

Restaurant recommendation system

Methods in AI Research Report Part 1

Group F3

Majo Bednár (2439905)

FINN JOOSTEN (3874639)

JOP PARO (1410555)

Lonneke Hormann (2475006)

The goal of this project was to design and implement a dialog system for restaurant recommendations capable of interacting with users and understanding their preferences. The system identifies user's preferences through dialog act classification and converses with the user via a state transition diagram and state transition function. To achieve this, several machine learning models were trained and evaluated. The models classify user utterances into predefined dialog acts. Simpler models provided interpretability and efficiency, but they were limited by a data imbalance and not being able to classify utterances not seen before. The system also has four configurability options: controlling Levenshtein distance to handle spelling mistakes, asking for confirmation for the user's preferences, choosing whether the output is in all caps or not, and selecting between the rule-based baseline model or more advanced dialog act classification. Possible improvements are expanding the dataset to make all dialog acts appear a more equal number of times, applying more text preprocessing techniques and incorporating transformer-based architectures for better contextual understanding and handling unseen words. In conclusion, this system provides a foundation for exploring more sophisticated dialog modeling techniques, and could assist in the development of task-orienting systems.

Keywords: AI, recommendation systems, restaurants, machine learning.

1 INTRODUCTION

With the rise of online platforms, consumers rely a lot on digital information. One simple Google search leads to thousands of options to choose from. This is the case in general, but can also be applied to searching for restaurants. To make the consumer's search easier, we designed a restaurant recommendation system. This system asks the user about their restaurant preferences, and based on the input, gives the user a recommendation that matches their preferences.

The models classifying user dialog acts are trained on a dataset [1] of exchanges between a system and a user in the restaurant domain. More about the data can be found in section 2. In section 3 we will discuss the design of the dialog act classification models, evaluate them and discuss some difficult instances. Section 4 will discuss the dialog manager, which uses a state transition function and a state transition diagram. An example dialog using the state transition diagram can be found in appendix 1. The reasoning component used can be found in section 5, including a description and an example. We will also discuss how to handle possible contradictions. A list of configurability options can be found in section 6. Limitations of the model and the conclusion of this paper can be found in section 7.

2 DATA

The dataset used consisted of dialogs from the DSTC2 dataset [1]. In total there are 3235 dialogs in the data, all of which are used in the restaurant domain. Each of these dialogs represent an interaction between a user and an automated system. The user was asked to get a restaurant recommendation based on a certain number of preferences. The dialog contains a sequence of utterances exchanged between the user and the system; the user telling the system their preferences, the system making a recommendation based on said preferences and also general conversational statements. User utterances were recorded using automated speech recognition. This means that both the transcripts are not always perfectly accurate, and the formality of the utterances are affected. This adds difficulty for the system to be trained. The dataset contains 25501 user utterances, many of them duplicated, which is a natural occurrence in human interaction. When leaving duplicates out, there are 5359 utterances left in the dataset. Users often repeat their request to clarify their intent. This could make it easier for the models to train on the data, but could also lead to failing generalization. However, duplicates do reflect realistic conversational patterns and can provide valuable training data for modeling the dialog flow.

To model these interactions, we used a dialog state transition diagram. This diagram provides a structured representation of all the possible conversation states, and the transition between these states based on the user's next dialog act. By mapping the user's input and the system's response to states, the diagram helps in formalizing the dialogue. This makes it easier to visualize and evaluate the behavior of the system. The dialog state transition diagram will be discussed further in section 4.

3 MACHINE LEARNING

The dialog data was organized into two components: the label of the dialog act and the corresponding utterance text. To ensure variety in the data, repeated utterances can be removed. After that, the dataset was randomly shuffled, and the data was split into 85% training data, and 15% test data.

First, we implemented a majority-class baseline. This baseline assigns every utterance to the most frequent dialog act in the dataset. As seen in table x, this gave us a lower bound against which our own models can be compared. Second, we implemented a rule-based system using simple keyword matching. Searches were done without considering upper-or-lowercase letters to find patterns in the utterances. For example, utterances with 'yes' were labeled as *affirm*, and 'thanks' as *thanks*.

as *thankyou*. If none of these conditions were met, the utterance was assigned to the default class *inform*. Table 1 shows that this resulted in a higher lower bound to compare our own models to.

Baseline	Accuracy with duplicates	Accuracy without duplicates	F1 score with duplicates	F1 score without duplicates
Majority label	0.40	0.57	0.04	0.05
Rule-Based	0.81	0.75	0.33	0.28

Table 1. Accuracy and F1 score for majority label baseline and rule-based baseline, with and without duplicate data.

These baselines required minimal parameter tuning. Random shuffling, duplicate removal by keeping the first occurrence of each utterance, and keyword searches were implemented. Together, these baselines were used as reference points for evaluation.

We trained three machine learning models – a decision tree, logistic regression and feedforward neural network (separately before and after removing duplicate utterances). We used bag of words to transform each utterance into a suitable feature vector for the input to the models. That is, these vectors were the size of the number of unique words, and every element in a vector corresponded to the number of tokens in the utterance matching the unique word type.

For the decision tree, we tuned the choice of node impurity measure (gini index, entropy) and maximum depth (5, 10, 20 or any). For logistic regression, we used l2 regularization and we tuned the strength factor of the regularization (0, 0.01, 0.1, 0.5, 1, 5, 10, 20). For the neural network, we used adam optimization with l2 regularization with a 10^{-5} strength factor. We used relu activation functions and trained the network for 200 epochs, unless it converged earlier. We tuned the learning rate (0.001, 0.01), batch size (32, 64) and the number and sizes (number of units) of the hidden layers (64, 128, 128 followed by 64, and 256 followed by 128). For both logistic regression and neural network, we used cross-entropy loss function.

For the hyperparameter tuning, we used grid search containing the mentioned hyperparameters. During this process we performed 5-fold cross-validation for each model, using accuracy as the fitness metric. The model performing best (average of the 5 validation accuracies) was chosen and re-trained on the full training set.

As for the preprocessing of the data, all words in user utterances were converted to lowercase. The user utterances consisted of 1 up to 23 words. As for the labels of the utterances, there was a large imbalance in the training sets. For example, the number of ‘inform’ label was as high as 8621 (2061 when duplicates were removed). On the other end of the spectrum, the ‘reqmore’ label only appeared 4 times (1 when duplicates were removed). Other label counts were somewhere in between. This class imbalance was taken into account when evaluating the models.

For the quantitative evaluation of our various machine learning models, we used accuracy and the F1 measure. Accuracy is used because it gives an overall correctness rate which is simple to interpret and compare between models. We used the F1 measure because it balances precision and recall. This means a high F1 score is only attainable when both precision and recall are high. Accuracy further compliments the F1 score by showing how often the model is right. Tables 2 and 3 show the accuracy and F1 score of the different machine learning models, using training data with and without duplicates.

Classifier	With duplicates	Without duplicates
Decision Trees	0.98	0.89
Logistic Regression	0.98	0.93
MLP	0.99	0.93

Table 2. Accuracy score per classifier, using the dataset with and without duplicates.

Classifier	With duplicates	Without duplicates
Decision Trees	0.92	0.73
Logistic Regression	0.91	0.69
MLP	0.93	0.70

Table 3. F1 score per classifier, using the dataset with and without duplicates.

The tables show a clear difference between model performance depending on whether duplicates were present in the dataset. When duplicates are present, all models achieved very high accuracy (0.98-0.99) and F1 scores (0.91-0.93). However, this does not represent the models’ ability to generalize, as there were (most likely) duplicates present in the test set. After duplicates are removed, accuracy decreases to 0.89-0.93 and the F1 score drops even more substantially to 0.69-0.73. This indicates that, while the models remain correct in many cases, their ability to balance precision and recall across all dialog act classes is more limited. Larger drops in F1 highlight difficulties in correctly identifying less frequent classes. Overall, these results show the importance of evaluating models on datasets without duplicates to get a realistic measure of their performance.

There also were some dialog acts which were more difficult to classify. In table 4 below are the three most difficult to classify dialog acts per classifier.

Classifier	With duplicates	Without duplicates
Decision Trees	Ack, deny, confirm	Deny, ack, bye
Logistic Regression	Ack, deny, repeat	Deny, ack, bye
MLP	Ack, deny, repeat	Deny, ack, bye

Table 4. The most difficult to classify dialog acts per classifier.

As seen in table 4, acknowledgements, deny, confirmation and dialog ending seem to be the hardest to classify for all systems. Acknowledgements can be difficult because they often are short, and ambiguous. They can also overlap with confirmations, like “yes, that is right”. Deny, just like acknowledgement, is also difficult because they often are short. They can also be misclassified because of negated confirmation. Confirmation may be difficult for the same reason as acknowledgement, because they can overlap. Dialog ends can be difficult because they are short, and because it could be hard to recognize more subtle ways of ending the dialog. Improvements in preprocessing can be made by stemming words and using tokenization.

Based on the most common misclassifications, we came up with two types of difficult instances. For each of these, we had two test instances to evaluate how the classifiers would classify these. Seeing as the classifiers struggle with negation, we tried to test on double negations. We also tested how implicit conversation enders would be classified, because the bye dialog act is also difficult for all classifiers. We used the following test instances, with the intended label:

1. I don't want to eat anything that is not Italian. (double negation, inform)
2. I don't disagree with your choice of restaurant. (double negation, confirm)
3. Great, that is all I needed. (implicit conversation ender, bye)
4. Okay, I think I'm good now. (implicit conversation ender, bye)

We have tested these utterances on models trained on both datasets with duplicate data and without duplicate data. Table 5 shows the results from the dataset with duplicate data, and table 6 shows the results from the dataset without duplicate data.

Classifier	1	2	3	4
Decision Trees	inform	inform	nan	nan
Logistic Regression	inform	request	nan	ack
MLP	inform	inform	nan	ack

Table 5. Classifications made by each classifier for each test instance. The dataset with duplicates was used.

Classifier	1	2	3	4
Decision Trees	inform	inform	deny	ack
Logistic Regression	deny	request	nan	ack
MLP	deny	inform	nan	ack

Table 6. Classifications made by each classifier for each test instance. The dataset without duplicates was used.

As seen in tables 5 and 6 above, if the label of a double negation is inform, this is easy to predict for all models, especially with duplicate data. However, it is harder when the negation is based on rarer acts like confirm, both with and without duplicates. The models misinterpret the negation and this results in the wrong class.

Implicit final utterances are always misclassified. The models fail to generalize on polite, implicit closings. The effect of duplicate data once again is present; without duplicate data, errors increase. It appears that instance 3 is hard to even classify, and in instance 4, the word okay causes it to be misclassified as acknowledgement.

4 DIALOG MANAGER

For each state that the dialog manager occupies, the next state is inferred. It does this based on the current state it is in, the context thus far and the current user utterance. The current dialog state is specified after the diagram states. The context contains information about the preferences and suggested restaurants and alternatives. The current user utterance is given and recorded while in the current state. Additionally, the dialog manager has access to data of all the possible restaurant

suggestions, which contains the area, food type, price range and other information like the address and phone number. The dialog manager starts in the welcome state (1) for the current state and no context.

The functional flow of the dialog manager can be seen in the State Transition Diagram (STD) as shown in Figure 1. After starting up, the user is welcomed by the system (1), the system will obtain the three user preferences (area, food type, price range) (2-4). These are extracted from the user input through matching tokens with regular expressions. Once these have been assigned to the context, a match is found through the restaurant lookup function. Depending on the amount of suitable restaurant matches, one of three states is chosen. If there are none the preferences are again asked (5). If there is just one restaurant match, this is suggested (6). If there are multiple suitable options, additional preferences are asked for: touristic/assigned seats/children/romantic. (8)

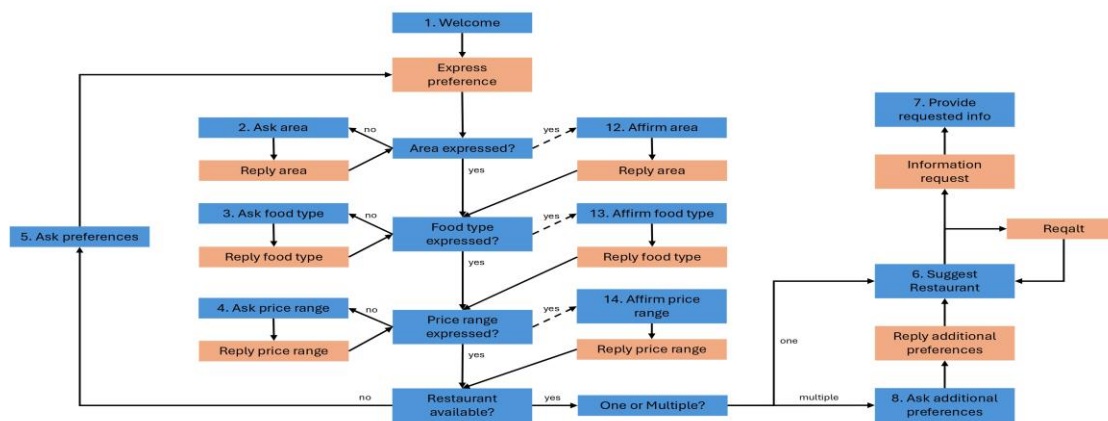


Figure 1. State Transition Diagram (STD) for dialog manager of restaurant recommendation system. Dashed arrows show optional paths.

Each state has a system utterance template. Most of the utterances are fixed, like that of the welcome state (1) and the ask preference states (2-5). Some, however, have a formatted template. The first states from the state diagram that have a formatted system utterance template are the confirmation states (12-14), as these have a variable for the area, food type and price range. The state where a restaurant suggestion is provided (6) has a variable for the restaurant name. The system utterance templates that provide information (7) have a variable for the phone number, address and postcode.

This is the complete list of system utterance templates:

1. "welcome": "Welcome to the restaurant recommendation system! \n"
"I can help you find a restaurant based on your preferences for area, food type, and price range.\n"
"If you don't care about a certain preference, just say 'any'.\n"

- "Lets get started! Please tell me your preferences."
2. "ask_area": "Which area are you interested in?"
 3. "ask_food": "What type of food would you like?"
 4. "ask_price": "What price range are you looking for?"
 5. "ask_preferences": "Please tell me your preferences (area, food type, price range)."
 6. "suggest_restaurant":
 - a. "I suggest: {restaurant}. Would you like more information or another suggestion?"
(Example: *I suggest: fitzbillies restaurant. Would you like more information or another suggestion?*)
 - b. "I suggest: {restaurant}. The restaurant is {additional preference} because {preference reasoning}.\nWould you like more information or another suggestion?"
(Example: *I suggest: fitzbillies restaurant. The restaurant is romantic because it allows you to stay for a long time. Would you like more information or another suggestion?*)
 7. "provide_postcode": "The postcode for {restaurant} is {postcode}."
(Example: *The postcode for galleria is c.b 2.*)
 "provide_phone": "The phone number for {restaurant} is {phone}."
(Example: *The phone number for galleria is 01223 362054.*)
 "provide_address": "The address for {restaurant} is {addr}."
(Example: *The address for galleria is unknown.*)
 8. "ask_additional_preferences": "There are multiple restaurants to choose from.\n"
 "What is your additional preference (touristic/assigned seats/children/romantic)?"
 12. "confirm_area": "You want a restaurant in the {area} area, correct?"
(Example: *You want a restaurant in the midtown area, correct?*)
 13. "confirm_food": "You want {food} food, correct?"
(Example: *You want Indian food, correct?*)
 14. "confirm_price": "You want a {price} restaurant, correct?"
(Example: *You want a cheap restaurant, correct?*)
- Miscellaneous system utterances:
- "goodbye": "Goodbye!",
- "clarify": "Sorry, I didn't understand. Could you please rephrase?"

In appendix 1 is a full example dialog that traverses the diagram from the start state to the end state.

5 REASONING

To further extend the dialog system, a reasoning component is added that enhances the configurability of restaurant choice. If multiple restaurants fit the user's preferences, the dialog system asks: "What is your additional preference (touristic/ assigned seats/ children/ romantic)?" Then the user input is scanned for the following words: {no, don't, touristic, assigned seats, children, romantic}.

If the user types "no" or "don't", the dialog system knows that there is a negation in the sentence. Then it searches for other keywords. If they are found, the dialog system starts reasoning so that it can suggest a restaurant that matches the desires of the user.

For the reasoning, the dialog system uses the following information available about the restaurants: {price range, food quality, food type, length of stay, crowdedness}. It uses these properties to infer whether the restaurant is touristic, has assigned seats, is romantic or is suitable for children.

The reasoner follows 6 implication rules:

1. Cheap and good food \rightarrow touristic
2. Romanian food \rightarrow not touristic
3. Busy \rightarrow assigned seats
4. Longstay \rightarrow no children
5. Busy \rightarrow not romantic
6. Long stay \rightarrow romantic

By examining this logic, one can see that possible contradictions can arise; for example, a restaurant could be busy and yet have a long wait. Following the logic rules stated above, this restaurant should be romantic and not romantic at the same time, which is impossible. When encountering such contradictions, false overrules true in our reasoner. Hence, the dialog system would mark this restaurant as not romantic.

After the reasoning part is complete, the dialog system outputs to the user which restaurants satisfy the extra preferences and explains its reasoning. An example of how a dialog with our dialog system looks can be found below.

Dialog system:

Welcome to the restaurant recommendation system!

I can help you find a restaurant based on your preferences for area, food type, and price range.

If you don't care about a certain preference, just say 'any'.

Let's get started! Please tell me your preferences.

User:

I would like to eat expensive Indian food in the east.

Dialog system:

There are multiple restaurants to choose from.

What is your additional preference (touristic/assigned seats/children/romantic)?

User:

I would like to eat in no romantic restaurant.

Dialog system:

I suggest curry prince. The restaurant is not romantic because it is busy. Would you like more information or another suggestion?

6 CONFIGURABILITY

Our recommendation system has the following four configurability options for the user:

- Maximum correcting distance for Levenshtein edit distance
- Always confirm for each preference or not
- Output in all caps or not
- Use baseline dialog act recognition

While using the restaurant recommendation system, the user can use any of these options if they want to. They are incorporated as follows:

By editing the maximum correcting distance for the Levenshtein edit distance, the system becomes stricter or more forgiving. This is used to recognize the user's words even if they are misspelled. If the maximum Levenshtein edit distance is small, there are only a small amount edits that can be made to compare two strings. This means the system only accepts inputs that are very close to known words. If the distance is set to be larger, the system will accept inputs that are further away from known words. The user's desired Levenshtein distance is extracted and used in the autocorrect function to determine how tolerant the system should be to the user's spelling errors.

The always confirm for each preference option is used to validate each user preference explicitly or to accept them automatically during dialog. Whether the user wants confirmation is extracted by the system. If the user wants confirmation for each preference, the system outputs a question asking for confirmation about the preference it extracted.

The option to output in all caps or not controls how the system formats its text responses. This is useful to maintain consistent formatting or to make outputs more standardized. In the code this is implemented by extracting the user's choice to determine whether the output should be in all caps or not, and if true the output will be in caps.

When the baseline dialog act recognition is enabled, the system uses the simpler, rule-based baseline model to recognize the user's input and recommend restaurants instead of the more complex models, giving a different output.

7 CONCLUSION

In summary, we have designed a dialog system for restaurant recommendations. We realized this by training different machine learning models for dialog act classification and by using a state transition diagram and state transition function for the flow of the dialog. The system also uses a reasoning component to interpret user preferences and has four configurability options for the user to choose from.

However, there are some limitations for the dialog system in general. Because of the large amount of duplicate data, there is a decrease in available training data without duplicates. This makes it harder for the system to learn. This could also explain why some dialog acts, like implicit final statements, are harder to classify accurately.

This could be improved upon by using a larger and more diverse dataset with less duplicates.

For the dialog act classification, we implemented two baseline systems, and trained three machine learning models: decision trees, logistic regression and MLPs. The goal was to automatically classify the user's input into one of several dialog act classes. The experimental methodology involved preprocessing the dataset, vectorizing text inputs and training

of the model. The data was converted to lowercase for processing and Levenshtein distance was used for keyword matching. The system used the machine learning models' predictions to incorporate the dialog acts into the state transitions.

Nevertheless, there are some limitations to the current approach. First of all, the model disregards words that were not seen before during training completely. So, some of the user's input will be ignored if the system has not seen it before. This can lead to misclassification of dialog acts that correspond with the unseen utterances. Secondly, the data distribution across dialog acts is not balanced. Some dialog acts appear far more frequently than others. This could bias the system, and limit its accuracy for less frequent dialog acts.

To improve upon this, more preprocessing of the data could be implemented. For example, stemming or removing stop words can be used for better generalization and noise reduction. Instead of disregarding unseen words, a potential approach could be to use a synonym dictionary to map unseen words to semantically similar words. This way, the model could handle the unseen words correctly.

The current machine learning models are simple and interpretable, but they cannot recognize the semantic relationships between words. An alternative to this would be to use transformer-based architectures. These provide contextualized word embeddings, meaning the model could generalize to unseen words better.

For future research, transformer-based architectures can be used to capture context and meaning, enabling more nuanced dialog act classification.

REFERENCES

- [1] M. Henderson, B. Thomson, and J. D. Williams, "Second Dialog State Tracking Challenge (DSTC2/3): human-computer dialogs in the restaurant information domain," 2014. [Online]. Available: <https://www.microsoft.com/en-us/research/event/dialog-state-tracking-challenge/>. [Accessed: 30-09-2025].

APPENDIX 1: EXAMPLE DIALOG

The user utterances are indicated with the ">" symbol:

Welcome to the restaurant recommendation system!

I can help you find a restaurant based on your preferences for area, food type, and price range.

If you don't care about a certain preference, just say 'any'.

Lets get started! Please tell me your preferences.

> I would like to eat in the south area

You want a restaurant in the south area, correct?

> yes

What type of food would you like?

> I would like Vietnamese food

You want vietnamese food, correct?

> yes

What price range are you looking for?

> I am looking for expensive food

You want a expensive restaurant, correct?

> yes

There are multiple restaurants to choose from.

What is your additional preference (touristic/assigned seats/children/romantic)?

> touristic

Sorry, no restaurant matches your preferences. Would you like to try different preferences?

Please tell me your preferences (area, food type, price range).

> I would like to eat in any area

There are multiple restaurants to choose from.

What is your additional preference (touristic/assigned seats/children/romantic)?

> I am looking for a romantic restaurant

I suggest: fitzbillies restaurant. The restaurant is romantic because it allows you to stay for a long time.

Would you like more information or another suggestion?

> another suggestion

I suggest: galleria. Would you like more information or another suggestion?

> What is the address?

The address for galleria is unknown.

> What is the phone number?

The phone number for galleria is 01223 362054.

> what is the post code?

Sorry, I couldn't understand what information you requested. Please ask for postcode, phone, or address.

> what is the postcode?

The postcode for galleria is c.b 2.

> thanks bye!

Goodbye

APPENDIX 2: OVERVIEW OF CONTRIBUTIONS OF INDIVIDUAL GROUP MEMBERS AND REFLECTION

Task	Majo	Finn	Jop	Lonneke
Reading dialog acts		0.5 hr		0.5 hr
Getting to work with the data (extracting/ exploring)			1 hr	
Implementing baselines			5 hr	
Implementing ML models	5 hr			
Evaluation of the different classifiers	1 hr			6 hr
Constructing STD		3 hr		
Implementing STD in code (including debugging)	1 hr	7 hr		
Implementing preference extraction	5 hr		1 hr	
Implementing lookup function			2 hr	
Implementing reasoning	1.5 hr			
Implementing configurability options			3 hr	
Writing and updating README file		2hr		
Formatting report, title page, abstract and references				1.5 hr
Report: introduction				1 hr
Report: data				1 hr
Report: machine learning	1 hr			3 hr
Report: dialog manager		4 hr		
Report: reasoning component			2 hr	
Report: configurability				2 hr
Report: conclusion				1.5 hr
Report: reviewing and editing written parts				0.5 hr
Total	14.5	16.5 hr	14 hr	17 hr

We had a little bit of a slow start due to members being sick at the start of the course, and a member not being able to come to the seminars in general. Nonetheless the cooperation was good and our productivity got better each week.