# Conversational Recommender System Project Instruction

*All the previous capstone **weekly meeting slides** can be found in this folder:*
*https://drive.google.com/open?id=13HpdSRvGXE2o65obFEO4XRMotlKH9g5E*

*All the **documentation** for the project can be found in this folder:*
*https://drive.google.com/drive/folders/1Z7IQ37hP4zbmX1uilBK-VBJrxUnq7dJP?usp=sharing*

*The **Final Design Specification** document can be found using this link:*
https://docs.google.com/document/d/1IqQp5szMM--H6LMn1vEntEDKenbILv2izSJUxu0kGqk/edit?usp=sharing

## Project code

**All data files will be located at ../data/*

### 1. Data preprocessing & Generation

*This file processes the Toronto business data and the review data, filtered out businesses that are non-restaurants by identifying the selected keywords contained in the restaurant categories.*

**Data preprocessing code:**

- data_processing.ipynb

**Imported Data files:**

- toronto_reviews.csv
- businesses_final_toronto.csv

**Cleaned exported data files:**

- Cleaned_Toronto_Reviews.json
- Cleaned_Toronto_Business.json

### 2. Project data generation (for Recommender System)

*Run following code for data files generation:*

```
python projectData_generation.py –data_dir ../data/ --data_name Cleaned_Toronto_Reviews.json
```

*The current application for the recommender system setting is item-based to make user-item preference predictions. The code currently leverages item-keyword information stored in vector space using TF-IDF to compute the item-item similarities matrix. Therefore, the current item similarity matrix is generated by using the item-keyword matrix.*

**Data file generations code:**

- projectData_generation.py

**Imported data files:**

- Cleaned_Toronto_Reviews.json

**Generated data files:**
- Dictionaries that map item id to their attributes:
    - icDictionary.json
    - ipDictionary.json
    - isDictionary.json
- Dictionaries that maps items id to their distance to each intersection:
    - idDictionary_yongefinch.json
    - idDictionary_bloorbathurst.json
    - idDictionary_spadinadundas.json
    - idDictionary_queenspadina.json
    - idDictionary_blooryonge.json
    - idDictionary_dundasyonge.json
- Other relevant matrices
    - rtrain.npz
    - icmatrix.npz
    - IKbased_II_similarity.npy
    - UI_prediction_matrix.npy

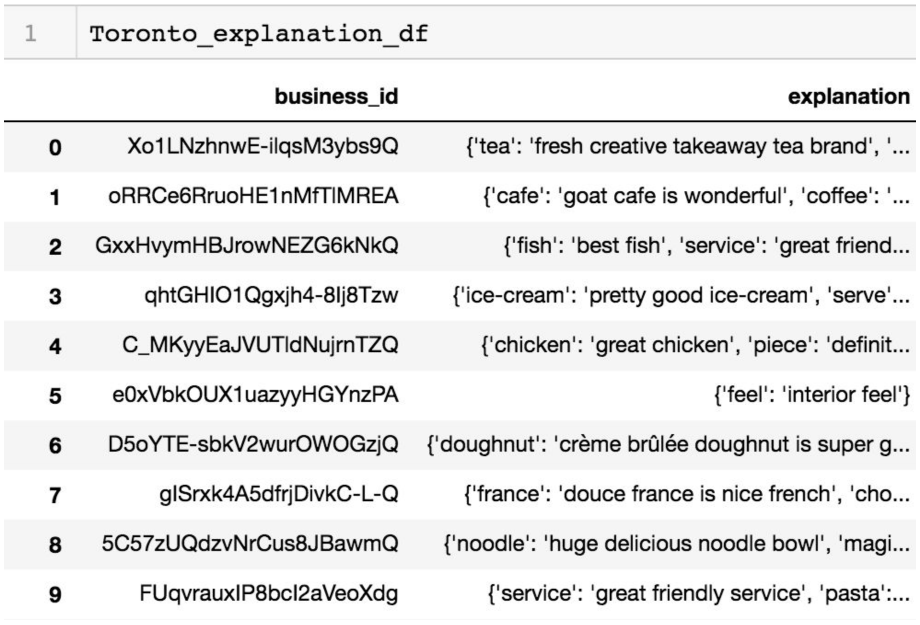
## 3. Explanation generation

*This code reads in the Toronto business data file and generates 3 explanation phrases for each business. Detailed logic behind the generation of the explanation phrases can be found in the Explanation documentation.*

**Data preprocessing code:**
- data_processing.ipynb

**Imported data files:**
- Export_TorontoData.json

**Exported explanation data file:**
- Toronto_explanation.json



```
Toronto_explanation_df
```

| | business_id | explanation |
|---|---|---|
| **0** | Xo1LNzhnwE-ilqsM3ybs9Q | {'tea': 'fresh creative takeaway tea brand', '... |
| **1** | oRRCe6RruoHE1nMfTlMREA | {'cafe': 'goat cafe is wonderful', 'coffee': '... |
| **2** | GxxHvymHBJrowNEZG6kNkQ | {'fish': 'best fish', 'service': 'great friend... |
| **3** | qhtGHIO1Qgxjh4-8Ij8Tzw | {'ice-cream': 'pretty good ice-cream', 'serve'... |
| **4** | C_MKyyEaJVUTldNujrnTZQ | {'chicken': 'great chicken', 'piece': 'definit... |
| **5** | e0xVbkOUX1uazyyHGYnzPA | {'feel': 'interior feel'} |
| **6** | D5oYTE-sbkV2wurOWOGzjQ | {'doughnut': 'crème brûlée doughnut is super g... |
| **7** | glSrxk4A5dfrjDivkC-L-Q | {'france': 'douce france is nice french', 'cho... |
| **8** | 5C57zUQdzvNrCus8JBawmQ | {'noodle': 'huge delicious noodle bowl', 'magi... |
| **9** | FUqvrauxIP8bcI2aVeoXdg | {'service': 'great friendly service', 'pasta':... |

## 4. Conversational Recommender System API

*The file reads in all the generated data files, runs the recommender system and connects it with an API. Different functions are enabled through different endpoints. Details for the design of these endpoints and user-system interaction(critiquing) logic can be found in the API documentation & Critiquing documentation.*

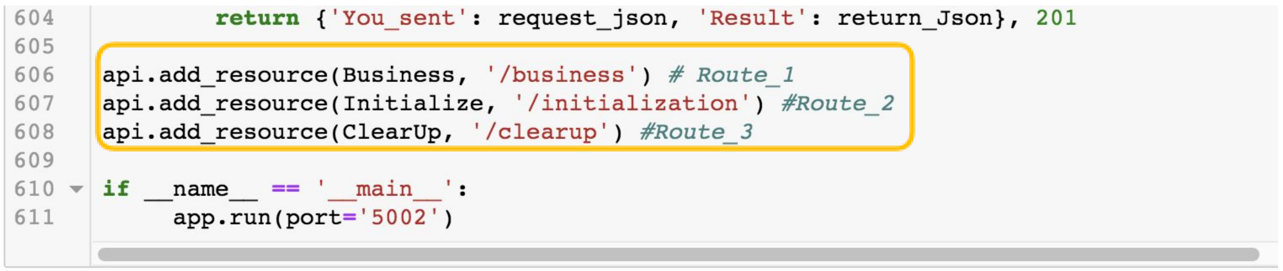
**API code:**

- convSys_API.ipynb

**Imported Data files:**

- Toronto_explanation.json
- All the data files generated from *Section 2*

*Currently, the project uses 3 endpoints to manage the user-system interactions*



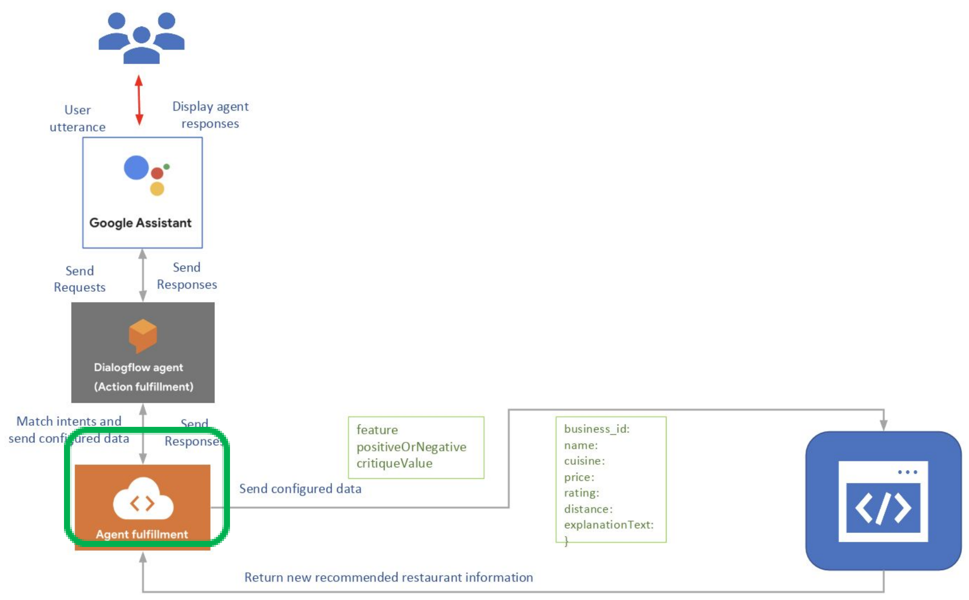
```
        return {'You_sent': request_json, 'Result': return_Json}, 201

api.add_resource(Business, '/business') # Route_1
api.add_resource(Initialize, '/initialization') #Route_2
api.add_resource(ClearUp, '/clearup') #Route_3

if __name__ == '__main__':
    app.run(port='5002')
```

## 5. Conversational Recommender System Middle tier (fulfillment agent)

*This part of the code handles the middle tier of the project, which matches the user's intents to the correct response, and sends retrieve the corresponding responses for the intent.*



**Fulfillment code:**

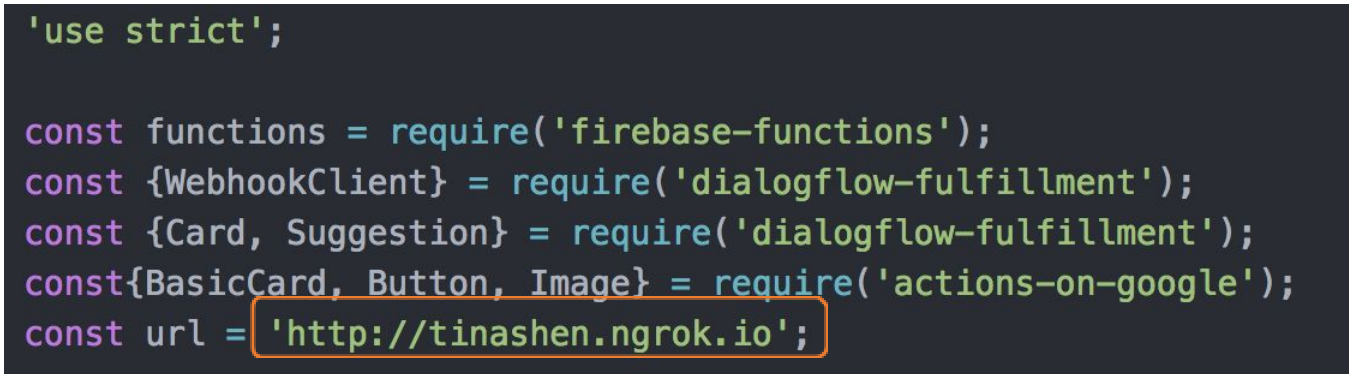
- index.js

**Using ngrok for connecting the API as webhook for the conversational system:**

Initial Setup:

In order to let our local API be connected to an external network, the team used ngrok for this connection. The circled section needs to be changed for, as it is the ngrok deployed webhook address used by the group.

Please follow the ngrok documentation to set up your personal account and token.

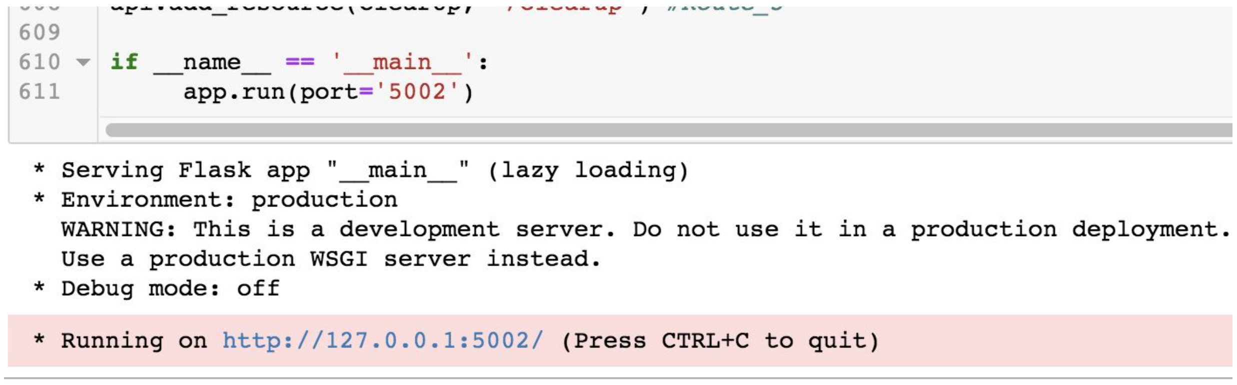


```
'use strict';

const functions = require('firebase-functions');
const {WebhookClient} = require('dialogflow-fulfillment');
const {Card, Suggestion} = require('dialogflow-fulfillment');
const{BasicCard, Button, Image} = require('actions-on-google');
const url = 'http://tinashen.ngrok.io';
```

Local API connection:

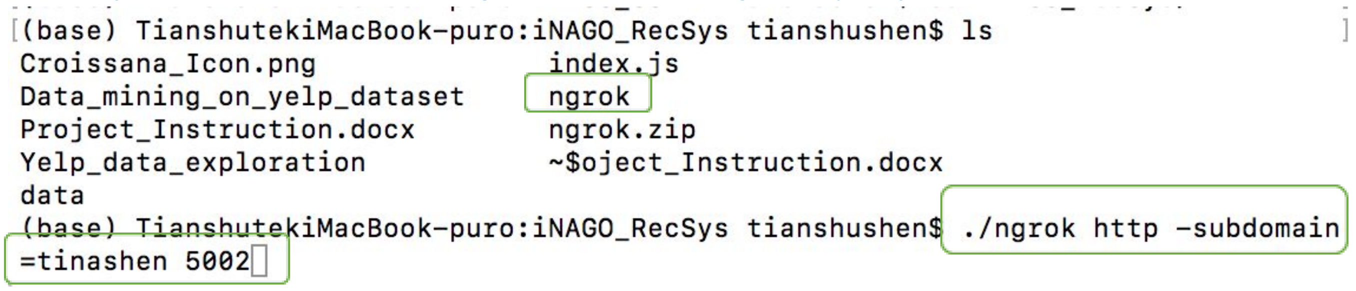
- *Run the API on localhost (5002 as indicated in the screenshot below)*



```
if __name__ == '__main__':
    app.run(port='5002')
```

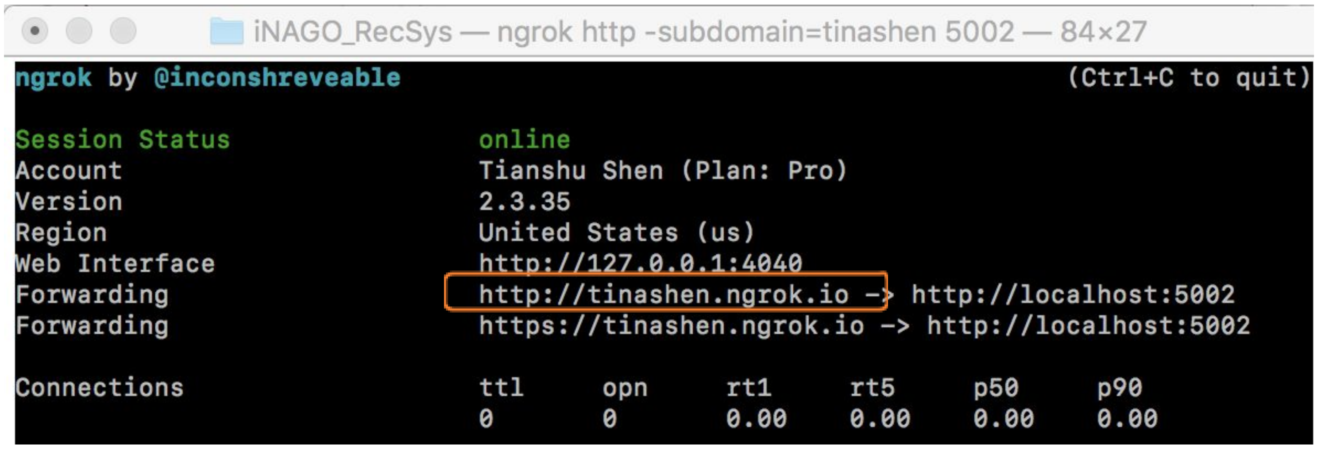
```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5002/ (Press CTRL+C to quit)
```

- *After the API has been running at localhost, run ngrok at your folder*



```
(base) TianshutekiMacBook-puro:iNAGO_RecSys tianshushen$ ls
Croissana_Icon.png              index.js
Data_mining_on_yelp_dataset     ngrok
Project_Instruction.docx        ngrok.zip
Yelp_data_exploration           ~$oject_Instruction.docx
data
(base) TianshutekiMacBook-puro:iNAGO_RecSys tianshushen$ ./ngrok http -subdomain
=tinashen 5002
```

- *Ngrok will connect the API to the external network, select the following address to put into index.js file*



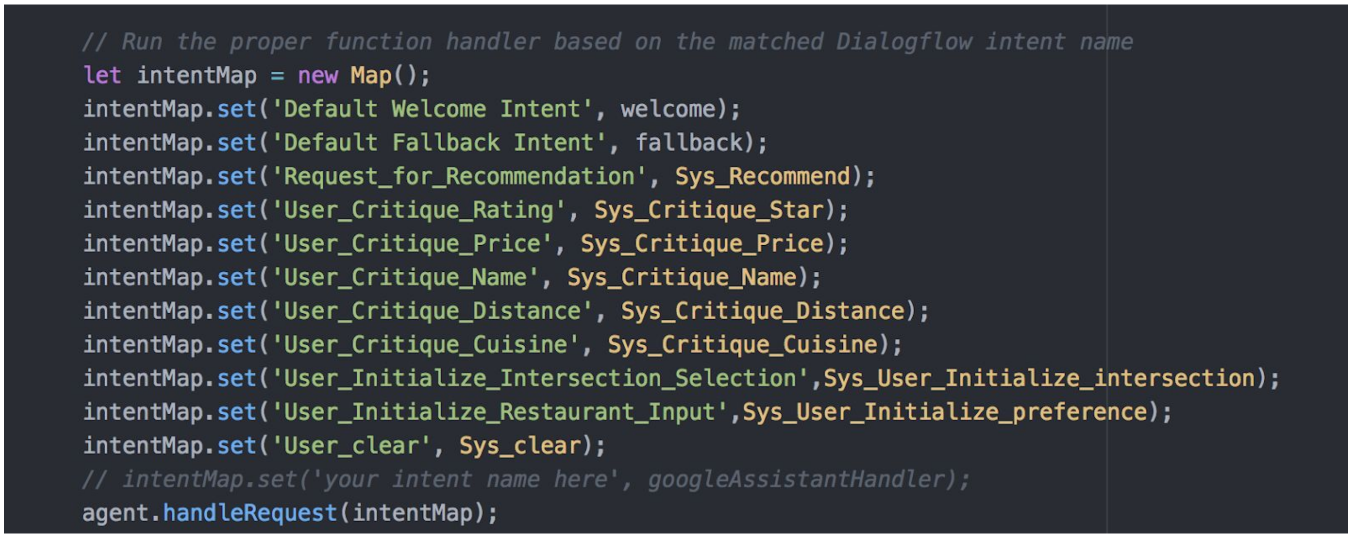
```
iNAGO_RecSys — ngrok http -subdomain=tinashen 5002 — 84×27

ngrok by @inconshreveable                                          (Ctrl+C to quit)

Session Status                online
Account                       Tianshu Shen (Plan: Pro)
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://tinashen.ngrok.io -> http://localhost:5002
Forwarding                    https://tinashen.ngrok.io -> http://localhost:5002

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

- *There are functions that maps the google dialog flow intents to the external API we created, this section in the file indicates the mapping of each intent to the function*



```
// Run the proper function handler based on the matched Dialogflow intent name
let intentMap = new Map();
intentMap.set('Default Welcome Intent', welcome);
intentMap.set('Default Fallback Intent', fallback);
intentMap.set('Request_for_Recommendation', Sys_Recommend);
intentMap.set('User_Critique_Rating', Sys_Critique_Star);
intentMap.set('User_Critique_Price', Sys_Critique_Price);
intentMap.set('User_Critique_Name', Sys_Critique_Name);
intentMap.set('User_Critique_Distance', Sys_Critique_Distance);
intentMap.set('User_Critique_Cuisine', Sys_Critique_Cuisine);
intentMap.set('User_Initialize_Intersection_Selection',Sys_User_Initialize_intersection);
intentMap.set('User_Initialize_Restaurant_Input',Sys_User_Initialize_preference);
intentMap.set('User_clear', Sys_clear);
// intentMap.set('your intent name here', googleAssistantHandler);
agent.handleRequest(intentMap);
```

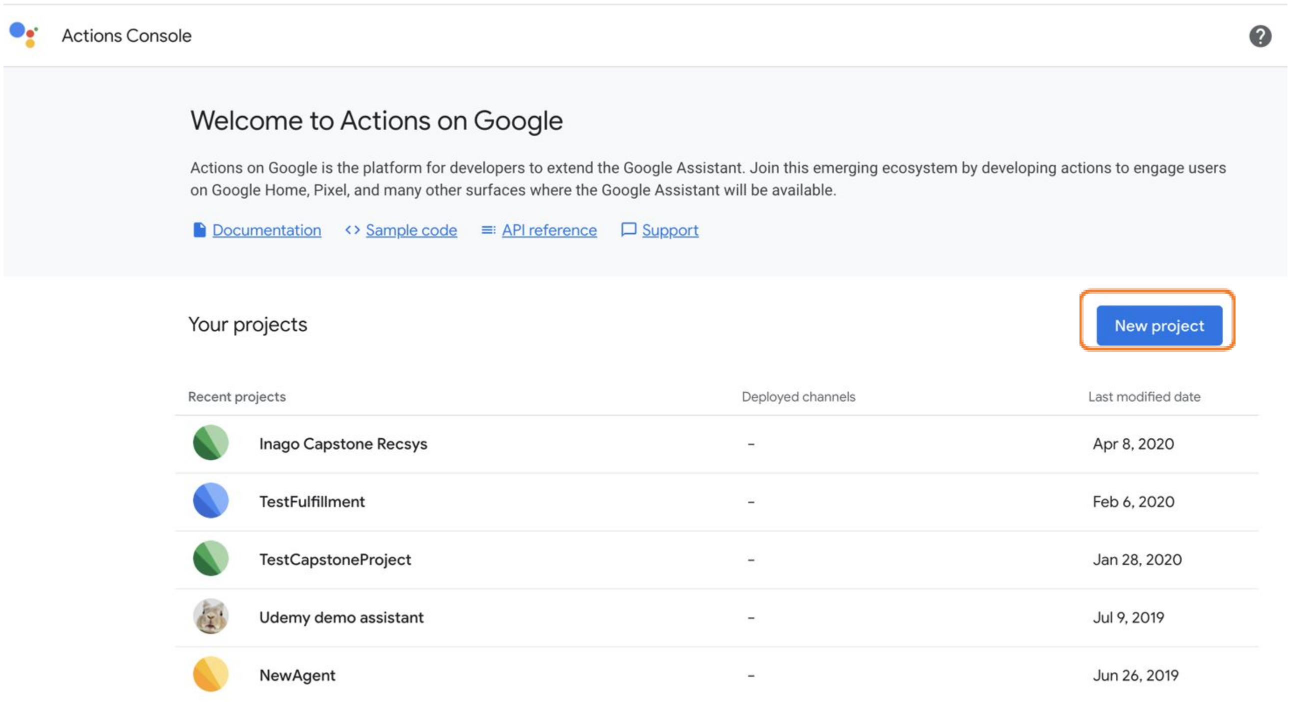
## 6. Conversational Recommender System front end (Google Dialog Flow agent)

*Use the following steps to import the capstone Google Dialog Flow project.*

**Dialogflow project to import:**

- Inago-Capstone-Recsys.zip

1. *To import an existing Google Dialogflow project, create a new project in Actions Console*



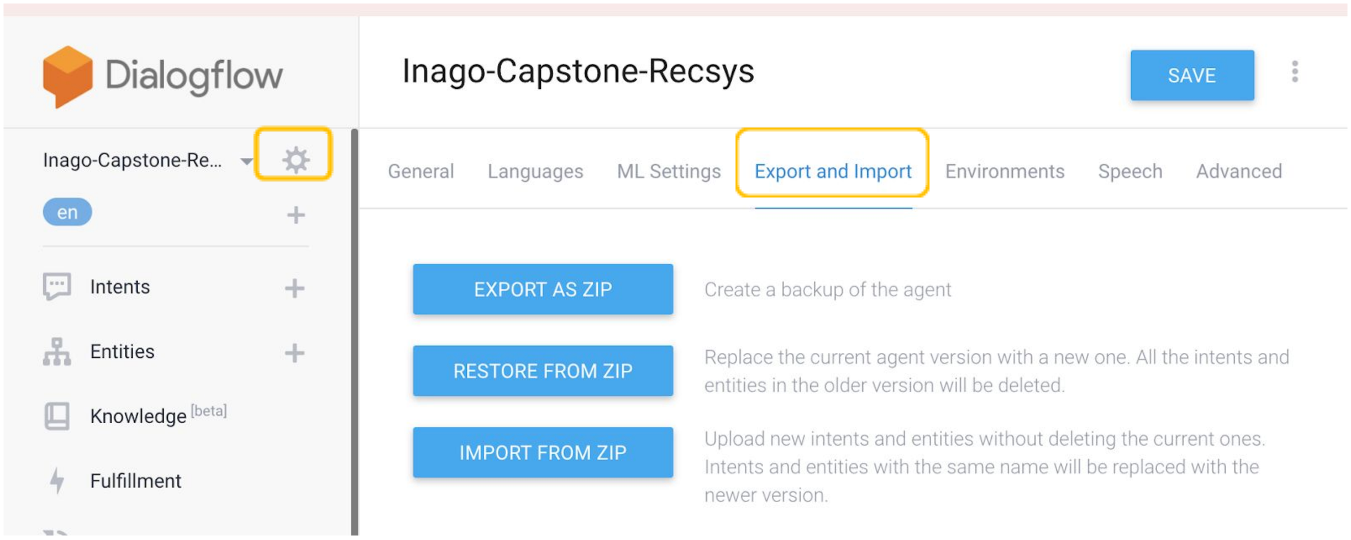
Welcome to Actions on Google

Actions on Google is the platform for developers to extend the Google Assistant. Join this emerging ecosystem by developing actions to engage users on Google Home, Pixel, and many other surfaces where the Google Assistant will be available.

Your projects

| Recent projects | Deployed channels | Last modified date |
|---|---|---|
| Inago Capstone Recsys | - | Apr 8, 2020 |
| TestFulfillment | - | Feb 6, 2020 |
| TestCapstoneProject | - | Jan 28, 2020 |
| Udemy demo assistant | - | Jul 9, 2019 |
| NewAgent | - | Jun 26, 2019 |

2. *Open project in Dialogflow, navigate to settings, Export and Import section and select Import from Zip. Then import the Inago-Capstone-Recsys.zip*



EXPORT AS ZIP — Create a backup of the agent

RESTORE FROM ZIP — Replace the current agent version with a new one. All the intents and entities in the older version will be deleted.

IMPORT FROM ZIP — Upload new intents and entities without deleting the current ones. Intents and entities with the same name will be replaced with the newer version.

3. *Follow Google Assistant documentation to learn the basic functions and for the dialog flow project.*

## 7. Additional files

*These files contain the research code for the project but do not directly contribute to the operation of the conversational recommender system.*

**allMethods_CrossValidation.ipynb:**
This research code compares all different algorithms the group considered for the recommender system. The cross-validation process was performed. All the functions were not formatted, the computation time is long. However, it gives a brief overview of all the different attempts the group used for the recommender system.

**algorithmSelection_userStudy.ipynb**
This research code was used for computing the first user study operated by the group. Where the users will be exposed with 5 different algorithms used for making restaurant recommendations.

**similarityCalculation_analysis.ipynb**
This file contains the research code for analyzing different ways of computing the similarity matrices for user-based and item-based collaborative filtering methods for restaurant recommendations.