

# "What would you like to eat?": A Robust Restaurant Recommendation Dialogue System Using Machine Learning, Dialogue Management and Reasoning

DAVID PANTOPHLET, 7002025

NICK BEUKENKAMP, 2604930

GOYA VAN BOVEN, 6981844

SIMON VAN DE FLIERT, 5708656

This paper presents a robust dialogue system, that suggests a restaurant according to user's preferences. We compare four models for dialogue act classification: two baseline models, a logistic regression model and a random forest model. We find that the logistic regression model performs best (accuracy = 0.9739), and implement this model as default in our system. We further design a state transition diagram and build a dialogue manager that handles reasoning within the system. We include several configurations: friendly or terse conversations; and written only or also spoken text. The system can process user preferences, make recommendations accordingly, and provide additional information on the location and atmosphere of the restaurant when requested. In future work, we aim to test the different configurations in a user study. The system could further be improved by including more details on the restaurants and by adding a broader set of system responses.

Additional Key Words and Phrases: dialogue system, restaurant recommendation, reasoning, machine learning

## 1 INTRODUCTION

In this study, we present a robust dialogue system for restaurant recommendations using automated text classification, dialogue management and reasoning. The system aims to efficiently recommend the user restaurants that match their wishes in terms of cuisine, price range, area and atmosphere. This can help users to navigate through the many places to eat and find a restaurant that suits their preferences quickly and easily. As a basis for our system, we use dialogues from The Second Dialogue State Tracking Challenge [4].

This report is structured as follows. We describe the data in Section 2 and describe our text classification models in Section 3.1: we create two baseline models, a logistic regression model and a random forest model. After evaluating our models (Section 3.2) we find that the logistic regression model performs best. Next, we discuss the dialogue manager (Section 4) and the reasoning component (Section 5). As described in Section 5.1, we create two settings for our dialogue system: a friendly vs. terse mode, and text-only vs. text and speech mode. Finally, we discuss the performance and limitations of the system (Section 6).

## 2 DATA

We use the data from The Second Dialogue State Tracking Challenge [4] as data for our text classification models. This dataset, which contains 3235 dialogues, includes sentences of user-system interactions in the restaurant domain, where

---

Authors' addresses: David Pantophlet, 7002025, d.n.pantophlet@students.uu.nl; Nick Beukenkamp, 2604930, n.beukenkamp@students.uu.nl; Goya van Boven, 6981844, J.G.vanboven@students.uu.nl; Simon van de Fliert, 5708656, S.P.J.Fliert@students.uu.nl.

---

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.

© 2021 Association for Computing Machinery.

Manuscript submitted to ACM

Manuscript submitted to ACM

dialogue act	description	example sentence
ack	acknowledgment	<i>good</i>
affirm	positive confirmation	<i>yes, that is true</i>
bye	greeting at the end of the dialogue	<i>see you later</i>
confirm	check if given information confirms to query	<i>does this restaurant serve seafood?</i>
deny	reject system suggestion	<i>i dont want vietnamese food</i>
hello	greeting at the start of the dialogue	<i>hello</i>
inform	state a preference or other information	<i>im looking for a cheap restaurant</i>
negate	negation	<i>no seafood</i>
null	noise or utterance without content	<i>laughing</i>
repeat	ask for repetition	<i>go back</i>
reqalts	request alternative suggestions	<i>is there anything else?</i>
reqmore	request more suggestions	<i>more</i>
request	ask for information	<i>I would like the address and phone number</i>
restart	attempt to restart the dialogue	<i>restart</i>
thankyou	express thanks	<i>thank you</i>

Table 1. dialogue act categories in our dataset, with description and an example sentence per act

the system is a spoken dialogue system. In these interactions, the user aims to get a restaurant recommendation from the system that corresponds to their preferences. The sentences in the dataset are annotated for their *dialogue act*: actions that speakers perform by saying the utterance [1]. In total, our data includes 15 different dialogue acts, which are described in Table 1, with an example per act. A sample interaction from the dataset looks as follows:

1. User: i am looking for a cheap restaurant; speech act: inform(type=restaurant,pricerange=cheap)
2. System: What part of town do you have in mind?
3. User: and it should be in the west part of town; speech act: inform(area=west)

The dialogues were transcribed through automatic speech recognition. This influences the quality of the descriptions, as some words might be recognized incorrectly, and all punctuation marks are missing from the user utterances. Additionally, standard spelling is always used in the transcriptions, while in digital environments - e.g. dialogue systems - people often use many different spelling variations [2, 8]. We evaluate the impact of these differences between the data and the task in Section 3.2. For the recommendations, we use a data set consisting of information on 109 restaurants. Specifically, it contains the following details on each restaurant: the name, address, postal code, telephone number, food type, area, and price range. To implement the reasoning component (Section 5) we later added details on food quality, crowdedness, and length of stay to each restaurant in the dataset.

Our first task is to classify the dialogue acts of the sentences (Section 3). To get a greater insight into the distribution of the dialogue acts in the data, we plot the number of occurrences per act in Figure 1. We find the distribution is heavily skewed: inform is the most common act in the dataset with 10.160 occurrences, followed by request and thankyou (6.494 and 3.259 occurrences respectively), while none of the 9 least common acts occur more than 450 times.

### 3 MACHINE LEARNING

We create four models to automatically predict the dialogue act  $d$  that belongs to a given sentence  $t$ , in the form  $f(t) \rightarrow d$ . Examples of such a classification would be:

- *yes*  $\rightarrow$  affirm
- *what is the price range*  $\rightarrow$  request

We shuffle and split our data into an 85% train set and a 15% test set. We first create two baseline models: one that always predicts the majority class label (Section 3.1) and one based on keyword matching (Section 3.1). Next, we transform our

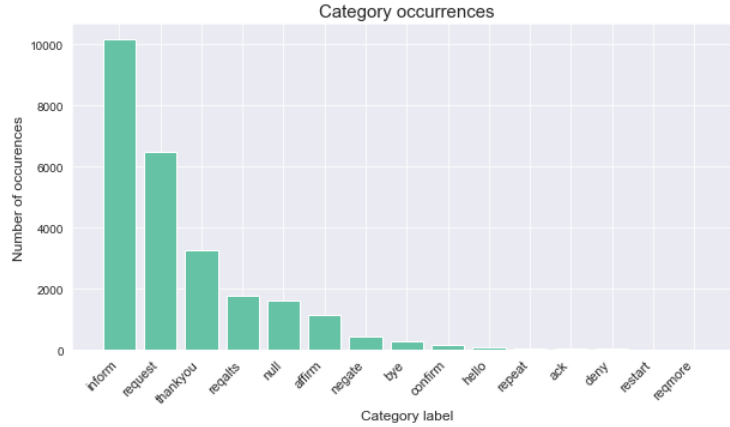


Fig. 1. Occurrences per dialog act in the data set

text data into sparse bag-of-words representations, in order to use our text data as input data for machine learning models. We first transform our training data, which results in vectors with a length of 711, and then transform our textual test data to represent them with the same mapping. Continuing, we create two machine learning models using Scikit-learn [6]: a logistic regression model and a random forest model. We evaluate the performance of all models in Section 3.2.

### 3.1 Model descriptions

**Majority Baseline.** The first model we build is a simple majority class algorithm, that predicts the same label for each sentence - the label that occurs most often in the dataset : `inform`. This model thus does not consider the content of the utterance, and is not expected to perform well, but therefore is a suitable baseline to compare our models to.

**Keyword Matching Baseline.** As a second baseline, we create a rule-based system based on keyword matching. We start by creating a list of keywords that are characteristic of each dialogue act. We identified keywords in two ways:

- Through a brainstorm about the words that are likely to be frequently used for each dialogue act;
- By investigating the words that co-occurred most often with each dialogue act in the data set.

The full list of keywords can be found in the Appendix (Table 4).

We then created an algorithm that classifies a given utterance by recognizing certain keywords in a sentence. The pseudocode for this algorithm can be found in Algorithm 1. This algorithm uses keywords to match test sentences to a label. The model predicts the label of a sentence in 2 parts:

- (1) If there are more keyword matches for dialogue act  $a$  than for any other act, the model predicts act  $a$
- (2) If there are multiple acts with the same highest number of keyword matches, the act with the highest number of occurrences in the dataset (Figure 1) is selected.

**Logistic Regression Model** We create a logistic regression model. Because our task is a multiclass classification problem, we use cross-entropy loss. Per sentence this model predicts the probability of each dialogue act and selects the label with the highest probability. The model further uses lbfgs optimization and l2 penalties.

**Random Forest model** The second machine learning model we build is a random forest model. This model creates multiple decision trees. When predicting the dialogue act for a given sentence, each tree makes a prediction, and the act that is predicted most often is the one that is assigned. We experiment with various maximum tree depths and find that using a depth of 20 gives a good performance in terms of accuracy, so we continue to use this depth.

**Algorithm 1** Pseudocode for the Keyword matching algorithm

---

```

157 labels ← [labels]
158 keywords ← [label: [keywords]]
159 train_pred ← []
160
161 for every sentence do
162   labels_counter ← [[label, counter]]
163   for every word in sentence do
164     for every keyword, label in keywords do
165       if word in keyword then
166         labels_counter[label] += 1
167       end if
168     end for
169   end for
170   max ← one label with highest counter
171   maxes ← all labels with equal highest counter
172   if len(maxes) == 1 then
173     train_pred.append(maxes[0])
174   else if len(maxes) == 0 then
175     train_pred.append("inform")
176   else
177     train_pred.append(highest ranked label)
178   end if
179 end for

```

---

	Accuracy	Macro F1	Precision	Recall
<i>Majority</i>	0.3761	0.042	0.0769	0.0289
<i>Keyword</i>	0.8204	0.5640	0.5552	0.6352
<i>Logistic</i>	<b>0.9739</b>	<b>0.8424</b>	<b>0.9028</b>	<b>0.8115</b>
<i>Random Forest</i>	0.9357	0.5985	0.6745	0.5653

Table 2. Model results for majority baseline model, keyword matching baseline model, logistic model and random forest model

**3.2 Model evaluation**

To evaluate our models we evaluate the accuracy, macro F1, precision and recall. We select macro rather than micro F1 because the distribution of the dialogue acts in our data is heavily skewed (Figure 1): the accuracy score therefore mostly reflects the performance for the common acts, while the less common acts remain underrepresented. Macro F1, precision and recall return the unweighted average of the performance on each dialogue act, providing better insight into the performance per act. Although this in turn overestimates the importance of some low-frequency acts which are less important for our task, the combination of accuracy and macro F1 gives a good overview of the performance for both high and less frequent acts. Table 1 shows the performance for each of our models. The logistic model performs best in terms of all metrics (F1=0.8424). Although the random forest model achieves a high accuracy score (0.9357), its F1 score is much lower (0.5985), indicating that the model likely performs better for labels that occur more frequently. Both our machine learning models outperform our baseline models in terms of accuracy and F1-score.

Additionally, we inspect the confusion matrix for the two machine learning models, to gain more insight into how well the models predict each dialogue act label. The heatmaps of the confusion matrices can be found in Figure 2a (logistic model) and Figure 2b (random forest model). We find that the logistic model has an overall good performance, classifying the majority part of the sentences correctly for all labels. The random forest model makes a lot more mistakes

as it seems to classify most sentences as inform. Most incorrect predictions of the logistic model are for act null (43 incorrect predictions). This problem is exasperated in the random forest model (92 incorrect predictions). Sentences with this label are particularly hard to classify for three reasons:

- (1) They often consist of unfinished words or sentences;
- (2) The variety of words they contain is very large;
- (3) Many words they contain also frequently occur in other dialogue act categories.

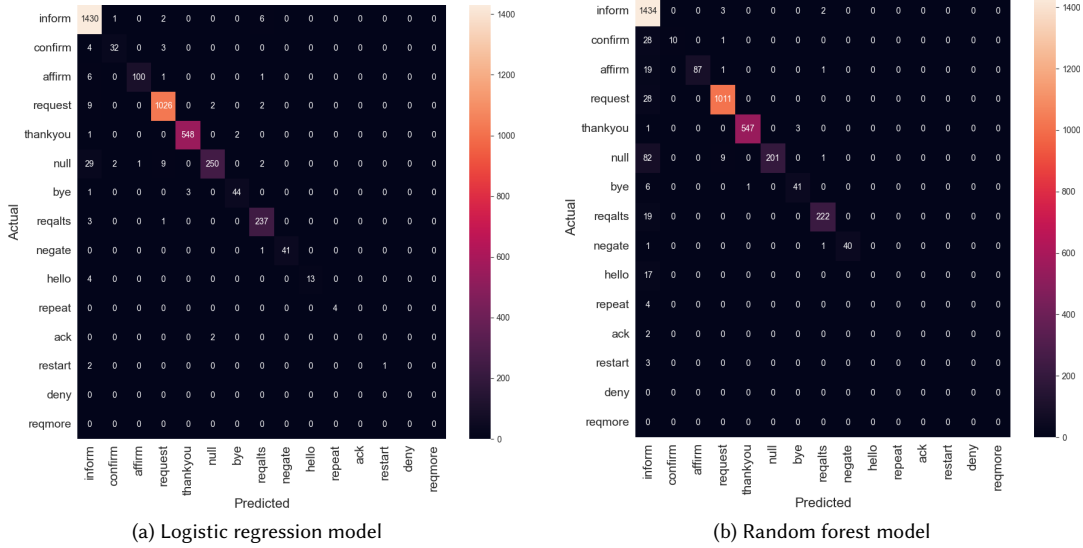


Fig. 2. Heatmaps of confusion matrices for (a) logistic regression model and (b) random forest model

When we compare the performance on utterances in our test set we find that a total of 71 utterances were misclassified by all models. Two sentence structures that appear to be particularly difficult to predict are questions such as:

- *is there a* [verb, independent clause, subject], because both the acts *inform* and *confirm* contain sentences of this structure;
- *okay, is it* [verb, independent clause, subject], as this is most commonly used for *confirm* but can interpreted to be *inform*. Both these dialogue acts depend on the rest of the sentence to provide the context, whether the user is conforming or informing about something.

The following structures we expected to be difficult for the models to classify:

- Incomplete words or sentences: *are there || the addre || arts town*. We find these utterances indeed got classified incorrectly: while *null* is the correct label, our classifiers either labeled them as *inform* or *confirm*. We believe this happens because the utterances usually form parts of complete utterances, for which the labels are *inform* or *confirm*. The classifiers do not notice that these utterances are incomplete, so they classify them like their most similar utterances.
- Instances that combine different dialogue acts: *hi i need a cheap restaurant in the south part of town || yeah im looking for food cheap restaurant*. This first utterance looks like a combination of the *hello* and the *inform* dialogue acts. Our classifiers predict *inform* for both sentences, while the first one is a combination of *hello* and *inform*, and the second sentence should be labelled as *affirm*. We believe that the issue for these types of utterances is that the majority of the utterance resemble the *inform* label, while the correct label usually can be

found at the start of the utterance. Our classifiers look at what these resemble the most, which in this case are the inform dialogue acts.

- Finally we compare whether spelling variations form a problem for the models, as the training data only contains standard spelling (Section 2). But spelling variations do not seem to form a problem for our models: these sentences are classified correctly by the machine learning models.

## 4 DIALOGUE MANAGER

### 4.1 State transition diagram

We analysed the dialogues in our data set (Section 2) to create a state transition diagram and noted several patterns that influenced our design choices. Firstly, we note that in each dialogue, the system requests the user to pronounce their preferences before it makes a suggestion. Continuing, in this process of identifying the preferences, the order in which these preferences are expressed does not matter, which we also identified as a valuable feature for our system. Another example that influenced our decision making was the usage of void comments, such as *uhh* and *um*. The system in the dataset sometimes attempts to identify such comments, but due to time constraints, we decided to ignore those comments and only listen to specific utterances that contribute to the process. We can ignore these void comments because they do not have a specific influence on any of the preferences of the user.

Based on this analysis we create a state transition diagram (Figure 3), which contains the structure that our dialogue system will follow during conversations. We implement this structure directly. We use the logistic regression model to recognize the dialogue act of each user utterance, as this model has the best performance (Section 3.2). Using this dialogue act and its current state, the system recognizes how it should update its state according to the dialogue structure. We will now elaborate more on this structure and the utterances that are expressed in each state.

The system opens the conversation with a welcome message to the user (State 1), stating the purpose of the system and the choices the user can make. In the friendly mode (see Section 5.1) this message looks as follows: Welcome! I hope you are having a nice day. Are you feeling hungry? If you let me know what and where you would like to eat and how much you are willing to spend, I can recommend you some nice places to eat.

The system thus asks for the (i) area, (ii) food type and (iii) price range preferences of the user (State 2), which the user can give in any order. These preferences are then recognized in utterances of the inform dialogue act as follows:

- We match keywords often used for expressing preferences. For instance from the sentence *I would like cheap Italian food tonight* the preferences (pricerange=cheap) and (food=Italian) are extracted, based on the keywords *cheap* and *Italian*. For several keywords we include multiple spelling variations (e.g., *centre* and *center*).
- If this does not yield any matches, we recognize common patterns, such as [ . . ] part of town, where the word left of this pattern usually indicates an area of preference. An example sentence using this pattern looks as follows: *I'd like to eat in the northern part of town*, where we extract (area=northern). Using Levenshtein distance we then map the extracted preference to its closest keyword, *north* in this case, where we take a maximum distance according to the word length: a distance of 4 if the words have 8+ characters, 3 for words between 5 and 7 characters, and 2 for shorter words.

After the system receives users input, it checks whether information is provided for all preference categories, and if so the system asks the user if they would like to add optional preferences (State 3), where the options are: romantic, busy, long stay or suitable for children. Note that this state is green in the figure to highlight it was only added later: after the reasoning part (Section 4) was implemented. Next, the system checks whether any restaurants match the

user's preferences. If there are none, the system returns a negative response (e.g. No matches) and asks the user to change their preferences (State 4). If a restaurant is found, it is suggested to the user, who can either accept or deny the suggestion (State 5). When giving the suggestion the system repeats the user's wishes and explains why this restaurant is a match (sentence 4 in the snippet below). An example dialogue of the system so far (after the welcome message, in the terse mode) looks as follows, where system internal shows the classifications the system makes internally:

```
1. User: Hi I'd like to eat some expensive Japanese food tonight, in the centre of town; System
internal: act= inform() preferences:(food=japanese, area=centre, pricerange=expensive)
2. System: OK, any other requirements? Choose from romantic, busy, suitable for children, and
places where you can stay long.
3. User: Stay long; System internal: additional preferences:(long)
4. System: I recommend Wagamama. It is an expensive Japanese restaurant in the centre of town. It
is a restaurant where you can stay long because it allows for long stays and the food is good and
the restaurant is expensive and it is usually nice and calm. OK?
5. User : yes please; System internal: act=affirm()
6. System: OK. The address is 36 saint andrews street. Do you want to know their phone number,
the postcode, or both?
```

If the user accepts the suggestion, the system gives the address of the restaurant. The user can then ask for the postal code, phone number or both. The system recognizes these request through recognizing keywords (e.g. *phone number* and *zip code*) in the user input. The responses given to the user are formed using a template, of which an example can be found in the dialogue below (sentence 3). When the requested information is missing, the response is formed through the following template: Sorry, we do not have [requested information] for [restaurant]. As requesting this information can be done in any order, we again combine these questions into one state (State 6). Once the user has received all requested information they can end the conversation by saying goodbye or giving thanks (State 7). An example of this final part of the conversation (in the friendly mode) looks as follows:

```
1. System: Great! The address is 36 saint andrews street. Would you like to know the phone number
or the postcode? Or maybe both?
2. User: Yes can I have both please?
3. System : The phone number is 01223 462354. The postal code is c.b 2.
4. User: Alright! Thank you
5. System : Enjoy your dinner
```

Finally, two user actions can be performed at any point during the dialogue. First, the user can request to restart the dialogue by stating *restart dialogue*, which brings the system back to State 1, where the welcome message is expressed. Second, the user can end the conversation by saying *goodbye*, bringing the system to State 7. These user actions are placed on the left of the diagram to not impede on the readability of the diagram, as they can be executed at any time.

## 4.2 Extra features

The dialogue system has two additional features to improve its usability. (i) When a user misspells a word or uses a different spelling variation than the one known to the system, the system asks for a confirmation whether the user meant the similar word the system has identified: I did not recognize weest. Did you mean west? Please reply yes (y) or no (n). (ii) To make the conversation more human-like [3], a delay of 0.5 seconds was added before the system's responses are displayed.



antecedent	consequent	description
calm, long stay, good food quality	romantic	There should not be any distraction in a romantic restaurant.
good food, cheap, long stay	busy	If a restaurant has good food and is cheap, many people will want to eat there.
short stay	children	Children usually have short concentration.
good food, expensive, calm	long stay	Expensive restaurant usually gives an extensive dining experience.
Spanish	long stay	Spanish restaurants usually serve a lot of food and therefore you tend to eat longer

Table 3. Consequences with corresponding antecedents and their descriptions. Antecedents are ordered from most important to least important for the corresponding consequent.

## 5 REASONING

To enrich the recommendation system we give the user the possibility to provide extra preferences. These preferences are *romantic*, *busy*, *suitable for children*, or places where you can *stay long*. The dialogue system will then pick restaurants that satisfy these wishes. We implement some reasoning capability in the dialogue system, by creating a set of antecedents that are used to determine which restaurants fulfill the additional preference of the user. Once the user expresses their extra preferences the system converts the user input into antecedents. Then the system uses these as rules to find the restaurant that satisfies most of the antecedents. If the system finds a suitable restaurant it presents this to the user where it also gives the reason why it chose that restaurant. Table 3 displays to rules available to the system.

Based on the rules above an example dialogue in the terse setting looks like this:

1. User: Hi I would like some cheap Chinese food please, in any area
2. System: OK, do you have any other requirements for your restaurant? You can choose from romantic, busy, suitable for children, and places where you can stay long
3. User: Yes, I'd like to eat with my children
4. System: I recommend you to go to the missing sock. It is a cheap Chinese restaurant in the south of town. It is a children restaurant because it allows for short stays. Would you like to go there?

We decided to add multiple antecedents to a consequent (see Table 3), because it prevents some conflicts. For example, a romantic restaurant is a place that is not busy and allows for long stays. However, if only one of these antecedents holds, the restaurant is both romantic and not romantic according to the assignment. To fix this conflict we created an ordering of importance of the antecedents, for each consequent, which can be found in Table 3. This way, we prevent conflicts as the most important antecedent decides.

A limitation of this method, however, is that e.g. a calm restaurant is always classified as being romantic, no matter the food quality or length of stay. To counter this, however, we first use all the antecedents and try to find a restaurant that matches all 3 values. Only if no matches are found, we eliminate the least important antecedent and search again. With this design, we retrieve the most romantic restaurants first.

### 5.1 Configurability

Finally, we create two configurations of the dialogue system. We aim to compare these different settings in a follow-up experiment. The configuration must be set before the dialogue is started. We have implemented the following settings:

- **Spoken vs. Written**, where the text is spoken out loud to the user before appearing in written form in the spoken setting, while in the written setting the text is not spoken out. The system's utterances remain the same between these two settings. We use the Pyttsx3 library to implement the spoken sentences.
- **Friendly vs. Terse**, which decides the tone of the dialogue system. Between these settings, the system uses different utterances: in the friendly setting, responses are longer, friendlier and more personal, while in the terse



setting the utterances are shorter and more to the point. For example, to the input 1. User: Hi I would like some cheap Italian food please, the system responds as follows in the friendly setting: 2. System: Great choice. So far I've found 2 restaurants that match your wishes. In what area would you like to eat?, while in the terse setting its response is: 2. System: 2 restaurant matches. What area?

## 6 DISCUSSION

In this study, we have introduced a robust dialogue system for restaurant recommendations. In the implementation of this system, we combine machine learning, reasoning and state management. The system asks for the preferences of the user and finds a restaurant that matches these requirements.

**Limitations** The system has several limitations. First of all, it is set up rigidly, forcing users to follow a specific structure and allowing only a small variety of textual input. While this issue is common in dialogue systems [9], it is exasperated in this study due to the time constraints of the project. The system can recognize a strictly limited set of patterns and keywords, and everything beyond that it does not understand. Importantly, the user does not know in advance which keywords are understood. This might lead to frustrating situations, e.g. if multiple user inputs are consecutively not recognized. A potential improvement would be to show the list of included keywords for the relevant attribute (e.g., price range) after the first failed attempt.

Secondly, the system gives standardized responses: each state has one fixed utterance that the system returns. This might cause users to get bored after a few conversations with the system. A way to counter this in future work could be to create a set of responses per state, one of which is randomly selected each time the system creates an utterance.

A third limitation is that we are currently unable to handle certain dialogue structures. For instance, we find the following dialogue in the dataset:

1. system: The phone number of grafton hotel restaurant is 01223 241387 .
2. user: type of food served
3. system: grafton hotel restaurant serves british food

Here, the user request information about the food type, after it has selected a restaurant and asked for their phone number. Currently, it is only possible to request address-related information after selecting a restaurant. Allowing the user to request a broader selection of information be a good addition in future work, but this was not implemented currently due to time constraints. Moreover, consider the following conversation with our system:

1. User: I'm looking for a restaurant in the north
2. System : 16 restaurants matches. What cuisine ?
3. User: swedish
4. System: 0 restaurant matches. What price range?

In this case, the system continues asking for preferences, even though the current preferences already do not correspond to any of the restaurants in the data set. This could be improved in two ways: (i) stop asking for preferences if the current preferences already do not yield any results; (ii) suggest an alternative preference for the second attribute (food type in this case) that does yield at least one option in combination with the first preference.

A fourth limitation is that the dialogue act classifier sometimes makes mistakes (Section 3.2). An example of such a mistake is that the logistic regression classifier classifies utterances such as *ok* as *inform* and *sure* as *null*, while these words are often used affirmatively in dialogues. The way we currently handle this problem is by hard coding these utterances to be recognized as *affirm*, but this is not an ideal solution. We find the source of these problems in our

data: both words are not used in isolation as a user response in the data, while users will likely respond this way when interacting with the system.

Fifth, the system only knows basic facts about the restaurant. Many existing platforms used for finding restaurants (e.g., Google Maps or Yelp) have access to much more data, so even though our system quickly finds a restaurant corresponding to the user's wishes, it might still not be a sufficiently attractive alternative. The system could therefore be enhanced by including restaurant ratings, images, website links and opening hours. Additionally, including an option to directly compare the data of two restaurants (e.g., *[Restaurant A] is low quality and cheap while [Restaurant B] is high quality but expensive*) could improve the usefulness of the system.

Finally, some limitations hold for any dialogue system. According to [9] they often fail to notice nuances and sometimes feel awkward, as they regularly fail to quickly jump between states. While humans often interrupt each other to bring their point across, systems lack this ability. Therefore, systems are more reliant on perfect decisions, and mistakes can cause frustration. Our system is not different in this respect and suffers from these limitations.

**Alternative approaches** In this work, we combine machine learning with reasoning and dialogue management. An alternative is a neural-based end-to-end approach (such as e.g. [10]) where the entire system is based on machine learning. However, this causes a system to be less interpretable [5], and requires more complex models that are more expensive, slower to train, and which have a much greater environmental impact [7].

Using our current set-up, we could have experimented with other machine learning methods for dialogue act classification, such as Neural Networks or Naive Bayes classifiers, but as logistic regression already showed a very high performance (accuracy = 0.9739) we continued using this model.

**Future work** In future work, we firstly aim to compare our different settings (spoken vs. written; friendly vs. terse, Section 5.1) in a user study. Additionally, the system could be improved by handling the limitations described above.

## 7 OVERVIEW OF CONTRIBUTIONS OF INDIVIDUAL GROUP MEMBERS

**Reflection:** At the start of the project, we came together, discussed our strengths and weaknesses and managed each other's expectations. We unanimously agreed to meet all the feedback deadlines and not to wait until the last possible moment to complete our work. Each week we had multiple moments we discussed what we needed to do and during these moments we split the work. Dividing the workload went smoothly, where each member was free to state their weaknesses and preferences. Each member completed their agreed section in time, and each member was more than willing to jump in to help when help was requested. It was overall a pleasant and respectful process.

Overview of individual contributions		
Task?	Who?	How much time?
Baseline classifiers	David, Nick	16
ML classifiers	Goya	6
State diagram	Simon	8
Dialogue System Code	All	120
Extract meaning Code	Goya	16
Bug fixing	All	40
Report	All	60

## REFERENCES

- [1] John Langshaw Austin. 1975. *How to do things with words*. Oxford university press.

- [2] Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 conference of the North American Chapter of the association for computational linguistics: Human language technologies*. 359–369.
- [3] Ulrich Gnewuch, Stefan Morana, Marc TP Adam, and Alexander Maedche. 2018. Faster is not always better: understanding the effect of dynamic response delays in human-chatbot interaction. In *26th European Conference on Information Systems: Beyond Digitization-Facets of Socio-Technical Change, ECIS 2018, Portsmouth, UK, June 23-28, 2018. Ed.: U. Frank*. 143975.
- [4] 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.
- [5] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu. com.
- [6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
- [7] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).
- [8] Rachael Tatman. 2015. # go awn: Sociophonetic variation in variant spellings on Twitter. *Working Papers of the Linguistics Circle* 25, 2 (2015), 97–108.
- [9] Nigel G Ward and David DeVault. 2015. Ten challenges in highly-interactive dialog system. In *2015 AAAI Spring Symposium Series*.
- [10] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* (2016).

## .1 Keywords

dialogue act	Keywords
inform	know, food, restaurant, town, part
request	number, phone, address, whats, code, post
thankyou	thanks, appreciate, thank
reqalts	else, anything
null	ah, uh, ugh, oh, noise, sil, unintelligible
affirm	yes, true, correct, affirmative, uh-huh, agree, acknowledge, concede, right, yea, yeah
bye	goodbye, bye, cya, farewell, later
confirm	serve, priced, center
hello	hi, hey, hello, greetings
negate	not, none
deny	dont, wont, no, negative, false, disagree, reject, wrong, want, incorrect
repeat	repeat, recur, echo, back
ack	okay, alright, ok, kay, um
restart	start, reset, restart
reqmore	more, other

Table 4

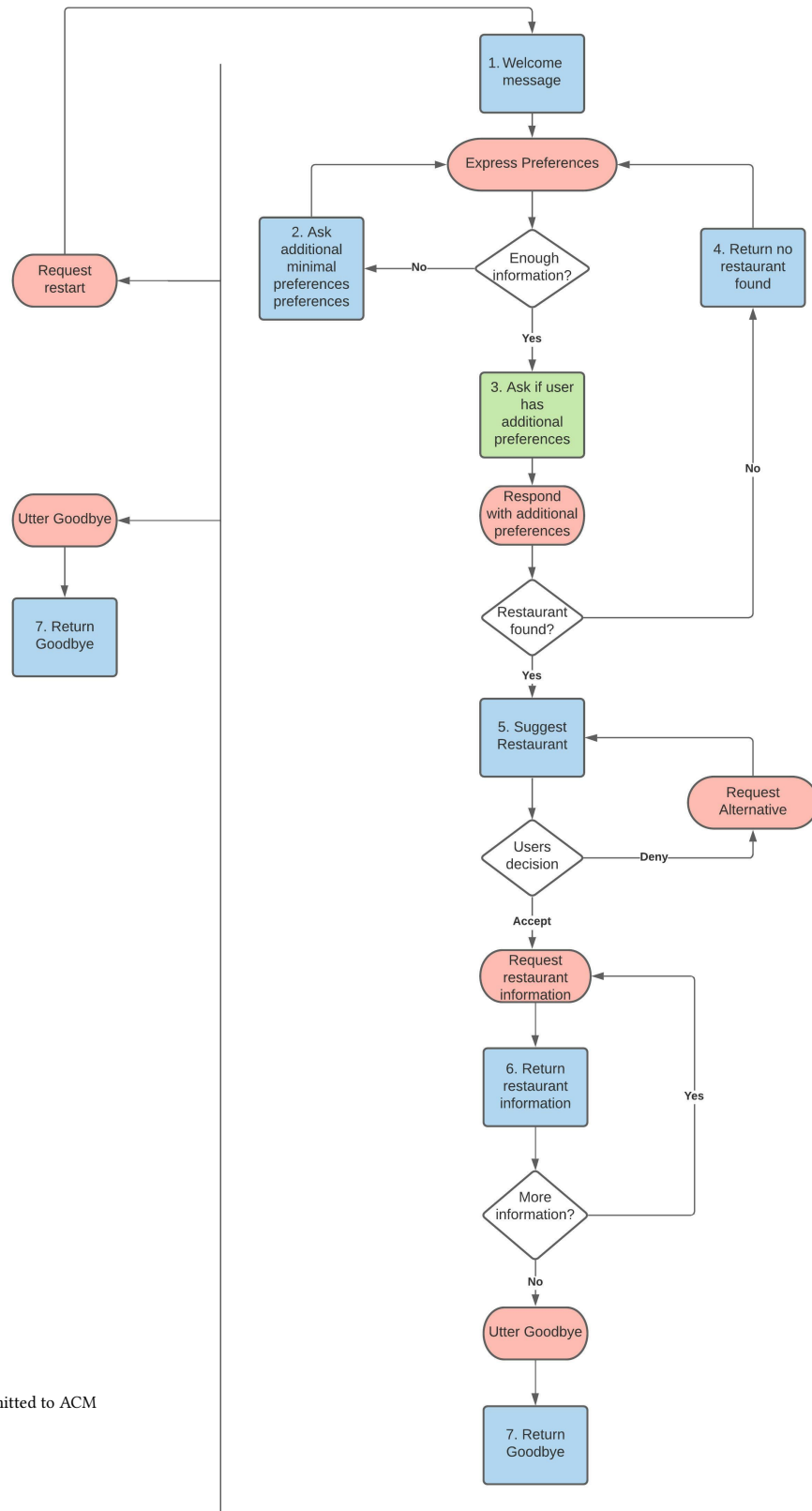


Fig. 3. State transition diagram used as basis for the dialogue system