



TASTY CHATS: AN INTELLIGENT RESTAURANT RECOMMENDATION SYSTEM

SEPTEMBER 27, 2023

Methods in AI Research Report Part 1 Group 28

Authors:

Samir Mengari, 6157106

Giorgio Momigliano, 5235545

Tijmen Oliehoek, 2140276

Yannick den Boer, 6803113

Sarah Abdalla, 4696182

Group 28

Contents

1	Introduction	4
2	Data	4
2.1	Description of the dataset	4
2.2	Dialogue state transition diagram	5
2.2.1	Example dialogue snippets	5
3	Machine Learning	7
3.1	Data Preparation	7
3.2	Baselines	7
3.2.1	Majority class assignment baseline	7
3.2.2	Keyword matching rule-based baseline	7
3.3	Machine learning classifiers	8
3.3.1	Logistic Regression	8
3.3.2	Feed-forward Neural Network	8
3.3.3	Decision Tree	8
3.4	Evaluation	8
3.5	Error Analysis and Difficult Cases	9
4	Dialogue Manager	10
5	Reasoning	11
6	Configurability	12
7	Conclusion	13
7.1	Discussion of the approach for the dialogue system in general	13
7.1.1	General overview	13
7.1.2	Limitations	13
7.1.3	Possible improvements	13
7.1.4	Alternative approaches	14
7.2	Discussion of the approach for dialogue act classification	14
7.2.1	General overview of the experimental methodology	14
7.2.2	Limitations	14
7.2.3	Possible improvements	14
7.2.4	Choice of machine learning algorithm and alternative approaches	14
8	Overview of contributions of individual group members	16
9	Appendix	17
9.1	Evaluation Tables	17
9.1.1	Baseline Majority	17
9.1.2	De-duplicated Baseline Majority	17
9.1.3	Baseline Rulebased	17
9.1.4	De-duplicated Rulebased	18
9.1.5	De-duplicated Feedforward Neural Network	18
9.1.6	Logistic Regression	19
9.1.7	De-duplicated Logistic Regression	19
9.1.8	Decision Tree	20
9.1.9	De-duplicated Decision Tree	20

Abstract

Recommender systems are becoming increasingly popular in the age of excessive information flow. These systems are efficient tools of filtering online information. Building application-focused recommender systems presents a challenge in the selection of techniques due to the diversity of available methods. Furthermore, each technique carries its unique features, benefits, and drawbacks, adding complexity to the decision-making process and giving rise to the need for extensive evaluation and analysis. This report explores a dialogue system designed for restaurant recommendations. The system classifies user inputs into dialogue acts, offering personalized suggestions. We analyze data preparation, baselines, and two machine learning techniques, with a feed-forward neural network as the top performer.

1 Introduction

This report provides an analysis of our dialogue system, a rather small platform designed to facilitate efficient interactions between users and the system. The dialogue system serves as a translation interface for user inputs, effectively processing and responding to queries and requests. The underlying framework of our system relies on dialogue act classification models, which are crucial in categorizing user utterances into distinct dialogue acts. The foundation of any dialogue system is the data it operates upon. In our case, the system works with the dataset "Restaurant Attributes". This dataset comprises information on 109 restaurants, containing various information on these restaurants, expressed in English. The system's operation is guided by a designed state transition diagram we designed, which interprets the input and generates the output, resulting in the desired dialogue and result. The system progresses through different states, each triggered by user input, with the transition diagram initially favoring system-controlled dialogue initiative. Goal-directed questions are strategically asked by the system to narrow down user responses, facilitating the classification of user preferences. Once the system has collected the requisite information, it can offer restaurant suggestions, allowing users to control the dialogue initiative. Section 2 discusses all the data preparation. To get the best potential outcome from the input data, thorough pre-processing is needed. We engage in data cleaning, consisting of tokenization and text consistency measures, to ensure that our data is useful for analysis and classification. This is followed by the exploration of two baseline models: for enhancing the performance of our dialogue system, we used machine learning techniques, discussed and evaluated in section 3. A dialogue manager is created and evaluated in section 4. A reasoning component is integrated into the dialogue manager, as elaborated section 5, enabling the derivation of further restaurant attributes from the original properties, such as price range and food quality. Lastly, we have incorporated configurability options in section 6, allowing users to adjust settings like caps-lock, typo checking, and Levenshtein distance, enhancing the user experience. This report will provide an understanding of our dialogue system's design and implementation process, providing an understanding of its capabilities.

2 Data

2.1 Description of the dataset

The dataset consists of 3235 dialogues expressed in English, each representing interactions between a user and a system in a restaurant recommendation domain. The user is looking

for a restaurant that matches their specific preferences based on a given task, including price range, area, and food type, or they could choose not to have any preferences, while the system offers corresponding restaurant recommendations. The used format in the dataset is `dialog_act [space] utterance_content` involving the occurrence of fifteen different dialogue acts, see Table 1: [Dialogue Acts and Examples](#). The dataset originates from phone conversations, thus containing instances of noise and unknown utterances, such as coughing, "um" instances, or solely parts of sentences. The system usually identifies these occurrences as noise under the dialogue act "null," signifying that no specific action is required from the system, and the conversation can continue. However, the formality of the language used in the dialogues varies, with some being more informal than others. This can sometimes cause the system to be unsure about what dialogue act the user actually performs, as the user may treat the system more like a person than a machine. This impacts the system's performance because the resulting confusion can occasionally prevent the system from fully comprehending the user's task, thus hindering its ability to provide a relevant restaurant recommendation.

dialogue Act	Description	Example Sentence
ack	Acknowledgment	Okay um
affirm	Positive Confirmation	Yes right
bye	Greeting at the End	See you, good bye
confirm	Check Confirmation	Is it in the center of town?
deny	Reject System Suggestion	I don't want Vietnamese food
hello	Greeting at the Start	Hi, I want a restaurant
inform	State Preference	I'm looking for a restaurant that serves seafood
negate	Negation	No, in any area
null	Noise or Utterance	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 postal code?
restart	Attempt to Restart	Okay, start over
thankyou	Express Thanks	Thank you, good bye

Table 1: Dialogue Acts and Examples

2.2 Dialogue state transition diagram

The system consists of several dialogue states that were modeled in the state transition diagram as depicted in Figure 1. The different states of the system are depicted by the boxes. After each state, user input as depicted by the rounded boxes is required for the system to move to its next state. The transition diagram is modeled deliberately such that the dialogue initiative is primarily being controlled by the system until a suggestion for a restaurant has been made. By using goal-directed questions we are able to narrow down the diversity in user responses, allowing the system to classify the user preferences more easily. After the system has collected the required information it is able to make a suggestion for a restaurant. After this dialogue state, the dialogue system becomes more reactive and the user is prompted to control the dialogue initiative. This allows the user to request additional information or clarification about the suggested restaurant. When the user input is classified as a 'bye' dialogue act, the system will reach its exit state 'Goodbye'.

2.2.1 Example dialogue snippets

The example dialogue snippet down below represents the "reqalts" dialogue act, which is state 7, "Update preferences, in the dialogue state transition diagram. A certain restaurant was already suggested based on the given preferences, however the user wanted a different food type and the system updated the user's preferences.

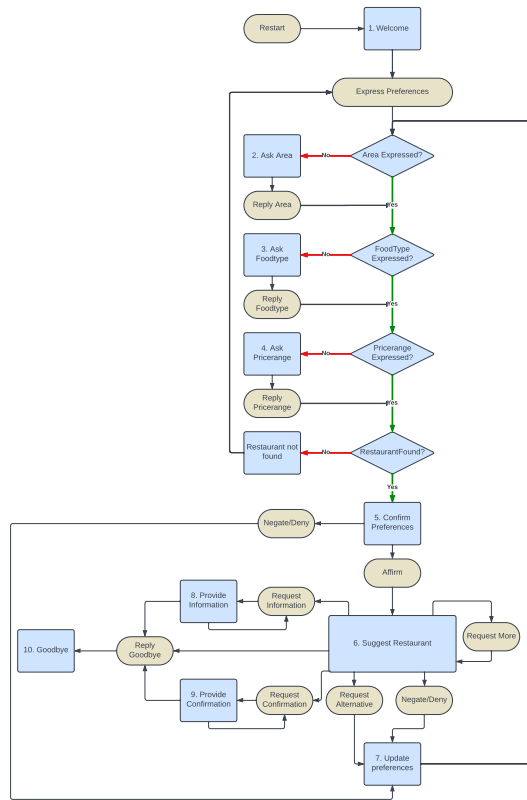


Figure 1: dialogue state transition diagram

Please confirm the following (yes/no):

Price range: cheap

Area: centre

Food: any

> yes

Here's a suggestion: mahal of cambridge!

It is priced 'cheap', in the centre of town. It serves indian food. You can ask for its address, phone number, or postcode.

(if you want to check for additional requirements (e.g. romantic, children, touristic, assigned seats), say 'additional requirements')

> wait what about polynesian food?

Please confirm the following (yes/no):

Price range: cheap

Area: centre

Food: polynesian

> yes

Here's a suggestion: golden house!

It is priced 'cheap', in the centre of town. It serves polynesian food. You can ask for its address,

Down below is another example dialogue snippet which represents the "request" dialogue

act, which is state 8, "Provide information" in the dialogue state transition diagram. The system suggested a restaurant, and upon the request from the user to receive the phone number the system returns the corresponding phone number.

Here's a suggestion: mahal of cambridge!

It is priced 'cheap', in the centre of town. It serves indian food. You can ask for its address, phone number (if you want to check for additional requirements (e.g. romantic, children, touristic, assigned seats)).

> can i have the phone number

For 'mahal of cambridge':

- the phone number is 01223 360409

3 Machine Learning

3.1 Data Preparation

To adequately make use of our data we need to process it. The data consists of pairs of sentences and dialogue acts. After this, the data is divided into a training set, a development set and a test set. The test set consists of 15% of the data. The remaining 85% of the data is divided once more into a new training set consisting 85% of the remaining data and a development set consisting of the 15% of the remaining data. We end up with 15% test-, 72.25% training- and 12.75% development data. In the data, there were a lot of duplicate act-sentence pairs (over 80%!) — because of this, simply splitting the dataset leads to a lot of leakage, since pairs that we think we're limiting to the test set are actually likely duplicated in the training set as well. To remedy this, we tested all the classifiers twice: first with a naive train/dev/test split, then removing all duplicates from the data before splitting. This enables us to accurately evaluate the performance of all the systems.

3.2 Baselines

For both baselines, and to get a general feel for the data, we performed some basic analysis. This fed into how our keyword-matching baseline picked its keywords. We also wanted to see the distribution of the acts across the dataset (see Figure 2). We will now explain how we made each baseline and each classifier, and evaluate their performance collectively in section 3.4.

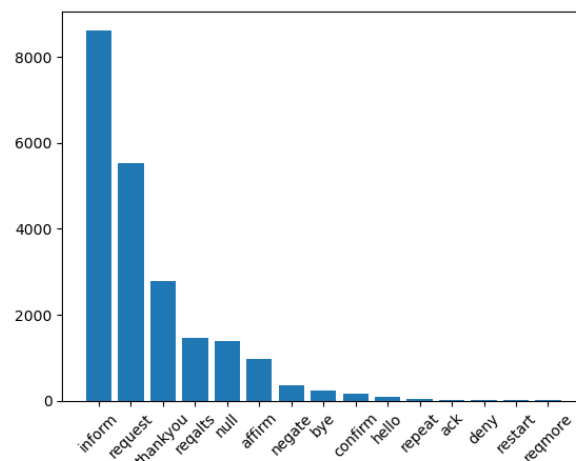
3.2.1 Majority class assignment baseline

The first baseline used is the majority-class baseline. This baseline takes the most common dialogue act and applies it to any sentence given. In our dataset this is the 'inform' act with a 40% occurrence rate.

3.2.2 Keyword matching rule-based baseline

The second baseline used is the keyword matching rule-based baseline. This works using certain keywords in the sentences and basing the dialogue act on the keywords found. This is done by finding common words in the dataset that most identify certain acts (if something identifies both "inform" and "request" a lot of the time, it wouldn't be a very good rule since we assign them one-to-one). These words are then put in a list, and the baseline then classifies sentences if they contain one of these words (or more accurately, regexes, to enable/partial-word matching). This baseline gets more accurate as we include more keywords. Additionally, a default is included for when no rules match — the majority act makes sense

Figure 2: Act distribution



here (i.e. "inform"). Using 5 keywords, we get 80% accuracy; with 13 keywords, 91%; and with 23 rules, we get a final accuracy of 93%.

3.3 Machine learning classifiers

3.3.1 Logistic Regression

For our first classifier, we used multinomial logistic regression to classify sentences with their corresponding dialogue acts. The training sentences are tokenized before the model attempts to fit them against their corresponding target acts by finding the optimal weights and biases. We tried tuning some of the hyperparameters for regression, like the type of penalty used, and different solvers, but none had a significant impact on final performance. Therefore, we stuck to sklearn's Logistic Regression (see reference section) defaults (L2 penalty, an 'lbfgs' solver, and no class weightings).

3.3.2 Feed-forward Neural Network

For our next classifier, we used a feed-forward neural network with a bag of words representation. This representation was set by the lab, and it makes sense as the utterances are largely quite simple — the ordering of their words doesn't really matter. Also, given the limit vocabulary size, it could be implemented efficiently. The network we ended up using was quite simple: it used 2 hidden layers of 128 units each, ReLU activations, a batch size of 5, running for 2 epochs (with duplicated data) and 8 epochs (without duplicate data). Changing these hyperparameters also didn't lead to major changes in performance: more layers, more hidden units, different activation functions, dropout layers. Nothing noticeably changed, other than much longer training times, so we decided to keep the final model simple.

3.3.3 Decision Tree

Initially, we were happy just implementing the 2 ML classifiers above. But when we looked at the final evaluation metrics for the above classifiers (see section 3.4), especially with the de-duplicated data, we wondered whether a Decision Tree might fare better. As we'll see, it largely didn't, but it was worth implementing given sklearn's defaults. We also tried tuning a few parameters here, but again, without any real gains in performance.

3.4 Evaluation

Initially, we largely looked at accuracy as the primary metric to judge each of our classifiers — though often useful as a quick check, especially when tuning neural network hyperparameters, accuracy isn't enough to paint a full picture. Therefore, we also calculated precision and recall (and subsequently, different F1 scores), to better understand how each classifier was performing. We ran the evaluation metrics 10 times for each classifier, and overall scores stayed roughly within 2% (we will mention exceptions to this below). Figure 3 includes, for each classifier, accuracy, weighted F1 scores (weighted by act distribution) and unweighted F1 scores, all as an average across the 10 runs.

As expected, most of the de-duplicated versions of each classifier scored worse than their duplicate-including counterparts. Out of around 25000 act-sentence pairs, only around 5000 were actually unique, meaning we were training on essentially the entire dataset, even after train/dev/test splitting. Interestingly, the de-duplicated baseline-majority classifier actually performed better on all metrics, but this doesn't mean too much — just that there were less duplicated "inform" acts than other acts, and so formed a larger share of the de-duplicated data. We will now just talk about the de-duplicated versions of each classifier, as they're the ones that avoided test-set contamination.

Overall, the 3 machine learning models all perform similarly in terms of accuracy and

Classifier	Accuracy:	Weighted F1:	Macro F1:
Baseline Majority	0.40	0.23	0.04
Deduped Baseline Majority	0.61	0.46	0.06
Baseline Rule Based	0.93	0.93	0.63
Deduped Baseline Rule Based	0.81	0.78	0.54
FeedForward NN	0.98	0.98	0.82
Deduped FeedForward NN	0.93	0.93	0.75
Logistic Regression Model	0.94	0.94	0.87
Deduped Logistic Regression Model	0.90	0.89	0.60
Decision Tree	0.97	0.97	0.88
Deduped Decision Tree	0.88	0.87	0.70

Figure 3: Evaluation metrics on all classifiers. Best de-duplicated performance in bold.

weighted F1. Having run each evaluation round multiple times, these figures only varied by a 1-2% percentage points max., so we can say that the neural network has *slightly* better accuracy, Weighted F1, and Macro-F1 compared to the others. This wasn't the case every time though: the Macro-F1 figure for the Decision Tree is an average over values ranging from 0.63 (minimum) to 0.80 (maximum). It had much higher variance than the others. This is due to the severely-underrepresented classes in the dataset: some acts like "reqmore" and "restart", especially post de-duplication, had so few examples that the regression and neural network just didn't learn them effectively (it's also possible they just weren't in the training data at all, and their lone examples were in the test set). The decision tree, on the other hand, would give them more importance even if seen very few times. Looking at the original dataset, some sentences appear across a lot of conversations, but only once. They are still important, but especially post de-duplication, we can struggle to classify them correctly.

The rule-based baseline has an acceptable accuracy when de-duplicated (around 81%), due to the way it was designed, performs badly on the macro-F1, with an average score of 54%. As you can see in Appendix 9.1.2, many classes have 0 accuracy — these are ones for which no rule existed. Still, it's worth noting how well a simple system of keyword-matching can perform, and highlights how the conversations these types of systems have tend to be simple. We can also see this directly in the example dialogues given, with short sentences, and little variation. It's in the tail-end of inputs that we struggle (as shown by even much larger neural networks struggling to surpass 93% accuracy).

Overall, as measured by the metrics we chose, the Feed-forward neural network performs best, and therefore is the system we will use in the dialogue system. It still has issues though, and we will have to design the dialogue system with this in mind — sometimes we will misclassify, so we have to be able to give the user plenty of opportunities to fix errors in the system state.

3.5 Error Analysis and Difficult Cases

There were a few challenges in accurate classifications. These error-prone scenarios include:

Underrepresented acts Some dialogue acts occur far less frequently in our dataset than others. The scarcity of these instances poses a challenge for our model. dialogue acts

like *'deny'* and *'reqmore'*, being infrequent, may result in inadequate training data for the model to comprehend these nuances.

Complex acts Some dialogue acts require a deeper understanding of the context in which the words are used. Also, words can have subtle distinctions in user intent, which can be challenging for the model to grasp. For instance, distinguishing between *'request'* and *'reqalts'* may pose a complex situation when a user seeks more information, but the system interprets it as an exploration of alternative choices.

Ambiguity in utterances Certain user utterances inherently possess ambiguity, making it hard to grasp the intended dialogue act. For instance: *"Can you tell me more?"*. The ambiguity in user input, leaves room for multiple interpretations. They could be categorized as either a *'request'* for more details or a *'reqalts'* expressing an interest in exploring alternative choices. Accurate classification depends on the models' ability to discern contextual differences.

Negation in utterances Utterances featuring negation, such as *"I don't want Italian food"* or *"I'm not interested in expensive places"* pose a peculiar challenge. The negation reverses the intended meaning of the phrase, requiring the model to correctly detect the intended negation (true negative). By accurately detecting negation cues in an utterance, these cues can be used for correctly classifying the dialogue acts.

4 Dialogue Manager

Using one of the implemented machine learning models. The system is able to take utterance as input and find a matching response. This is done by the state transition function, which takes in the current utterance and the current dialogue state. Based on the utterance and state, the dialogue transition function classifies the utterance's act and proceeds to either request missing info or respond to the user's utterance. When the user makes a typing error and the word is not recognised, the system will calculate the Levenshtein distance to possible keywords. If the Levenshtein distance between the misspelled word and the keyword is less than three. The system will assume the corresponding keyword as the intended word.

As described in section 2.2, the dialogue initiative plays a crucial role in this. When the dialogue manager has welcomed the user in state 1 and is in a state prior to making a restaurant suggestion (e.g. states 2 to 4), the dialogue initiative is controlled by the system. In these states the system will ask questions to narrow down the user's preferences. Examples of these questions are *"What is your price range (cheap, moderate, expensive, or no preference)?"* and *"What area would you like to eat in (north, east, south, west, or no preference)?"*. The system suggests possible answers to guide the user through the conversation, but the user is able to input more sophisticated utterances which can be recognised by the system. As seen in the examples, the user is also able to opt for having no preference. In this case, all possible options for this category are considered by the system. When all preferences are known to the system it will try to make a suggestion in state 6. If there are no restaurants that match the user's criteria, the system will respond with *"Sorry, there's no suggestion(s) given your requirements. Please try something else."*. The system will remove all saved preferences and returns to after state 1.

After confirming all preferences in state 5 and making a suggestion for a restaurant in state 6, the system will enter a set of more idle states in which the user can ask for additional information (states 6 to 9) For example, the user can ask *"Can I have the phone number?"*. A possible response by the system could be *"For 'saint johns chop house': - the phone number is 01223 353110"*. The user can also ask for additional requirements by typing *'additional requirements'*. When the user for example asks for *"assigned seats"* the system may suggest a new restaurant and respond with *"Its crowdedness is usually 'busy', the usual length of stay is 'short stay', and the food quality is 'good food'. It's classified as 'assigned seats' because in a busy restaurant the waiter decides where you sit."*.

When the user is finished, they can tell the system goodbye. The system will know the final state (state 10) is reached and says: *"Goodbye! Thanks for using our restaurant recommender."*

5 Reasoning

A reasoning component is integrated into the dialogue manager by utilizing multiple inference rules to derive further restaurant attributes from the initial properties. These initial properties are the price range, food quality, food type, length of stay (short or long stay), and crowdedness (quiet or busy). The properties food quality, length of stay and crowdedness were initially not a part of the original data set, these properties have been added manually to the restaurants in the dataset with a focus on achieving as much distribution as possible. The purpose of this distribution is to higher the possibility of satisfying the user's preferences, to avoid the scenario where no restaurants align with the user's preferences as much as possible.

The reasoning component allows users to request additional requirements, which is possible after the user has provided their primary preferences: price range, area and food type. Based on these primary preferences, the dialogue manager creates a set of suggested restaurants. The user has the option to choose from four different additional requirements: romantic, children, assigned seats, and touristic. Whether an additional requirement, i.e. the consequent, can be applied depends on whether the antecedent (the properties of the restaurant) actually leads to this particular consequent according to the inference rules. Each additional requirement has it's own set of inference rules which will be discussed below. The inference rule is applied to the set of suggested restaurants, and the restaurant(s) that return *True* are the ones where the additional requirement is possible. When none of the restaurants from the set of suggestions based on the primary preferences include the correct antecedent(s), no restaurant is suggested as no restaurants match this additional requirement and *False* is returned.

The first additional requirement is 'romantic' and is applicable on restaurants that offer a long stay, or/and when the 'crowdedness' property of the restaurant is quiet and so the inference rules will return *True* for these. However, when it is the opposite, i.e. 'length of stay' is 'short stay', and/or the 'crowdedness' is 'busy', the inference rules will return this as *False*. See the scenario down below for an example of the scenario where a restaurant is romantic because it offers a long stay.

```
> romantic
Here's a suggestion: cambridge lodge restaurant!
It is priced 'expensive', in the west of town. It serves european food.
Its crowdedness is usually 'quiet', the usual length of stay is 'long stay', and the food
quality is 'good food'.
It's classified as 'romantic' because spending a long time in a restaurant is romantic, and a
quiet restaurant is romantic.
```

A possible contradiction for the romantic requirement arises when a certain restaurant includes solely one of the valid properties, but excludes the other. More explicitly: when a restaurant is set 'busy', but the length of stay allowed in the restaurant is long, or vice versa. Note that this restaurant will not be considered romantic when following the inference rules because the inference rule for one of the property returns *False*, see an example down below:

```
Here's a suggestion: nandos!
It is priced 'cheap', in the south of town. It serves portuguese food.
You can ask for its address, phone number, or postcode.
(if you want to check for additional requirements (e.g. romantic, children, touristic,
```

```

assigned seats), say 'additional requirements')
> additional requirements
Please specify your additional requirement (children, touristic, romantic,
assigned seats):
> romantic
Sorry, there are no suggestions given your additional requirements.

```

The restaurant 'Nandos' offers the possibility of a long stay, however because it is set as 'busy' for its 'crowdedness' in the CSV file, the restaurant is not considered romantic. While only one property is needed for a restaurant to be considered, there are two possible properties for a restaurant to be considered: for its 'crowdedness' and for its 'length of stay'. When one of these returns *True*, but the other one returns *False*, the final return is *False*.

Secondly, the additional requirement 'children' is requested in case the user plans on taking children with them to the restaurant. When the user requests for this, the suggested restaurant should have 'long stay' for its 'length of stay' property in order for that restaurant to be suggested. In this case, the antecedent is 'long stay', and the consequent is 'children', and the inference rule will return *True* for this. On the contrary, when the antecedent is 'short stay' for the property 'length of stay' the inference rules will return *False*.

Thirdly, the additional requirement 'touristic' has as its antecedent cheap for its price range property AND good food for its food quality property to lead to the inference rules returning *True*. In the case of the antecedents existing of 'expensive' or 'moderate' for 'price range', and/or 'high quality' for 'food quality', the inference rule will return this restaurant as *False*. Additionally, in the case of the 'food type' property being 'Romanian', the inference rule will also return *False*, as Romanian food is usually unknown for tourists, and so is not the typical touristic food type.

Lastly, the additional requirement 'assigned seats' requires the antecedent 'busy' for the property 'crowdedness' in order for the inference rules to return *True*. When the property 'crowdedness' entails 'quiet', the inference rules will return *False*.

6 Configurability

We have chosen to implement a few configurability options. These can be set from the command line before the system starts and are also accessible with the -config command during the dialogue. This will redirect them to a separate section where the system will ask if they want to adjust any settings;

```

Current settings:
capslock:      False
typochecker:    False
debug:         True
system delay:  0
levenshtein distance:  3
To change a setting, type "[setting] [value]". e.g. "capslock True"
To go back, type 'return'

```

From this command the user can adjust the dialogue system to their preference. The options we have available are as follows.

Caps-lock: all responses are returned in CAPS.

Typo-checker: an option to ask for confirmation of the correct word when a typo is made.

Debug Mode: if true, the system will return the act for each utterance. And the current and new restaurant preferences. System delay: this will delay the dialogue systems response by a set amount of seconds.

Levenshtein distance: adjusting the Levenshtein distance will increase or reduce the distance from a misspelled word to a correct one. Here a higher number will result in the

system making larger leaps to find the correct word.

Some of these configuration options are more useful than others, but all have their uses. Caps-lock can be switched on in case the dialogue responses have improper capitalisation or no capitalisation at all. Typo-checker will reduce errors made in our spelling correction. This is especially useful when changing the Levenshtein edit distance. Speaking of, changing the Levenshtein edit distance is good for dyslexic people. When making a lot of spelling errors we want a higher edit distance, even at the cost of some accuracy.

The system delay will introduce a "thinking time" in the system. This will make the AI seem more human like and less mechanical. Lastly the debug mode is mostly for our own testing purposes. However, it also makes the AI more transparent and explainable. Therefore enabling it will make the dialogue system more trustworthy.

7 Conclusion

7.1 Discussion of the approach for the dialogue system in general

7.1.1 General overview

The approach lead to using the feed-forward neural network model as a classifier for the dialogue system, as with this model we obtained better scores. With the use of a state transition function, which takes in the current utterance and the current dialogue state, the system is able to take the users utterance as input in order to find a matching response. Based on this, the dialogue transition function classifies the utterance into a dialogue act. The function either proceeds to ask the user for the missing info regarding the preferences, or responds to the user's utterance input.

7.1.2 Limitations

A limitation worth noting in the dialogue system is that the restaurant recommendations may not include information regarding their working hours. This information could be quite valuable when seeking recommendations, as it helps users know when the suggested restaurant is open for business and vice versa it helps the system suggesting a restaurant if the preferred time is gave up by the user.

Another limitation worth noting is that the implementation of the dialogue system is very basic, and might struggle when the user has complex requests. A user might want to add more personal requests than the dialogue system allows, such as specific dietary restrictions or requests. As of now, the dialogue system cannot handle these.

7.1.3 Possible improvements

Having more data is always an improvement. Getting more restaurant data will translate to a more effective way to give the user recommendations.

We often don't handle negation well. We mitigated this by allowing for "deny" and "negate" acts to also change user preferences (so if the user says "no, i want spanish food" it will be corrected), but double negation is a problem. We could add some logic to handle this, and look at levels of negation, but we were content with our system. As long as we often ask for user confirmation, they're aware what the system currently thinks, and they can correct it when wrong, it should be good enough. Additionally, "deny" and "negate" acts didn't always have the best F1 scores, so creating some logic based on a brittle base could cause more hassle than it's worth.

7.1.4 Alternative approaches

The design of the system could've been better — we could've added a proxy for "where" in the transition diagram we are, and explicitly move between states. As it stands, the system just figures it out every time. This isn't too bad, as it's largely a matter of whether enough information was given or not, and if a suggestion was made, but it could get unwieldy if the system was made a lot more complex.

7.2 Discussion of the approach for dialogue act classification

7.2.1 General overview of the experimental methodology

Using the baselines mentioned above, we got a accuracy goal to beat with our machine learning classifiers. The three classifiers chosen for the project were the feed-forward neural network, the logistic regression, and the decision tree. This was mostly done out of general interest in the classifiers and our experience with working with them prior to this assignment. The neural network was created with the help of the Keras package and the logistic regression & decision tree with SkLearn. After making sure they worked we experimented with the hyperparameters of the neural network and logistic regression to get a high accuracy. This was mostly done by trial and error. As soon as our evaluation models for recall, precision and F1 were done this was repeated, but with no significant changes. We added a decision tree out of general interest, it performed adequately compared to the other classifiers as shown above, though not quite as good.

7.2.2 Limitations

Most of the limitations are discussed in section 3.5. Here is a quick overview of our analysis: Underrepresented acts. Some acts are barely present in the data set. This results in our model not classifying them correctly. Complex input. A few utterances are too complicated for the system to classify correctly. Or can be classified as two or more separate acts Ambiguity. Some forms of input are dependant on the model state or previous conversation, which the system does not account for. And lastly negation. As of now our classifier is unable to deal with negative informs such as: "I don't want Indian food", though it can deal with: "No, I want Korean food".

7.2.3 Possible improvements

Logistic regression was chosen as one of the initial classifiers for its simplicity and effectiveness in handling text classification tasks, though we ended up settling on the feed-forward neural network. It provided reasonably good results with an accuracy of 89 percent. However, it was outperformed by other models in terms of F1 scores. The performance was refined through hyperparameter adjustments, but their impact on the final accuracy was rather limited. However, the adoption of a far more substantial model could return significantly improved results. Another improvement for enhancement could be extending the classifiers' scope beyond just the previous utterance, encompassing the entire dialogue history. Moreover, we have previously discussed the potential of incorporating specific keyword rules prior to engaging with the ML models. This proactive approach has the potential to return significant better macro F1 scores, particularly for under-represented classes such as *'reqmore.'*. By introducing rules like *'more = reqmore'* for select instances, we anticipate a boost in performance, especially for acts with limited examples in the dataset. Regarding the limitations of specific dialogue acts, it is probably best to place particular emphasis on the positive values of rule-based approaches with neural networks, particularly in addressing the under-represented acts.

7.2.4 Choice of machine learning algorithm and alternative approaches

While the machine learning techniques performed reasonably well, it is important to acknowledge that dialogues can still present certain data processing challenges. As outlined

in the limitations section, one recurring issue in conversations is the presence of information gaps. To address these gaps, collaborative filtering could be a promising solution. Collaborative filtering can be categorized into two primary approaches: 'Model-based filtering' and 'Memory-based filtering'. 'Memory-based collaborative approaches recommend new items by taking into consideration the preferences of its neighbourhood.' (Roy, D., Dutta, M., 2022). In our recommendation system, the latter approach proves valuable by considering the preferences of neighboring users. Specifically, it employs techniques such as restaurant similarity calculation (item-similarity calculation) in combination with clustering methods. This technique would take in account the choice history of the user and by clustering similar restaurants, with a technique such as K-Means, the system could recommend restaurants that are not only Italian for example, but also share similarities with the user's previously liked Italian restaurants. This personalization ensures that the recommendations are tailored to the user's specific taste and preferences.

8 Overview of contributions of individual group members

	Samir	Giorgio	Tijmen	Yannick	Sarah
Baseline majority	0.5h	0.5h	0.5h	0.5h	0.5h
Baseline rule-based	1h	1.5h	1h	1h	1h
Logistic Regression ML	3h	0.5h	0.5h	2.5h	0.5h
Feedforward NN ML	1h	3h	3h	1h	3h
Evaluation Metrics	0.5h	1.5h	1.5h	0.5h	0.5h
Dialogue Act Diagram	1.5h	1.5h	1.5h	1.5h	1.5h
Dialogue System Interface	2h	5h	2h	3h	4.5h
User preference algorithm	1h	1.5h	2h	2h	1.5h
Adding new properties	1h	0.5h	2h	2.5h	1h
Reasoning implementation	1h	2.5h	1h	1h	2.5h
Report: Data	1h	1h	5h	1.5h	3h
Report: Machine Learning	1.5h	4h	1.5h	1.5h	1.5h
Report: Dialog Manager	1h	1h	1h	3h	1h
Report: Reasoning	1.5h	0.5h	1.5h	1.5h	2h
Report: Conclusion	2.5h	0.5h	1h	0.5h	1h
Report: General	4h	4h	4h	4h	4h
Total	24	29	29	27.5	29

9 Appendix

9.1 Evaluation Tables

9.1.1 Baseline Majority

Baseline Majority	Accuracy	Precision	Recall	F1
inform	0.40	0.40	1.00	0.57
other	0	0	0	0

9.1.2 De-duplicated Baseline Majority

Deduped Baseline Majority	Accuracy	Precision	Recall	F1
inform	0.57	0.57	1.00	0.73
other	0	0	0	0

9.1.3 Baseline Rulebased

Baseline RuleBased	Accuracy	Recall	Precision	F1
inform	0.953	0.908	0.93	0.869
reqalts	0.981	0.869	0.922	0.855
request	0.943	0.956	0.949	0.904
null	0.677	0.912	0.777	0.635
bye	0.8	1.0	0.889	0.8
confirm	0.346	0.6	0.439	0.281
negate	0.945	0.963	0.954	0.912
thankyou	0.993	0.982	0.987	0.975
affirm	0.96	0.993	0.976	0.954
repeat	0.6	1.0	0.75	0.6
ack	0	0	0	0
reqmore	0	0	0	0
deny	0	0	0	0
hello	0.778	1.0	0.875	0.778
restart	0	0	0	0

9.1.4 De-duplicated Rulebased

Deduped Baseline RuleBased	Accuracy	Recall	Precision	F1
inform	0.77	0.88	0.86	0.87
reqalts	0.65	0.91	0.70	0.79
request	0.71	0.89	0.78	0.83
null	0.04	0.05	0.20	0.08
bye	0.00	0.00	0.00	0.00
confirm	0.29	0.33	0.71	0.45
negate	0.77	0.83	0.91	0.87
thankyou	0.67	1.00	0.67	0.80
affirm	0.67	0.86	0.75	0.80
repeat	1.00	1.00	1.00	1.00
ack	0.00	0.00	0.00	0.00
reqmore	0.00	0.00	0.00	0.00
deny	0.00	0.00	0.00	0.00
hello	0.33	0.40	0.67	0.50
restart	0.00	0.00	0.00	0.00

9.1.5 De-duplicated Feedforward Neural Network

Deduped FeedForward NN	Accuracy	Recall	Precision	F1
request	0.978	0.957	0.967	0.936
inform	0.949	0.975	0.962	0.927
affirm	0.905	0.95	0.927	0.864
null	0.879	0.674	0.763	0.617
confirm	0.944	0.944	0.944	0.895
reqalts	0.867	0.867	0.867	0.765
ack	0	0	0	0
bye	0.556	0.833	0.667	0.5
negate	0.875	0.913	0.894	0.808
hello	1.0	0.857	0.923	0.857
deny	0.333	1.0	0.5	0.333
thankyou	1.0	0.647	0.786	0.647
restart	1.0	1.0	1.0	1.0

9.1.6 Logistic Regression

Logistic Regression Model	Accuracy	Recall	Precision	F1
inform	0.987	0.976	0.981	0.964
reqalts	0.963	0.937	0.95	0.904
request	0.993	0.995	0.994	0.988
null	0.889	0.936	0.912	0.838
bye	0.9	0.947	0.923	0.857
confirm	0.808	0.875	0.84	0.724
negate	1.0	0.965	0.982	0.965
thankyou	0.993	0.993	0.993	0.986
affirm	0.987	0.993	0.99	0.98
repeat	0.8	1.0	0.889	0.8
ack	0.25	1.0	0.4	0.25
reqmore	0	0	0	0
deny	0	0	0	0
hello	1.0	1.0	1.0	1.0
restart	1.0	1.0	1.0	1.0

9.1.7 De-duplicated Logistic Regression

Deduped Logistic Regression Model	Accuracy	Recall	Precision	F1
request	0.97	0.936	0.953	0.91
inform	0.971	0.915	0.942	0.89
affirm	0.762	0.941	0.842	0.727
null	0.242	0.571	0.34	0.205
confirm	0.778	0.933	0.848	0.737
reqalts	0.911	0.82	0.863	0.759
ack	0	0	0	0
bye	0.333	0.75	0.462	0.3
negate	0.917	0.957	0.936	0.88
hello	0.833	1.0	0.909	0.833
deny	0	0	0	0
thankyou	1.0	0.647	0.786	0.647
restart	1.0	1.0	1.0	1.0

9.1.8 Decision Tree

Decision Tree	Accuracy	Recall	Precision	F1
inform	0.982	0.978	0.98	0.96
reqalts	0.954	0.972	0.963	0.928
request	0.985	0.989	0.987	0.974
null	0.884	0.893	0.888	0.799
bye	0.975	0.907	0.94	0.886
confirm	0.885	0.821	0.852	0.742
negate	1.0	0.982	0.991	0.982
thankyou	0.993	0.993	0.993	0.986
affirm	0.987	0.993	0.99	0.98
repeat	1.0	1.0	1.0	1.0
ack	0.5	1.0	0.667	0.5
reqmore	1.0	1.0	1.0	1.0
deny	0	0	0	0
hello	1.0	1.0	1.0	1.0
restart	1.0	1.0	1.0	1.0

9.1.9 De-duplicated Decision Tree

Deduped Decision Tree	Accuracy	Recall	Precision	F1
request	0.919	0.925	0.922	0.855
inform	0.952	0.918	0.935	0.877
affirm	0.81	0.81	0.81	0.68
null	0.273	0.529	0.36	0.22
confirm	0.778	0.824	0.8	0.667
reqalts	0.822	0.725	0.771	0.627
ack	0	0	0	0
bye	0.667	0.857	0.75	0.6
negate	0.833	0.952	0.889	0.8
hello	1.0	0.857	0.923	0.857
deny	0.333	0.25	0.286	0.167
thankyou	1.0	0.733	0.846	0.733
restart	1.0	1.0	1.0	1.0

References

1. Roy, D., Dutta, M. *A systematic review and research perspective on recommender systems.* (J Big Data) 9, 59 (2022). <https://doi.org/10.1186/s40537-022-00592-5>
2. https://scikit-learn.org/stable/supervised_learning.html#supervised-learning