Team 13

DAVID PANTOPHLET, studentennummer
NICK BEUKENKAMP, 2604930
GOYA VAN BOVEN, 6981844
SIMON VAN DE FLIERT, 5708656

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

1 INTRODUCTION

In this study we create a robust dialogue system for restaurant recommendation using automated text classification, dialogue management and reasoning. We use dialogs from the second Dialog State Tracking Challenge [2] as a basis for our system. The aim of the system we develop and evaluate is to advise users about restaurants that meet their wishes.

This report is structured as follows. We give a description of the data in Section 2 and describe our text classification models in Section 3.1. After evaluating our models (Section 3.2) we discuss the dialogue manager (Section 4) and the reasoning supporting (Section 5). Finally we evaluate our system and find that **... to be analysed**

2 DATA

We use the data from the second Dialog State Tracking Challenge [2] as data to train our text classification models on. This dataset, which consists of 3235 dialogs, includes sentences user-system interactions in the restaurant domain. In these interactions, the user aims to get a restaurant recommendation from the system that corresponds to their preferences. The dialogs were collected through automatic speech recognition. We must thus keep in mind that this influences the quality of the descriptions, as some words might be recognized incorrectly. A sample interaction from the dataset looks as follows:

```
system: Hello
user: I am looking for a cheap restaurant
system: what kind of food would you like?
user: indian
```

The sentences in the dataset are annotated for their *dialog act*: actions that are being performed by the speaker 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.

Authors' addresses: David Pantophlet, studentennummer, 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.

Manuscript submitted to ACM

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

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

Additionally, we have another dataset containing information on 109 restaurants, which will be used for the recommendations made by the system. Specifically we have the following information for each restaurant: the name, address, postal code, telephone number, food type, area and pricerange.

Our first task is to automatically classify the dialog statement for a given sentence. In order to get a greater insight into the distribution of the dialogue statements in our dataset, we plot the number of occurrences per statement in Figure 1. We find the distribution is heavily skewed: `inform` is with 10.160 occurrences by far the most common statement in our dataset, followed by `request` and `thankyou` (6.494 and 3.259 occurences respectively), while none of the 9 least common labels occur more than 450 times.

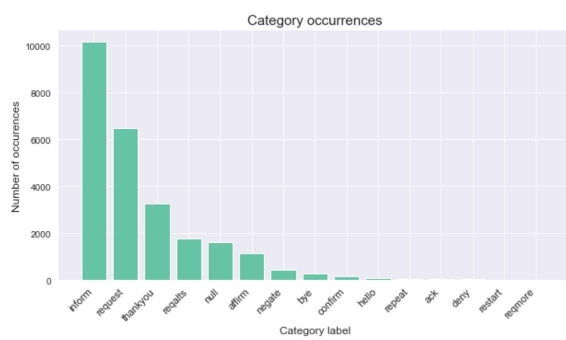


Fig. 1. Occurrences per category label in our dataset

3 MACHINE LEARNING

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

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

For this, 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 marjority class label (Section 3.1.1) and one based on keyword matching (Section 3.1.2). Next, we transform our 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 [3]: a logistic model (Section 3.1.3) and a random forest model (Section 3.1.4). We evaluate the performance of all models in Section 3.2.

3.1 Model descriptions

3.1.1 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 is therefore a suitable baseline to compare our models against.

3.1.2 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 for each dialog act. We identified keywords in two ways:

- through a brainstorm about the words that are likely to be used frequently for each dialog act;
- by investigating the words that co-occured most often with each dialog act in the dataset.

The full list of keywords can be found in the Appendix (Table 3).

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 2. 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 a certain dialog act than for any other, the model predicts that act;
(2) But if there are multiple acts which have an equal number of keyword matches, the system selects the act with the highest number of occurrences in the dataset (see Figure 1).

.

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

3.1.4 Random forest model. The second machine learning model that we build is a random forest model. This model creates multiple decision trees. When predicting the dialog act for a given sentence, each tree makes a prediction, and the act that is predicted most often is the one that is assigned. [add benefits random forest over DT]

We experiment with various maximum tree depths and find that using a depth of 20 gives a good performance in terms of accuracy. [add other hyperparameters].

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 dialog 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,

Algorithm 2 Pseudocode for the Keyword matching algorithm

```
labels ← [labels]
keywords ← [label: [keywords]]
train_pred ← []
for every sentence do
    labels_counter ← [[label, counter]]
    for every word in sentence do
        for every keyword, label in keywords do
            if word in keyword then
                labels_counter[label]+ = 1
            end if
        end for
    end for
    max ← one label with highest counter
    maxes ← all labels with equal highest counter
    if len(maxes) == 1 then
        train_pred.append(maxes[0])
    else if len(maxes) == 0 then
        train_pred.append("inform")
    else
        train_pred.append(highest ranked label)
    end if
end for
```

	Accuracy	Macro F1	Precision	Recall
Majority	0.3761	0.042	0.0769	0.0289
Keyword	0.8204	0.5640	0.5552	0.6352
Logistic	**0.9739**	**0.8424**	**0.9028**	**0.8115**
Random Forstest	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

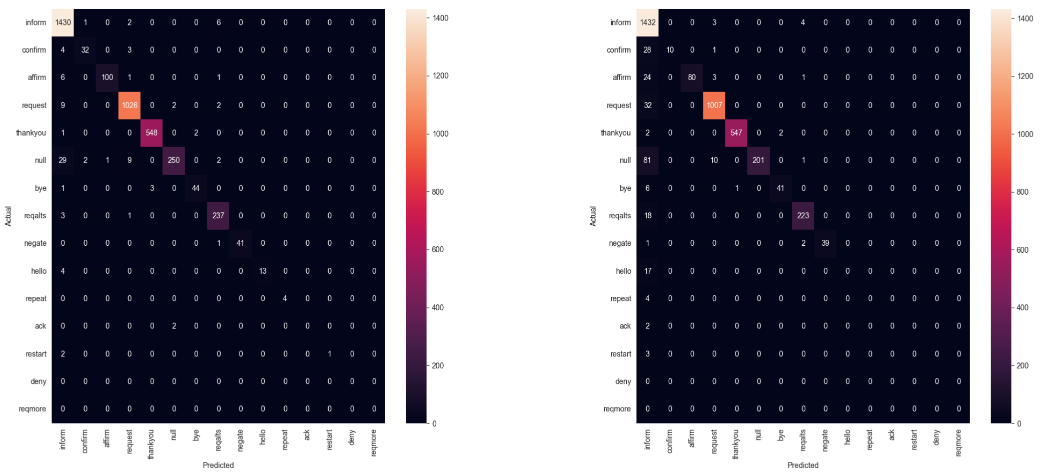
precision and recall return the unweighted average of the performance on each dialog act, providing better insight into the performance per act. Although this in turn overestimates the importance of some low frequency acts which are less important for our task, the combination of accuracy and marco F1 gives a good overview of the performance for both highly 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=8424). Although the random forest model achieves a high accuracy score (0.9357), its F1 score is much lower (0.5985), indicating that model likely performance 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, so that we get more insight into how well the model predicts each dialog 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 were for act `null` (43 incorrect predictions). This problem was exasperated in the random forest model (102 incorrect predictions). Sentences with this label are particularly hard to classify for three reasons:

(1) These utterances often consist of unfinished words or sentences;
(2) The variety of words was very large in this dialog act category;
(3) Many words in the sentences belonging to this act are also frequent in other dialog act categories.



(a) Linear regession model

(b) Random forest model

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

When we compare the performance on utterances in our testset we find that a total of 71 utterances misclassified by all models. Two sentence structures that proved 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.

Difficult cases:

Incomplete words or sentences: *are there* || *the addre* || *arts town*

All the classifiers predicted the wrong dialog act for these sentences.

Instances that combine different dialog acts: *hi im looking for an expensive restaurant* || *uh yeah can i find a restaurant in the south part of town*

The classifiers also predicted the wrong label for both of these instances.

4 DIALOG MANAGER

4.1 State transition diagram

We analyse the dialogs in our dataset (Section 2) and based on these we create a state transition diagram (Figure 3), which contains the structure our dialog system will follow during conversations. We implement this structure directly. We recognize the dialog act of each user utterance using the logistic classification model, as this model gives the best performance. Using this dialog act and the current state of the system, the system recognizes how it should update its state, according to our dialog structure. We will now elaborate more on this structure, and the utterance that are expressed in each state.

The system opens the conversation with a welcome welcome message to the user (State 1), stating the purpose of the chatbot and the choices the user can make. This message look 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 restaurants:`

The system thus asks for the (i) area, (ii) food type and (iii) price range preferences of the user (State 2). When the user expresses any or multiple of these preferences (utterances that are of the `inform` dialog act), they are recognized as follows:

- We match if any keywords that are often used for expressing preferences. For instance from the user input *I would like cheap Italian food tonight* the preferences `(pricerange=cheap), (food=Italian)` are extracted, based on the keywords *cheap* and *Italian*. For some of the keywords we include multiple spelling variations.
- If this does not yield any matches, we try to recognize common patterns in the text, such as for example `... part of town`, where the word on the left of this pattern usually indicates the area preference of the user. A sentence using this pattern looks as follows *I'd like to eat in the northern part of town*, where we extract `(area=northern)`. Using Levnsthein distance we then map the extracted preference to its closest keyword, *north* in this case, where we take a maximum distance of 3.

After receiving user input, the system checks whether information is provided for all categories, and if so the system can make a recommendation. The system checks whether there are any restaurants that match the users preferences. If there are none, the system returns a negative response and asks the user to change their preferences (State 3). If a restaurant is found, it is suggested to the user, who can either accept or deny the suggestion (State 4). An example dialog of the system so far (after the welcome message) looks as follows, where `system internal` shows the classifications the system makes internally:

```
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)
System: I recommend you to go to wagamama. Would you like to go there?
User : yes please
System: Great! The address is 36 saint andrews street. Would you like to know the phone number or the postcode? Or maybe both?
```

If the user accepts the suggestion, they can ask for the following information about the restaurant: the postal code, address, phone number or any combination of the three. The system recognizes these request through recognizing keywords (such as *phone number, zip code* and *all*) in the user input. The responses given to the user are formed from a template, where negative responses have the template "Sorry, we do not have *requested information* for *restaurant*", whilst a positive response can be seen in the example above, where the system responds with a recommendation. As this can be done in any order, we again combine these questions into one state. Once the user has received all requested information they can end the dialogue through saying goodbye (State 7) or thank you (State 6). An example of this final part of the converation looks as follows:

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

The user can also exit the conversation at any time by saying goodbye, which is denoted by state 7 on the left hand side of the diagram.

5 REASONING

6 DISCUSSION

7 OVERVIEW OF CONTRIBUTIONS OF INDIVIDUAL GROUP MEMBERS

REFERENCES

[1] John Langshaw Austin. 1975. *How to do things with words.* Oxford university press.
[2] 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.
[3] 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.

.1 Keywords

Dialog 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, laters
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 3

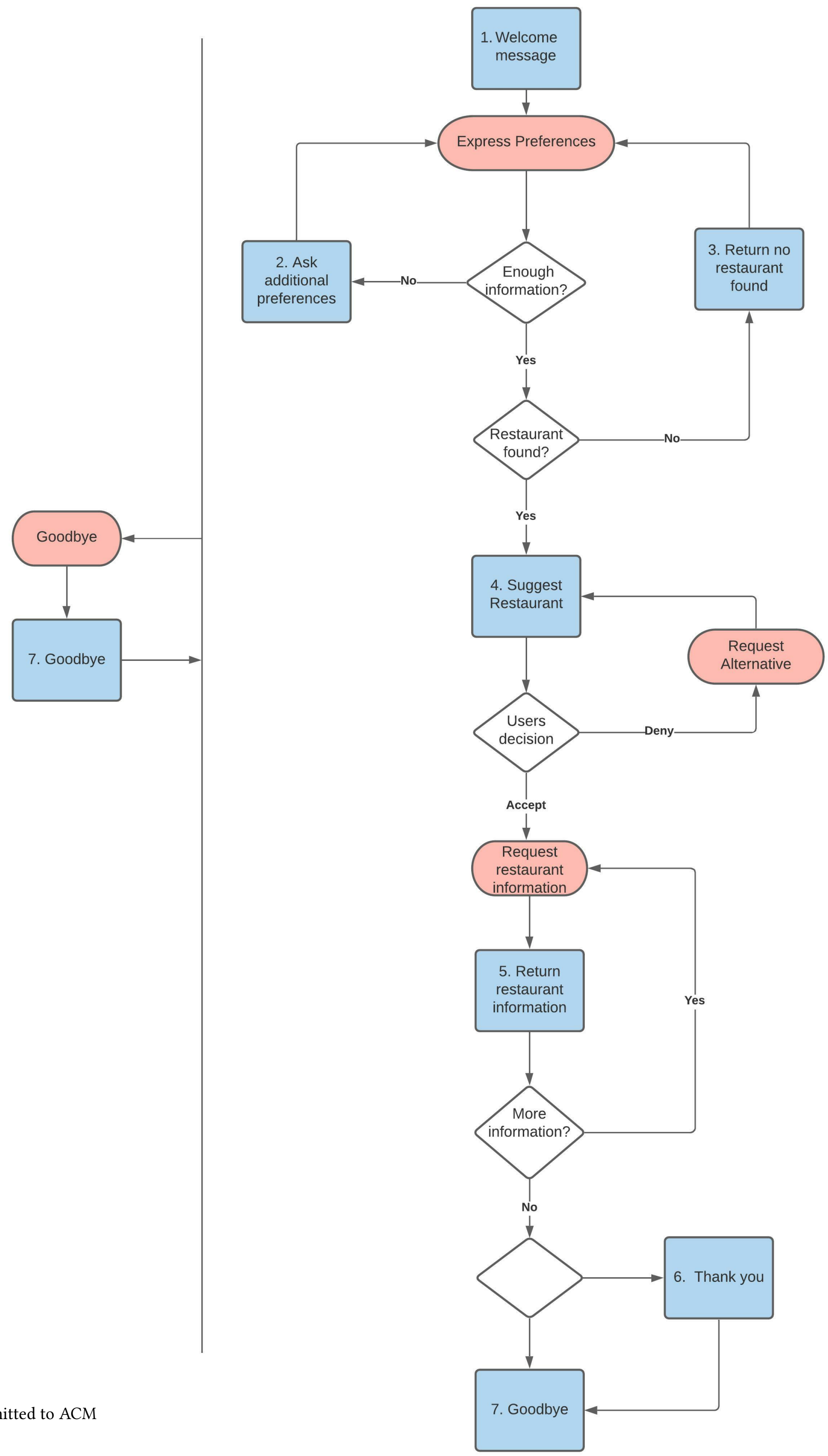


Fig. 3. State transition diagram created after analysing the dataset