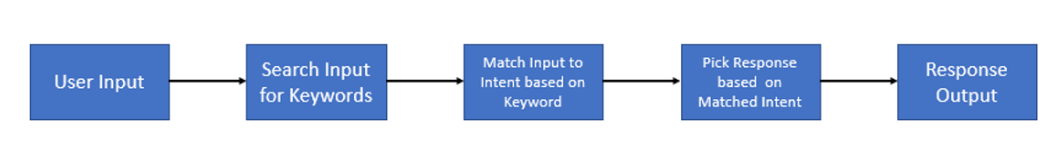
References

Input processing flow chart



install the NLTK (natural language toolkit) library in the terminal

* pip install nltk

import Python modules

* import nltk
* import re
  + re module in Python provides support for working with regular expressions, allowing you to search, match, and replace text based on specific patterns.
  + Functions that can be used in the re module:

re.match(): Checks for a match at the beginning of a string.

import re

result = re.match(r"hello", "hello world")

print(result.group()) # Output: hello

re.search(): Searches the entire string for a match.

result = re.search(r"world", "hello world")

print(result.group()) # Output: world

\d+ is a pattern used to match one or more digits.

\d: 0 to 9

+ makes sure the preceding element appears one or more times in the string.

Matches any single digit character (0–9).

re.findall(): Finds all matches in the string and returns them as a list.

result = re.findall(r"\d+", "123 abc 456")

print(result) # Output: ['123', '456']

re.sub(): Replaces occurrences of a pattern with a different string.

result = re.sub(r"\d+", "#", "123 abc 456")

print(result) # Output: # abc #

re.compile(): Compiles a pattern into a regex object for reuse.

If you use the same pattern multiple times, re.compile() is more efficient.

import re

pattern = re.compile(r"\d+")

print(pattern.findall("123 abc 456.")) # Output: ['123', '456']

match = pattern.search("Price: 123 dollars")

print(match.group()) # Output: '123'

* from nltk.chat.util import Chat, reflections
  + ‘reflections’ automatically swaps certain words. For eg, "I" becomes "you," "am" becomes "are," etc.
  + The ‘chat’ function helps initiate a conversation based on these patterns.
  + ‘pairs’ is list of patterns and corresponding responses.
    - from nltk.chat.util import Chat, reflections

pairs = [

[r"hi|hello|hey", ["Hello!", "Hi there!"]],

[r"my name is (.\*)", ["Hello %1, nice to meet you!"]],

[r"quit", ["Goodbye!"]]

]

chatbot = Chat(pairs, reflections)

chatbot.converse()

Download necessary NLTK datasets

* nltk.download('punkt')
  + ‘punkt’ is used for sentence splitting and word tokenization.
* nltk.download('averaged\_perceptron\_tagger')
  + this model tags words with their corresponding grammatical roles, such as noun, verb, adjective, etc.

define a class for your chatbot

* The ‘respond’ method takes user input as an argument and uses the ‘chat’ object to return a corresponding response.
* the \_\_init\_\_ function is a special method used to initialize objects in a class.
  + class SimpleChatbot:

def \_\_init\_\_(self, pairs, reflections):

self.chatbot = Chat(pairs, reflections)

def start\_chat(self):

print("Chatbot is ready to chat! Type 'quit' to exit.")

self.chatbot.converse()

initialise chatbot

* chatbot = SimpleChatbot(pairs, reflections)

chatbot.start\_chat()

create function

* def chat\_with\_bot():

print("Hi, I'm your chatbot. Type 'quit' to exit.")

while True:

user\_input = input("You: ") # Take input from the user

if user\_input.lower() == 'quit':

print("Chatbot: Bye! Have a great day!")

break

response = chatbot.respond(user\_input) # Get chatbot response

print(f"Chatbot: {response}") # Print the chatbot's response

start

* chat\_with\_bot()