UiPath Chatbot Testing

Problem Statement:

Chatbot Testing for every release is painful for any developer or testing team. With UiPath, you can automate the chatbot testing.

Solution:

Use UiPath to design an easy-customizable solution that interacts with the chatbots on different platforms, just like the humans do. The prototype demonstrates this by testing two chatbots across two different platforms depending on which one is enabled as a part of the Config; a Skype Chatbot named “Horoscope” as well as a OrderDrink chatbot on AWS Lex.

Components for Prototype:

RE Framework has been used.

- Instead of Queues, datatable is used. The transaction item has been modified accordingly to data row.
- Transaction Status, Error Type, Error Message is being reported into the Output excel generated by the chatbot.

Config File:

- Solution is mainly controlled by Config.xlsx.
- Chatbot Application will indicate what is the application to be tested – Skype or AWS or Facebook etc.
- Depending on whether it is a chatbot on Skype Web or AWS Lex that you want to test out, update the ChatbotApplication value in Config file.
- For Lex, AWSLex_ChatbotName should be the name of the Lex chatbot the solution will try to test.
- Likewise for Skype Web, Skype_ChatbotName should be the name of the Skype chatbot the solution will try to test.
- Credential Assets for the platform like Skype, AWS, Facebook, etc. can be added as needed in a separate worksheet.

Utterances Source File:
The developer will place all of the utterances to be tested in an excel spreadsheet along with the expected responses. The worksheet name is of the format - “ChatbotApplication”_”ChatbotName”

Example is AWSLex_OrderDrink where OrderDrink is the name of the chatbot in AWS Lex platform.

This is placed in “Data\” directory.

Copy the template from one of the existing sheets to a new tab to add the utterances and its expected response.

ChatbotOutputResults:

Uipath will then write each utterance to the chatbot and capturing its responses in the excel sheet. The actual response from the chatbot will be compared to the expected response based on the matching strategy.

For the prototype, the matching strategy choices are kept either as:

1. Equals: which means the actual response should be an exact match with the expected response.
2. Starts With: Verifies whether the actual response starts with the text

mentioned in the expected response field.

This can be extended to use Ends With, Contains and any other expressions by means of which a response is expected to be validated as correct.

This is also placed in the “Data\” directory.

For now it is already present, hence it will overwrite the Sheet1.

Skype_LoginWeb:
Used to login to Skype using Chrome in InPrivate Mode. It passes a browser object as an output that is then used across the project.

Skype_StartChat:
Used to find the contact and start the chat.

Skype_ChatbotTesting:
Enters the utterances one by one into the chatbot and captures results.
The response for this chatbot has an idx value that is the current (TransactionNumber + 2). Selectors for this may have to be modified depending on the specific skype chatbot configured

Skype_LogoutWeb:
Logs out of Skype and closes the browser.

AWSLex_Login:
Used to login to AWS Lex using Chrome in InPrivate Mode. It passes a browser object as an output that is then used across the project.

AWSLex_StartChat:
Used to navigate across AWS Console into the specific chatbot and enable the test console for the chatbot.

AWSLex_ChatbotTesting:
Enters the utterances one by one into the chatbot and captures results.

AWSLex_LogoutWeb:
Logs out of AWS Lex and closes the browser.

Enhancements

1. Extension to Other Platforms

The concept can be extended to work with any chatbot platform:
AWS, Facebook, Slack, etc. including any inhouse chatbots developed for your enterprise or for your customers

2. Extension to Other Response Types

The prototype only compares response text with expected text. However, the solution can be extended to compare chatbot responses that are buttons, images, files or any other response types

3. Aid Troubleshooting and Analysis

- Chatbot response time can be additionally measured and baselined, automated performance testing can be executed during expected peak time or post heavy releases.
- If an extract is available that maps the responses to the intents configured in the chatbot, then the troubleshooting time taken by the developers to identify which intent was incorrectly picked up for failed cases can be eliminated.
- A dashboard can be created to analyze each performance (in terms of accuracy, response time, intents with most failed cases, most incorrectly picked intents, etc) as well as analze the trend of is performance over time as new features are added in subsequent releases.

4. Scalability

With orchestrator and multiple robot servers, the UiPath process can be configured

- to either simulate concurrent users testing a chatbot
- or to test multiple chatbots at once one on each server
- or test the same chatbot but with different utterances per robot server to lower the overall testing TAT.

5. Complete the end to end chatbot testing with UiPath Robots for Knowledge Extraction

Add provider robots that generates the utterances from either of these sources:

- Existing Chatbot configurations
- Conversation logs of the chatbot
- Email conversations with IT Team
- ITIL requests raised for IT Team