Report: Part 1

Maarten San Giorgi
m.e.sangiorgi@students.uu.nl
5723493

Otto Mättas
o.mattas@students.uu.nl
6324363

David-Paul Niland
d.p.niland@students.uu.nl
6688721

Lonnie Bregman
l.bregman@students.uu.nl
6980562

ABSTRACT

Dialog systems are a fantastic way to use AI to complete tasks that if performed by a human, would require intelligence. Dialog systems have seen considerable progress in recent years with examples of dialog systems passing the Alan Turing test. They have also made considerable progress with the advancement of processing power and machine learning techniques.

The goal of this project was to design, build and implement a dialog system that would be used to help users find restaurants. The team made use of a text-based dialog system that utilizes machine learning and text classification in order to select which restaurant would suit the user the best. It achieves this goal through utilising previous user's conversations. In order to do this, the system prompts the user with questions in order to narrow down options and to select a restaurant from the database. The project team made use of data and domain modelling, machine learning, pattern recognition and text classification to achieve its goal.

Author Keywords

Abstract; dialog; AI; dialog systems; text-based dialog system; classification; order system; prompts questions; data pattern recognition.

INTRODUCTION

The main aim of the project was to provide the user with a clear, working system that gives restaurant recommendations. Team 18 used a large pre-made corpus of data, which consisted of previously recorded text conversations between users and a dialog system. This system was from previous third-party research called the Dialog State Tracking Challenge.

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

DATA

In order to navigate through the large corpus of data, the user is asked questions about the restaurant in order to narrow down the selection. The group noted three main distinctions when manually going through the data which were potential ways of narrowing down the selection process, which were; price, area and restaurant type. An implicit diagram was designed as a blueprint of how the system would be implemented.

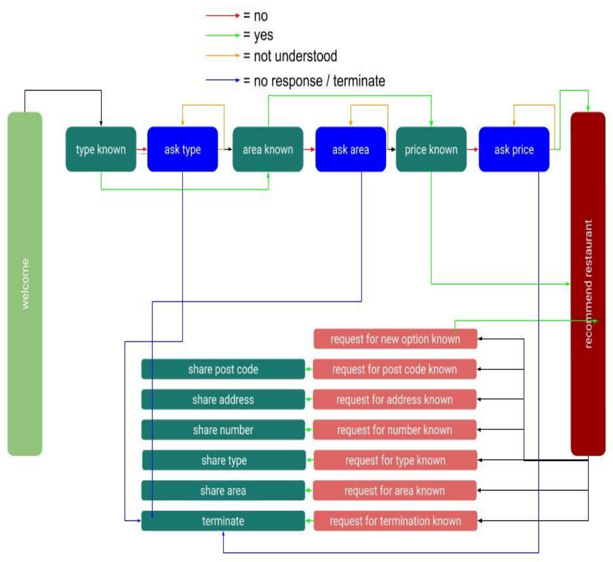

Diagram 1. State Transition Diagram.

The diagram above represents how the system will respond to user utterances. Once price, area and type are known, the system can then make a restaurant recommendation to the user, based from what was said about the restaurants in the corpus of data.

Once the diagram was made, the team focused on building the first part of the system using the programming language Python. Because the corpus of data was presented in JSON format, the team had to write code to read through all of the files in the directory and display them line by line. Users of the system may use the Enter key to scan through the dialogs.

Another step was carried out by the team to make the dialogs easier to use for later stages. This involved writing all of the dialogs to a single text file which was human-readable.

MACHINE LEARNING

Following this, the team implemented two baseline systems. The first was a rule-based system that focused on keyword matching to associate the correct dialog act with whatever the user utterance had been. With a file per dialog act, including a few keywords to match, the rule-based system looks for a possible match between the keywords in the files and the words in the input sentence.

The second was a baseline system which randomly picked a dialog act based on the distribution in the dataset. First, the system calculated the distribution and created a normalized cumulative distribution from 0 to 1. Then, the system picked a random number between 0 and 1, and checks for each dialog act if it is a lower number. When this is the case, the system returns this dialog act. This is completely independent from the test data.

The machine learning part is implemented with a logistic regression. First, the data is split into training data and testing data (85% and 15%). After this, the data is transformed into vectors, so that the input is in the correct format for the logistic regression. Then the data is trained, using the sklearn library. Finally, the logistic regression can be tested by inputting sentences.

In the source code a function for a neural network approach can be found as well. This approach uses the keras library, but because of an error, this does not work, so the results can't be seen.

One of the things that was hard to get right was the Levenshtein distance. This tool helps the system when user makes a grammar mistake, but it can give false positives as well. For example, the word "eat" can be corrected to "east", while the user clearly meant "eat". This is solved by only using the Levenshtein in pattern finding, so that it will only correct expected words.

The system finds it quite hard to recognize requests for certain properties of a restaurant. Sometimes the user has to type the request in multiple ways before the system understands what the user wants. Furthermore, the system does not always wait for a right answer, but assumes that the given answer is what it needs. For example, when the system asks the preferences of the user and the user answers with "no", the system continues with the dialog, instead of offering other alternatives.

Another thing which might be frustrating for the user is when there is no existing restaurant with the user preferences. This could be solved by finding a restaurant that is close to those preferences, but with the current implementation the user has to keep adjusting its preferences until a restaurant is given. The user is not able to fill in a wildcard as well.

The styles contained in this document have been modified from the default styles to reflect ACM formatting conventions. For example, content paragraphs like this one are formatted using the `Normal` style.

DIALOG MANAGER

The state transition function starts in the "get info" state. In this state, the function keeps track of what restaurant choice information is still missing (type of food, area and price class) and asks for that information until everything is known.

After all the information is retrieved and confirmed by the user, the system transits to the "first proposal" state. If a restaurant can be found that meets the requirements, it will be proposed to the user. Otherwise, the system returns to the "get info" state. After a restaurant is suggested, the system transits to the "waiting for approval" state. Upon the user's approval of the restaurant suggestion, the system prompts the user if they have any further requests and goes to the "waiting for request" state. If the user does not approve of the suggestion, the system changes to the "next proposal" state.

While in the "waiting for request" state, the user can ask for information about the restaurant, like telephone number or address. If they deny this, the system is finished and the function returns. While in the "next proposal" state, the system checks if there are any further options for restaurant that meet the given requirements. If so, it proposes one of these and goes to the "waiting for approval" state. If not, it asks the user to change the requirements and it goes back to the "get info" state.

Our state transition function does not reflect our state diagram, in the sense that the welcoming state is skipped and the function immediately starts in an information collecting state. Moreover, the states "ask type", "ask area" and "ask price" are put together in a single state "get info". The "recommend restaurant" state in the diagram is split into two parts: "first proposal" and "next proposal". This was done for programming reasons.

Finally, the user requests for information about the restaurant are put together in a single state "waiting for request".

Example of a dialog:
S: Welcome to the A-team Restaurant Finder!
U: Hello!
S: What type of food would you like?
U:i am in the mood for chinese food
S: In what area would you like to eat?
U: new york
S: In what area would you like to eat?
U: in the east
S: What price range would you like?
U: i am a broke student
S: What price range would you like?
U: cheap please!
S: Is a cheap chinese restaurant in the east part of town what you are looking for?
U: yes
S: I can't find a cheap chinese restaurant in the east part of town.
Try changing your preferences.
U: ok a little more expansive is ok, i will loan money
S: Is a cheap chinese restaurant in the east part of town what you are looking for?
U: no
S: Please change your preferences.
U: expensive
S: Is a expensive chinese restaurant in the east part of town what you are looking for?
U: yes
S: How about yu garden ?
U: sounds great
S: I can't tell if you are confirming the offered restaurant. Please, try again.
U: yes
S: Okay. Do you have any requests?
U: phone number
S: phone: 01223 248882
Do you have any other requests?
U: no thank you
S: Okay, goodbye!

DISCUSSION

Overall, the system meets the main aims of the research topic. In some areas the system works quite well, however, there are some areas that may be attributed to further research. Some time was spent trying to get a neural network classifier to work running alongside the system to test how well it might work against the original logistic regression algorithm. The system proved to have some bugs and the team had to keep moving forward with other parts of the project so this part was left incomplete. The team also discussed the alternative approach of using a decision tree as a machine learning classifier, but decided that this approach could be taxing on processing power, given the size of the dataset.
Another option for improvement might be to begin contextualising how the user inputs sentences into the system. If a user mentions that they wanted to eat somewhere "East", the user would read that in the same way as "I do not want to eat somewhere East".

Future work might also include an amended dialog flow chart that closer matches what was implemented in our system.

TEAM MEMBER CONTRIBUTIONS

Overall, all team members were active. There was unified aim towards deadlines as well as quality output. While Maarten and Lonnie were focusing more on coding and implementation, ideation with related tasks were handled by David-Paul and Otto. During labs, everyone was participating in the collective effort. Team members were supportive and attentive towards each others work, seeing ways to improve as a group.

Task	Team Member	Time Spent
Setting up the repository	all	1h
Generating State Transition Diagram	Otto	4h
Improving State Transition Diagram	Maarten, David-Paul, Lonnie	1h
Generating code for Part 1a	Maarten, David-Paul, Lonnie	1h
Improving code for Part 1a	Lonnie, Maarten	1h
Submitting Part 1a	Otto, David-Paul, Lonnie	1h
Generating code for Part 1b	all	2h
Improving State Transition Diagram	all	1h
Managing the repository	Otto	1h
Creating baseline rules template	Otto	1h

Generating baseline rules	Otto, David-Paul	1h
Generating code for Part 1b	Lonnie	3h
Generating code for Part 1b	David-Paul	3h
Generating code for Part 1b	Maarten	5h
Submitting Part 1b	Otto, Maarten	1h
Setting up a session	Maarten, David-Paul, Lonnie	1h
Generating code for Part 1c	Otto	2h
Generating code for Part 1c	David-Paul	4h
Generating code for Part 1c	Lonnie	2h
Generating code for Part 1c	Maarten	4h
Testing code for Part 1c	Maarten, Otto, David-Paul	1h
Submitting Part 1c	Otto, David-Paul, Maarten	1h
Writing content for the report	David-Paul	4h
Writing content for the report	Maarten	1h
Improving code for Part 1b	Maarten	1h
Writing the report for Part 1	all	4h

Table 1. Contributions.

ACKNOWLEDGMENTS

We would like to express our special thanks of gratitude to our professor Dong Nguyen as well as our lab instructor Davide Dell'Anna who gave us the toolset to do this wonderful project on the topic. Also, we are thankful for lab assistant instructor Marc for giving us advice and sharing their own experience, which helped us in doing research improving our implementation.

Secondly, we would like to thank our families and friends who helped us a lot in finalizing this project within the limited time frame.

REFERENCES

1. 2019-2020 1-GS Methods in AI research (INFOMAIR) 2019. Retrieved October 7, 2014 from https://uu.blackboard.com/webapps/blackboard/content/listContent.jsp?course_id=_122545_1&content_id=_3177786_1