project was great in both coding and writing parts. It was quite hard to implement the dialogue state, but I have learned a lot about FSM. It was also interesting to compare the familiar keyword classifier with a machine learning system. I would like to thank Luca for his huge effort in putting the code together. Also, I wish it was more convenient to debug the code and add minor improvements. When the project was moved from Colab to GitHub, I could not change it directly anymore and had to explain my considerations in messages. Perhaps we could figure out a solution for collaborative debugging.
Stan: Due to some personal circumstances, I wasn't available for little over a week to help with the project. This is

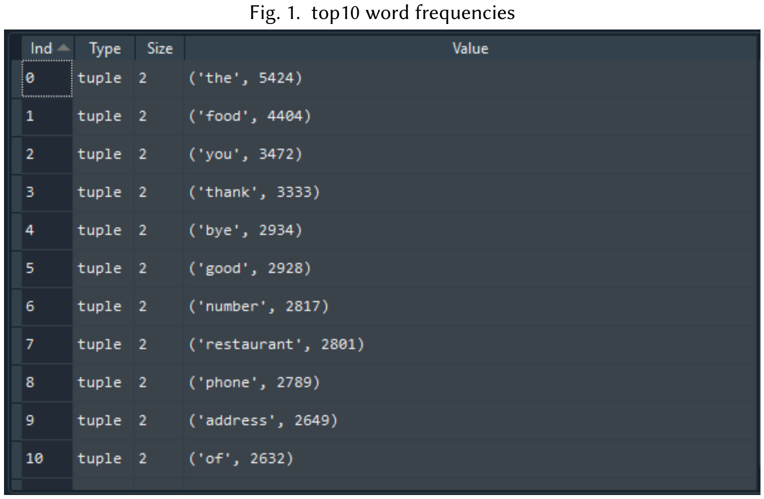
especially unfortunate since, with my background in linguistics and text mining, I could have had a bigger impact on the language related part. I would have also been able to learn more on the machine learning related stuff. I'm extremely glad that my group managed to take on the parts I wasn't able to complete. As for the content of the project, I think the most interesting was the degree of interdisciplinary tasks. It was fun to discuss the project from all of our backgrounds, to see what others thought was the best approach for the task at hand.

A APPENDIX

Dialog Act	Description	Example Sentence
ack	acknowledgment	okay um
affirm	positive confirmation	yes right
bye	greeting at the end of the dialog	see you good bye
confirm	check if given information confirms to query	is it in the center of town
deny	reject system suggestion	i dont want vietnamese food
hello	greeting at the start of the dialog	hi i want a restaurant
inform	state a preference or other information	im looking for a restaurant that serves seafood
negate	negation	no in any area
null	noise or utterance without content	cough
repeat	ask for repetition	can you repeat that
reqalts	request alternative suggestions	how about korean food
reqmore	request more suggestions	more
request	ask for information	what is the post code
restart	attempt to restart the dialog	okay start over
thankyou	express thanks	thank you good bye

Table 10. Dialog Acts Labels, from the course Methods in AI at the University of Utrecht

Fig. 1. top10 word frequencies



Ind	Type	Size	Value
0	tuple	2	('the', 5424)
1	tuple	2	('food', 4404)
2	tuple	2	('you', 3472)
3	tuple	2	('thank', 3333)
4	tuple	2	('bye', 2934)
5	tuple	2	('good', 2928)
6	tuple	2	('number', 2817)
7	tuple	2	('restaurant', 2801)
8	tuple	2	('phone', 2789)
9	tuple	2	('address', 2649)
10	tuple	2	('of', 2632)

id	antecedent	consequent	t/f	level	description
1	cheap,good food	busy	T	1	a cheap restaurant with good food is busy
2	spanish	long time	T	1	Spanish restaurants serve extensive dinners that take a long time to finish
3	busy	long time	T	2	you spend a long time in a busy restaurant (waiting for food)
4	long time	children	F	2	spending a long time is not advised when taking children
5	busy	romantic	F	2	a busy restaurant is not romantic
6	long time	romantic	T	2	spending a long time in a restaurant is romantic
7	bad hygiene, open kitchen	romantic	F	1	A restaurant where you can see inside the kitchen and where the kitchen is not clean is not romantic
8	bad food, bad hygiene	busy	F	1	Restaurants with bad quality and low standards are not busy
9	open kitchen	children	T	1	Restaurants with open kitchens and all you can eats are popular for families
10	long time, not open kitchen	boring	T	2	Restaurants which serve slowly and 'a la carte' tend to be boring
11	boring, expensive	busy	F	2	A la carte restaurants which are expensive and where you have to wait long for food are not very popular
12	boring	romantic	F	2	Boring restaurants are not a place to impress someone

Table 11. Complete list of implication rules for Reasoning

Table 12. Logistic Regression frequency of misclassifications on the left side, Neural Network frequencies of misclassification on the right

y_pred	**y_true**	**freq**	**y_pred**	**y_true**	**freq**
reqalts	null	5	null	inform	5
inform	null	14	null	request	3
inform	reqalts	9	confirm	inform	3
bye	affirm	1	null	reqalts	2
null	reqalts	2	reqalts	inform	8
request	inform	3	inform	reqalts	8
affirm	inform	2	confirm	deny	1
inform	hello	1	thankyou	bye	3
reqalts	inform	4	inform	affirm	1
thankyou	bye	5	request	ack	2
reqalts	ack	1	inform	negate	2
inform	confirm	3	reqalts	deny	1
null	hello	1	inform	null	2
null	reqmore	2	request	confirm	2
request	null	2	hello	inform	1
inform	affirm	3	bye	thankyou	2
confirm	reqalts	1	deny	inform	1
inform	ack	2	confirm	reqalts	2
thankyou	inform	1	inform	confirm	1
null	inform	2	inform	request	1
inform	request	3	reqalts	negate	1
confirm	request	2	affirm	null	1
inform	restart	1	null	affirm	1
ack	request	1	request	hello	1
inform	repeat	1	null	bye	1
null	ack	2			
hello	null	1			
confirm	null	1			