Battleship as a Dialog System
Aaron Brackett, Gerry Meixiong, Tony Tan-Torres, Jeffrey Yu

Abstract

For our project, we built a conversational agent for Battleship using Dialog systems. In this paper, we start off discussing our motivations for choosing this project, and the baseline we started working with. Then we review the historical role of dialog systems in conversational agents. We then move on to explaining our implementation and the dialog system elements we used in crafting our conversational element. We show off some example traces of our conversational agent in action, highlighting the key elements discussed in our implementation. We then conclude with an assessment of our dialog system.

Intro

The game of Battleship has always been a beloved staple of board game culture. As the rise of technology changes how we play board games in the modern age, we were inspired to use what we learned about dialogue system to upgrade the classic board game of Battleship. We are all game enthusiasts and had originally planned on doing some type of natural language project applied to a game. Our original idea was to build a system for text-based adventure games. However, after re-thinking our idea and discussing it at office hours, we decided to build a text-based dialogue system specifically for battleship.

We found a simple baseline implementation of a dialogue system that lets the user play a basic game of Battleship against the computer. This implementation was very basic. The computer opponent had one ship of size 1x1 which the user had to try to hit. This baseline implementation did not allow the user to have their own ships, so the user could only play by guessing the computer’s ships. The user would input row and column numbers, to which the system would respond by telling them if they missed or hit. This baseline implementation was strictly system-initiative and had very simple responses. It was not very fun to play. We decided to use this as our baseline, and improve the system by allowing the user to place their own ships, expanding the size and number of ships, and implementing a robust dialogue system to facilitate natural, engaging game-play.
Background

Dialog systems have become a key component in creating conversational agents. With the likes of Siri, Cortana, and Alexa now at the disposal of millions of consumers, both the understanding and usage of dialog systems have skyrocketed in recent years. At its core, a dialog system represents a computer system that can converse with human-users in a coherent fashion. Making use of components such as a Natural Language Generator and Spoken Language Understanding, they are able to nearly replicate human language in specific systems. In particular, text-based dialog systems have also seen remarkable improvement, with various implementations backed by end-to-end neural nets, hidden markov models, and even regular expressions.

Modern systems have faced criticism because they often lack natural, or human-like, behavior\(^1\). This problem can be alleviated when the dialog system is used for information and control systems such as in travel-booking or reservation systems. We make use of this specificity in creating our dialog system since it pertains only to playing Battleship. A key challenge in understanding “natural language” is having coherence of all sorts of human language, ranging from short, one-line commands to full, complex sentences. In practice, this is often done through the use of a lexicon, parser, and grammar. For the purposes of our project, we chose to use a series of regular expressions.

With regards to conversation and generating responses, there are also various nuances that characterize speech as “human-like.” For example, dialog systems often fall fate to being single-initiative, meaning the computer continuously prompts the user, asking certain questions. The user can only answer these questions, with no opportunity to guide the conversation. In her thesis, Dr. Stent describes a dialogue as “interaction involving two participants, each of whom contributes by listening to the other and responding appropriately”\(^2\). Dr. Fong supports this idea by calling it “the process of communication between two or more parties” and “a joint process”\(^3\). Having mixed-initiative, where both parties involved contribute and direct the conversation, allows for dialog systems to become much more lifelike. We tried to keep this into account in developing our project, allowing the user to ask questions relevant to the game as well as answering those prompted by the computer.

Dialog System Implementation

We introduced a conversational agent to the game of text-based battleship. We started with a baseline version of Battleship from Python Fiddle. It was originally Battleship with only ai ships of one unit length of which you needed to guess the
coordinate. We added player ships and ships of length 3 with differing orientations so it involves a lot more interaction handled by the conversational agent.

This added complexity introduced more user initiative driven dialog acts such as asking at anytime to see where a player’s ships are placed, where a player has fired, at which coordinates ships were hit. To see these user initiated questions see screenshots. Along those lines of user initiative, we implemented some universal commands to start over or get a help menu at anytime. For system initiative driven dialog, our flow was into, to get coordinates for ship placement, and fire coordinates, finally checking for winner at each fire iteration. We wanted to add some grounding responses to allow the player to know they are inputting coordinates correctly.

We also wanted to be coherent and not ask in a redundant robotic way, by using responses showing progression (“Where do you want to attack first?” “Next attack coordinates, Captain?”). We also tapered our prompts when doing a repetitive task like asking for firing coordinates (“Please input a coordinate for attack” “Please input a coordinate” “Next?”). While and after we’ve filled the board we maintain an information state through out of current ships/fires/hits accessible through user initiated questions. There are also some two turn commands like when you ask if you’ve fired at a certain coordinate, from then it will tell you if you have or not, and you can then fire there as a command; This maintains the previous state of question, “where do you want to fire?” meanwhile with the follow-up prompt, so you can input a different coordinate if you want.

Conclusion

In the end, we have succeeded in our goal of creating a dialog system for battleship. Ours is very robust, allowing a multitude of inputs. It takes away the usual
pain of playing text games, which is that if you do not input in exactly what they want you to handle you get an endless sea of “I don’t understand you” or even worse, the game just crashes. Not only this, but we actually ended up adding a bunch of functionality to the game. One of the decisions we made early on was to not display a board to really force us to make the game as robust as possible. With the absence of a board that a player can look at, we had to include the ability to ask about the state of the board and the history of the game. This would be helpful for the eventual future of the game, having it be speech-to-speech. We have eliminated the need to see the board so adding speech would allow for people lying on their beds, people doing chores, and even blind people to be able to enjoy a good game of battleship. Speech recognition comes with its own set of problems that we would need to solve, but, given more time, would definitely be worth it. Because speech tends to be less reliable than text, if we went this route we would probably have to move away from pure regular expressions to a more probabilistic model. But our current product gives us a great jumping off point to get started on these improvements.

(Note for Jeffrey’s Contributions: designed robust coordinate parser coord.py and helped with dialog system design considerations and implementation).

Appendix:
Insert Code
placeShips

Why are you up at this hour playing Battleship? Anyway, I'm Aarrrrrrrrrrron the Battleship Assistant here to help you in anyway possible!
For example, I can tell you where your ships are, where you've fired, and what ships you've hit.
That said, let's play Battleship!
Where would you like to place your ship? a3 a4 a5
I'll go check on the feasibility of coordinates: a3 a4 a5
Success! I'll place your ship now, mate
Where would you like to place your ship? c2 3c row 3 col 4
I'll go check on the feasibility of coordinates: c4 c2 c3
Success! I'll place your ship now, mate
Where would you like to place your ship? d1
That's a great start! Now just tell me two more coordinates to place your ship:
Great! So the last coordinate will be d3, correct? correct
I'll see if I can place your ship at coordinates: d1 d2 d3
[[4, 1], [4, 2], [4, 3]]
Success! I'll place your ship now, mate
Initial target coordinates, captain? where are my ships
You're ships are stationed at the following coordinates:
Row: 3, Column: 2
Row: 3, Column: 0
Row: 0, Column: 4
Row: 2, Column: 1
Row: 2, Column: 3
Row: 2, Column: 0
Row: 0, Column: 3
Row: 0, Column: 2
Where do you want to attack first?
fireMode

Where do you want to attack first? a2
Firing...
Miss...
Miss... from the CPU
A little more thataways, Captain! Please input a coordinate for attack! d4
Firing...
Hit!
Miss... from the CPU
Let's get 'em again, Captain! Please input a coordinate for attack! d3
Firing...
Miss...
Hit! from the CPU
A little more thataways, Captain! Please input a coordinate for attack! c4
Firing...
Hit!
Miss... from the CPU
Another! Where do you want to attack next? b4
Firing...
Miss...
Hit! from the CPU
A little more thataways, Captain! Next attack coordinates, captain? e4
Firing...
Hit!
Hit! from the CPU
Once more, with feeling! Please input a coordinate where have i fired
You have already targeted the following coordinates:
Row: 1, Column: 2
Row: 5, Column: 4
Row: 4, Column: 4
Row: 4, Column: 3
Row: 3, Column: 4
Row: 2, Column: 4

Let's get 'em again, Captain! Where next? which ships have i sunk?
You have hit ships at the following coordinates:
Row: 5, Column: 4
Row: 4, Column: 4
Row: 3, Column: 4
References

