SwaggerBot

Generating Chat Bots from Web API Specifications

Mandana Vaziri, Louis Mandel, Avi Shinnar, Jerome Simeon, Martin Hirzel

IBM T. J. Watson Research Center

Onward’17
Problem Statement

• Chat bots are becoming ubiquitous
  – Provide customer care by phone or online
  – Industries: banking, healthcare, telecom, retail, ...
  – They can query resources and take actions via Web APIs

• Chat bots are difficult to maintain and modify
  – Require traditional software and machine learning skills
    • Natural Language Understanding (NLU)
  – Rapidly changing business needs requires modifications

• Objective
  – Generate chat bots automatically
  – Focus on IBM Watson Conversation Service
  – Generate chat bots from Web API specifications
From Web APIs to Chat Bots

• Generate chat bots from web APIs
  – Start from a Swagger specification
  – Generate a chat bot that can:
    • Help the user navigate an API
    • Make calls conversationally
  – Examples:
    Human: Please translate the text `la vie` to target language `en`
    Bot : `Life`

    Human: What is the weather in NYC?
    Bot : `windy`
Swagger

• Standard for specifying Web APIs
  – Based on JSON
  – Specifies endpoints, operations, parameters
  – Has descriptions and summaries

• Ecosystem of tools generated from Swagger
  – API documentation
  – UI interface, tryme
  – Client Software Development Kits (SDKs)
  – Server stubs
  – Automated tests
-> Swaggerbot adds chat bots to this ecosystem
Challenges

• Swagger is insufficient
  – Missing documentation
  – Inaccurate
  – Missing logical relationships
  – Need multiple API calls to make one call
  – Need to pass output of one call to another
  – API may need specialization for an end-user’s needs
    • Example: Translation from French to English, not just any language

• Need tuning to generate a useful bot
  – How can we allow the specification to be tuned by a bot designer?
Approach

- Generated bot allows live programming
  - In dialog tuning: bot designer can tune bot by conversation
  - Analogy: Spreadsheet interface
    - End-user interacts with the table
    - Advanced user can create scripts and make them available
  - Bot designer can create higher-level actions
    - In a conversational manner
    - Make them available to end-user in the same interface

Your words just changed my code!
Example: Watson Language Translator

```json
{
  "swagger": "2.0",
  "info": {
    "version": "2.0.0",
    "title": "Language Translator"
  },
  "securityDefinitions": {},
  "security": [],
  "basePath": "/language-translator/api",
  "paths": {
    "/v2/models": {
      "get": {
        "tags": [
          "models"
        ],
        "operationId": "listModels",
        "summary": "Lists available standard and custom models by source or target language",
        "description": "",
        "consumes": [
          "application/x-www-form-urlencoded"
        ],
        "produces": [
          "application/json"
        ],
        "parameters": [
          {
            "name": "source",
            "in": "query",
            "description": "Filter models by source language."
          },
          {
            "name": "target",
            "in": "query",
            "description": "Filter models by target language."
          }
        ]
      }
    }
  }
}
...
Hi, SwaggerBot is here to help you explore the language translator.

What would you like to do? To begin name an action or resource.

Please identify the text "la vie est belle".

You chose: identify. Calling now.

Please translate the text "la vie est belle" from source language "fr" to target language "en".

You chose: translate. Calling now.

Translate GET text "la vie est belle" source "fr" target "en" accept "text/plain".

"Life is beautiful!"

Name this action

What would you like to name this action?

Englify

Thanks, action englify created.

Englify but now please the text should be "le monde est beau".

You chose: englify. Calling now.

Translate GET text "le monde est beau" source "fr" target "en".

"The world is beautiful!"
It’s a red ball!
version 2016-05-20
{
  "images": [
    {
      "faces": [
        {
          "age": {
            "max": 24,
            "min": 18,
            "score": 0.394433
          },
          "face_location": {
            "height": 815,
            "left": 0,
            "top": 0,
            "width": 1413
          },
          "gender": {
            "gender": "FEMALE",
            "score": 0.0
          }
        }
      ]
    }
  ]
}
Features

• API Exploration and calling
  – Ability to walk user through parameters
  – Swagger extension to specify relation between parameters
    • Example:
      – translate needs one of \{model_id\} or \{source, target\}
  – Swagger extension for call chaining
    • Example:
      – Obtaining source language when missing during translate
  – Swagger extension for data shaping
    • Using Jsonpath to specify how the output of a call should be shaped
Features

• Direct calls
  – All parameters given conversationally in a few lines
  – Use WCS frames to help fill out missing parameters
    • Parameters can be provided in any order
  – Background context kept from one call to the next
  – Ability to set fields independently of calls

• Tuning mode
  – Ability to name actions and add synonyms
  – Swagger updated
Watson Conversation Service (WCS)
WCS Workspace

• Intents and entities
  – Intents capture verbs or actions
  – Entities capture nouns or objects
  – Their presence detected by NLU

• Dialog
  – Finite state machine defining the logic and flow of chat
  – Each state is a conditional
    • When an intent or entity is detected, then state can be updated

You cannot reduce me to an FSM!
WCS Workspace UI
SwaggerBot Prototype

• Compile Swagger to a workspace and app
  – Endpoints become intents
  – Operations and parameters become entities
  – Names used for classifier training data
  – Descriptions and summaries used for help dialog

• Support for tuning and live programming
  – Action creation
    • New intents added to underlying workspace
  – Synonyms added to enrich NLU classifier
  – Workspace uploaded without stopping the chat bot

• Case studies
  – On a variety of APIs
  – Automatically downloaded and compiled ~1070 APIs from APIGuru.
Future Work

• Conversational data shaping
  – Passing Json data between calls
  – Manipulating Json via conversation

• Serverless implementation
  – Based on IBM Functions
  – Functions actions for pre/post processing steps
  – No need for a server for the app

This talk is almost over!
Conclusion

• Swaggerbot is a conversational programming language
  – Easier to use than curl
  – Fuzzy syntax, but requires precision for passing parameters

• What tools/concepts from PLSE can be reused here?

Thank you!