

Chatbot Project: Wordle Chatbot

Wordle is a simple game that I have been playing with my friends and family for the past year. The goal is relatively simple where you have to try and guess a 5-letter word in 6 tries or less. Each letter in a guess is evaluated with the target word and then is displayed back to the user. The user then has to make new guesses using hints from the letters of previous guesses.

When I was trying to decide what to do this project over, I really wanted to do something completely unique. Most chatbots aim to have a conversation with the user to offer support or provide information about a topic but those didn't really excite me all that much. After sending a wordle to a friend one night, I had a vision of a chatbot that could play the game alongside the user and then it would be able give help if the user was to ask for it.

I created this chatbot using Rasa which is an open-source Python framework designed for rule-based chatbots. The ML models are trained with YAML files and the custom actions can be coded using python. In hindsight, working with Rasa was much harder than just building one from scratch. Because of the YAML files, I was unable to use the normal debugging tools in any IDE. I had to essentially run the program and try to trace the dialog to see why it was not working exactly the way I wanted it to. It took hours to bugfix minor syntax errors. Additionally, creating one from scratch would allow me to make a bot that was more flexible with playing wordle.

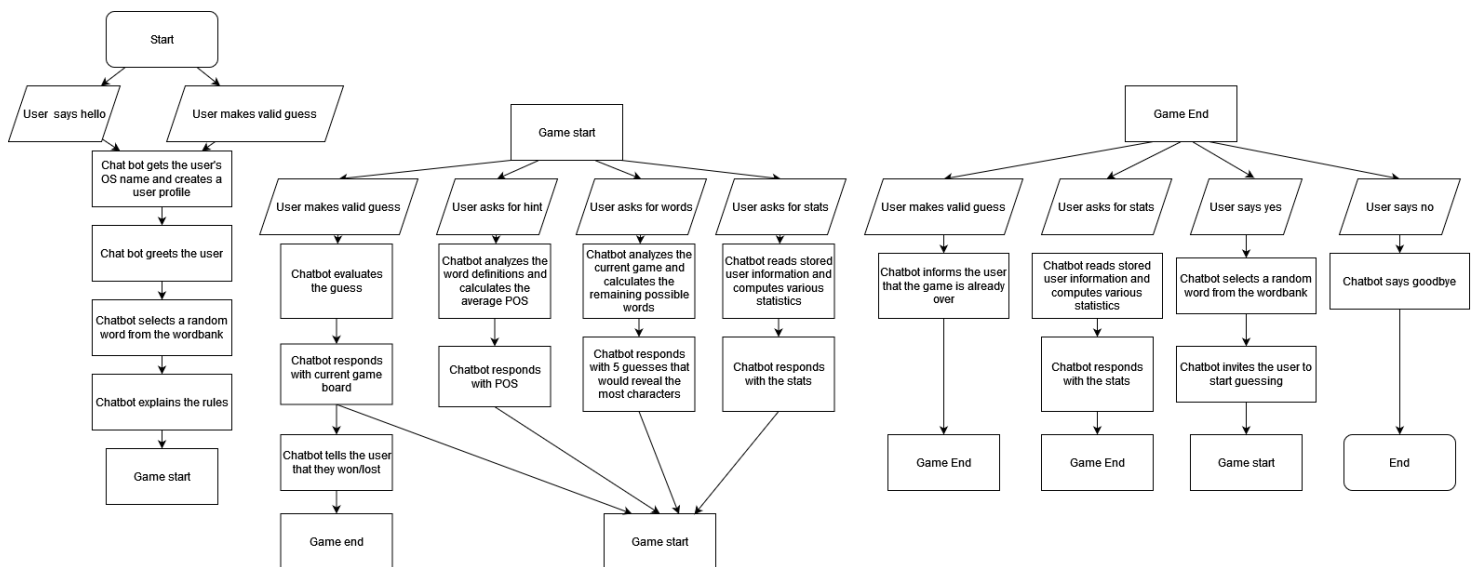
The logic for the chatbot is best described in three points: start, middle, and end. At the start the user can either greet the chatbot or just jump straight into guessing. This is where the bot gets the user information, creates a user model, and sets up the game. If user had begun with a guess, it would go ahead and evaluate that guess. Initially I had wanted the bot to start the conversation as to lead the user into starting a specific way, however due to limitations of Rasa, this would not of been possible.

The middle of the game is the bulk of the chatbot. During this block, the user may make guesses, ask for help, and check their user statistics. With each guess, the chatbot updates the game board with the new guess and displays the progress. Unfortunately, I only had time to implement two hint actions. For

the first hint, the user can ask what the part of speech is for the word and the bot will respond accordingly.

This is calculated by using nltk wordnet and counting POS for each synonym. Whichever POS had the highest count, that was the POS for the word. This probably wasn't the ideal way of doing this but it worked for most words I tested it on. The next action the bot could perform for the user, was that it could run an analysis on the current progress and suggest five words that would reveal the most about the target. The only NLP technique used in this action was a frequency dictionary that counted the frequency of each character in all the words. This section repeats until the user either guesses the correct word or runs out of attempts.

Once the target has been found, the bot will move to the end section. Here the bot asks if the user would like to play again and either starts the game over or exists the conversation. At any point after the game has been started, the user can ask for their wordle stats where it will tell them how well they perform.



Chatbot Logic Diagram

Sample Interaction

```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> hello
Welcome back!
Let me pick out a word real quick.
The goal is to try and guess a word that I have picked out in 6 guesses or less.
Your guess can be any 5-letter word. I will tell you how each character related to the target word.
If any letter is highlighted in yellow, that means that the letter is hidden in the word.
If any letter is highlighted green, then that letter is in the correct location.
If the letter is highlighted red, that letter is not in the target.
Your input -> cramp
c r a m p
Your input -> spoke
c r a m p
s p o k e
Your input -> bowls
c r a m p
s p o k e
b o w l s
Your input -> i need a hind
I'm sorry, I can't seem to find the part of speech for this word.
Your input -> opens
c r a m p
s p o k e
b o w l s
o p e n s
Your input -> mason
c r a m p
s p o k e
b o w l s
o p e n s
m a s o n
Your input -> help, I cant think of any words
I suggest trying dyson, and tyson
Your input -> dyson
c r a m p
s p o k e
b o w l s
o p e n s
m a s o n
d y s o n
Congrats! You found the word! Would you like to play again?
Your input -> no
Thank you for playing! Please come again soon.
Your input -> |
```

Evaluation

Overall, the chatbot works as intended. The bot does a good job at responding with the correct action and handles misclassifications very well. I was able to play it consistently for a long period of time without it breaking. I feel that the bot could have more functionality to assist the user. I had some more ideas but just ran out of time due to how tricky it is to work with Rasa.

Strengths

I personally find it fun to interact with the bot and I feel proud of the complexity of the logic behind it. The bot is able to accurately process user guesses with only a few words that didn't make it into the word bank. The bot is able to suggest words that would reveal the most words by analyzing the current progress and filtering out words from the word bank.

Weaknesses

This bot's biggest weakness is that I built it using the Rasa framework. One of the biggest problems I was having was getting the bot to reliably extract the valid guesses while acknowledging the other intents. If I had built it from scratch, I would have had much more flexibility in how I processed the user input. Another weakness is that the bot is relatively slow at responding but that is due to how Rasa handles all the rules.

Appendix A: Knowledge Base

The knowledge base is essentially any 5 letter word in the English language. I created this list by picking a random website and Wikipedia and recursively crawling over connecting Wikipedia links with BeautifulSoup. I set a time limit of 20 minutes where it would pull each valid English word of 5 characters in length. Then, I had to filter out roman numerals and profanity. Using these, I created a text file to use in the actions code and a YAML lookup table to use for the regex entity extractor.

```
word_bank.txt X
actions > Resources > word_bank > word_bank.txt
1 aalst
2 aalto
3 aarau
4 aaron
5 aback
6 abate
7 abbas
8 abbes
9 abbey
10 abbot
11 abdul
12 abele
13 abets
14 abide
15 abler
16 abner
17 abode
18 abort
19 about
20 above
21 abram
22 abuja
23 abuse
24 abuts
25 abuzz
26 abyss
27 accel
28 accra
29 aches
30 acids
31 acorn
32 acres
33 acted
34 acton
35 actor
36 acute
37 adage
38 adams
39 adana
40 adapt
41 added
```

Word bank text file

```
valid_guess.yml X
data > lookups > valid_guess.yml > version
1 version: "3.1"
2 nlu:
3   - lookup: valid_guess
4     examples: |
5       - aalst
6       - aalto
7       - aarau
8       - aaron
9       - aback
10      - abate
11      - abbas
12      - abbes
13      - abbey
14      - abbot
15      - abdul
16      - abele
17      - abets
18      - abide
19      - abler
20      - abner
21      - abode
22      - abort
23      - about
24      - above
25      - abram
26      - abuja
27      - abuse
28      - abuts
29      - abuzz
30      - abyss
31      - accel
32      - accra
33      - aches
34      - acids
35      - acorn
36      - acres
37      - acted
38      - acton
```

YAML wordlist

Appendix B: Sample User Model

I set up the user models to be based off of the system username. I tried to get the chatbot to prompt for the user name as the conversation starts but, there was no way to do that in Rasa without setting up a server and sending a post request. The user information tracks the user's stats and can be displayed at any time the user asks for it after the game has started at least once. The data is stored in 3 text files. One for the statistics, one for the guess history, and one for the answer history.

```
data.txt M X
actions > Resources > Users > dylan > data.txt
1 actions/Resources/Users/dylan
2 $guess_history_file=actions/Resources/Users/dylan/guess_history.txt
3 $answer_history_file=actions/Resources/Users/dylan/answer_history.txt
4 $system_user_name=dylan
5 $initial_login=11/15/2022, 20:41
6 $most_recent_login=11/15/2022, 23:49
7 $total_num_games=21
8 $total_num_wins=15
9 $correct_guess_1=0
10 $correct_guess_2=1
11 $correct_guess_3=3
12 $correct_guess_4=4
13 $correct_guess_5=4
14 $correct_guess_6=3
15
```

Stored User Information

```
Your input -> how am I doing
Fetching your stats...
Total number of games: 20
Total number of wins: 15
You have a win percentage of: 75%
Average number of guesses to win: 4.333333333333333
Your favorite words are: trace brain track
Your input ->
```

Computed Stats