User Guide

Smart Night Safari Chatbot

25/08/2019 | v1.0

Team name: NUS-ISS F4

Ma Weizhong, Xu Kaixin, Yu Xiaoxi, Zhang zhiling

CA project for Cognitive System Course in NUS-ISS Intelligent System

Contents

Our Approach

We implement our chatbot upon DialogFlow and we extended a fulfillment in python. After the dialogflow agent recognizes the intent and entities from frontend query, it sends a request to our backend fulfillment. The fulfillment is divided into 2 modules, 1 of them deals with json data, we implemented a fuzzy matching method using synonym library in nltk wordnet, this increase the recall of the successful retrieval while not hurting the accuracy; the other module uses TF-IDF method and cosine similarity to match fallback intents with FAQ data from Night safari website.

Requirements

1. **Dialogflow**: API to retrieve a set of trained intents and entities from user's continuous queries.
2. **Google actions**: API to deploy our project on google assistant interface, enabling rich and pure text response.
3. **Ngrok**: Expose local server address to a remotely accessible Internet url.

Dependencies

4. Dialogflow
5. Anaconda3
6. Python frameworks: flask, nltk, ...
7. Ngrok

Supported User Interfaces

1. Google Assistant mobile app (Recommended), on IOS 9.0+ iPhones/ Android 5.0+ phones, etc.;
2. Actions console simulator;
3. Web demo, viewing on Google Chrome version 74 and above.

Deployment

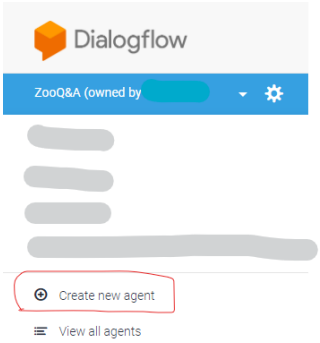
Prerequisites

1. Install anaconda3 from https://www.anaconda.com/distribution/ correspond to your OS version;
2. Download ngrok binary from https://ngrok.com/download correspond to your OS version;
3. Both installations above should be added to your environment before starting our product.
4. Install google assistant app on your mobile device. (optional)

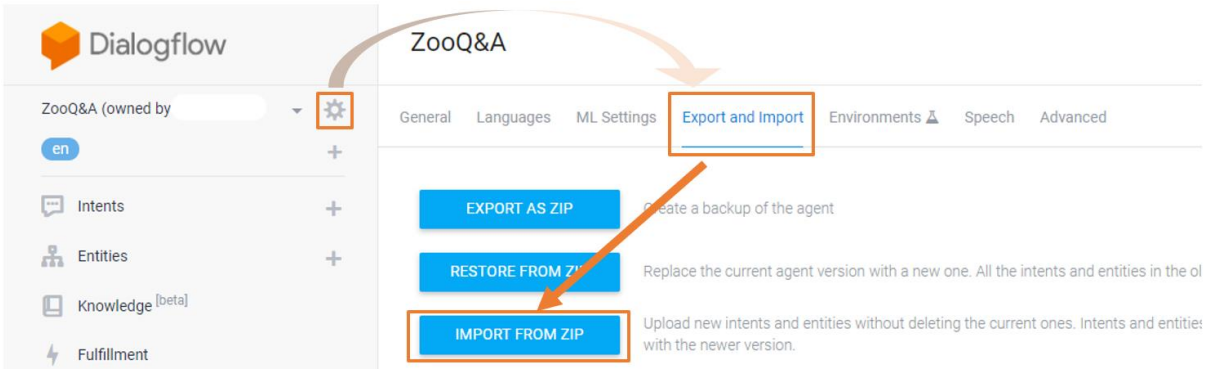
Configure Dialogflow

1. Git clone our project repository to your local machine, by either run:
   `git clone https://github.com/harry0916/NUS_Chatbot_F4.git`
   or simply direct to our git repo and download it as a zip.
2. [Optional], if you don't have a google account, please create one here: https://dialogflow.com/_d/signin?continue=https%3A%2F%2Fdialogflow.com%2F. skip this step if you have signed up already;

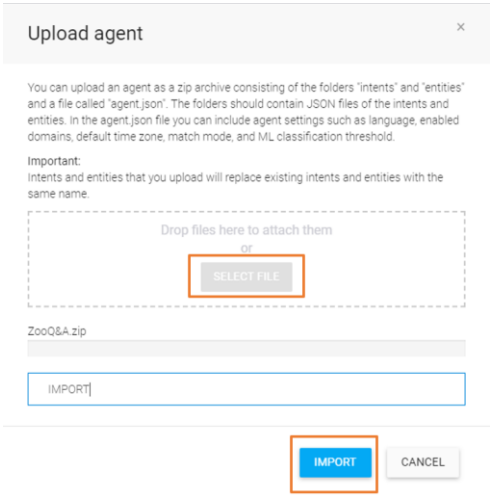
3. Log in to the dialogflow console: https://console.dialogflow.com/api-client/#/login;
4. Create a new agent on the up-left side panel, name it as whatever you like:



5. Click gear button in new agent, select “Import From Zip” option under “Export and Import” panel:



6. Click “Select File” and select “ZooQ&A.zip” file under `[project root]/SystemCode/` directory. Then follow the displayed instructions to proceed importing:

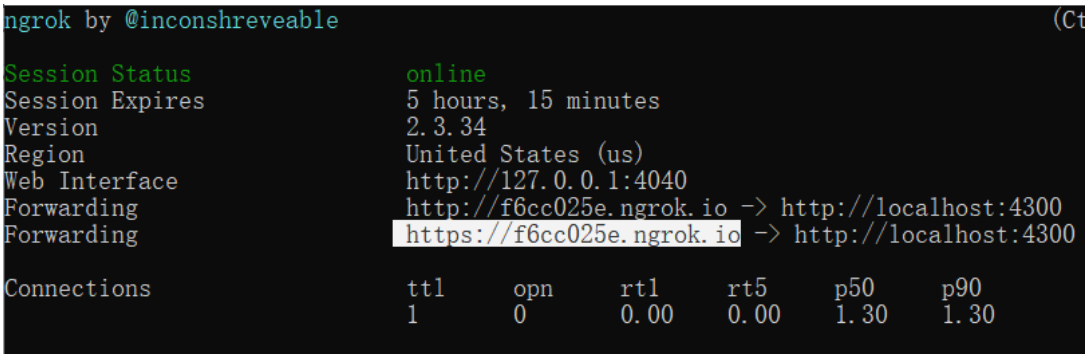


Run backend script

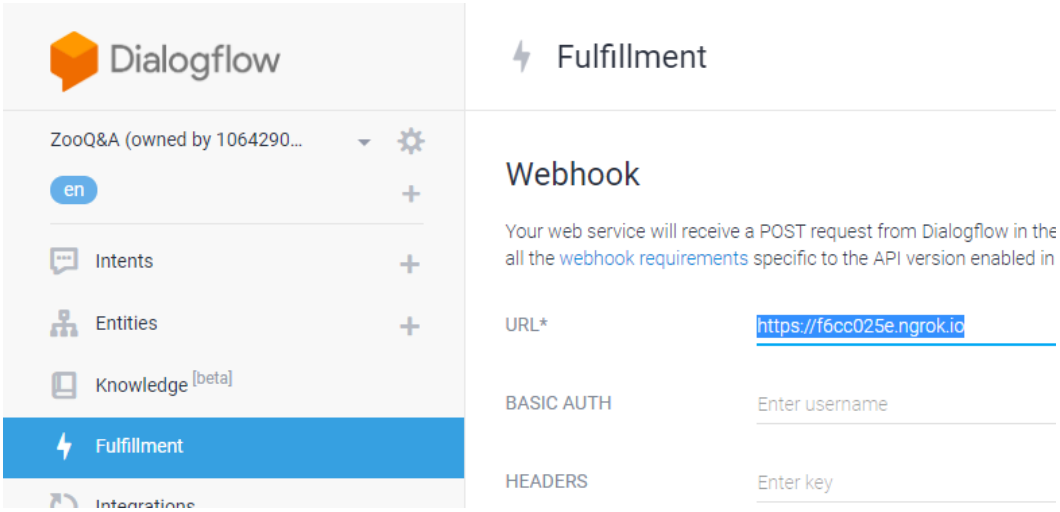
1. Test anaconda binary is detectable by your cmd/shell: open your cmd/shell and run this command: `conda –version`, ensure it shows the right version you have installed;
2. Start the server:
   a) Win10 users: double click the “setup.bat” batch file under `[project_root]/SystemCode/intelligentchatrobot` directory.
   b) Linux users: navigate your terminal to the `[project_root]/SystemCode/intelligentchatrobot` directory, and run: `bash setup.sh`

Start a ngrok session

1. Open another terminal/command line tool;
2. Run `ngrok http 4300` to generate a random url;
3. Copy the highlighted url to your clipboard;



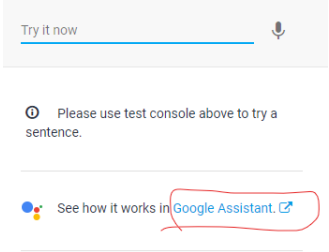
4. Open the Dialogfow console in your browser, paste the above url to “Fulfillment -> Webhook -> URL” column, REMEMBER TO SAVE before proceed.



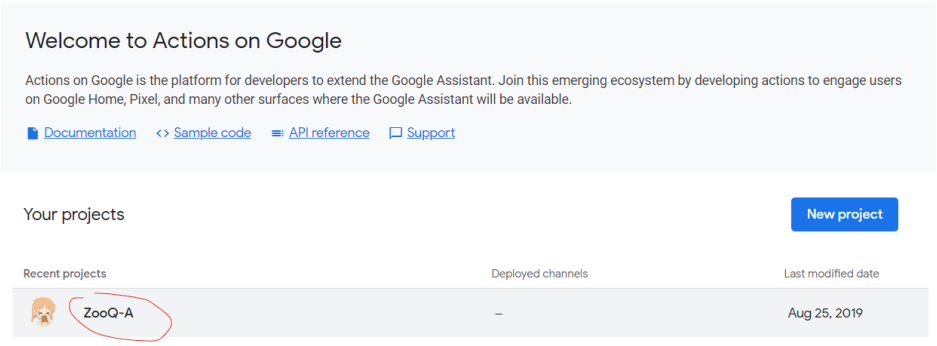
5. Notice that every ngrok session is only valid for 8 hours if you downloaded the free version, beyond that, according to our observation, ngrok service can be unstable from time to time, so if anything works abnormally, consider repeating the step 2~4.

Deploy Actions on google

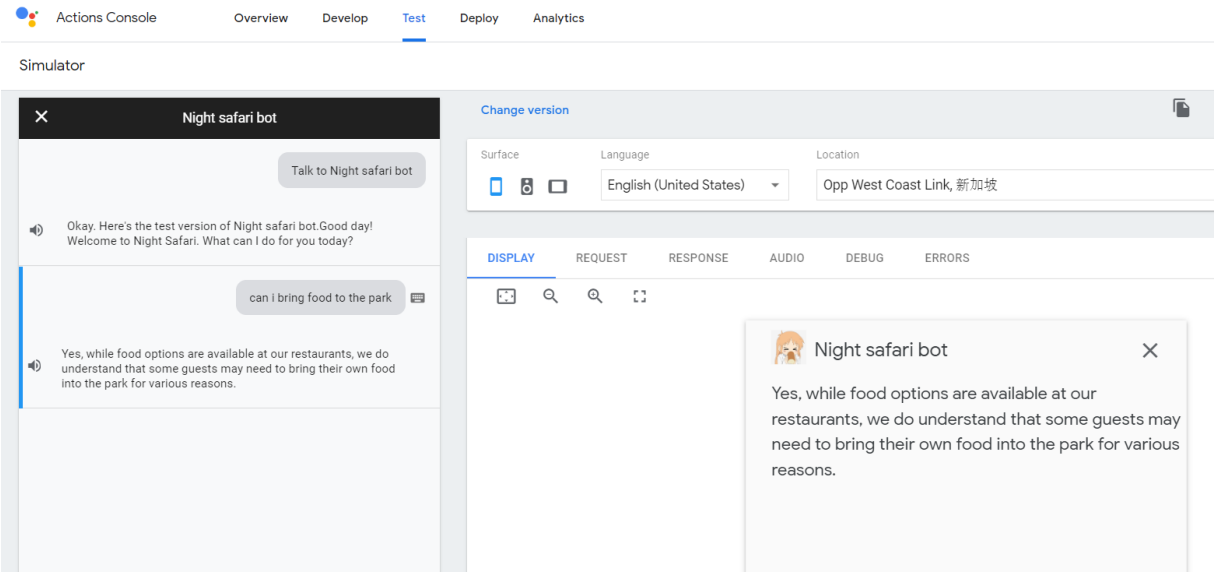
1. Redirect to Actions on google through right side panel’s “google assistant” link:



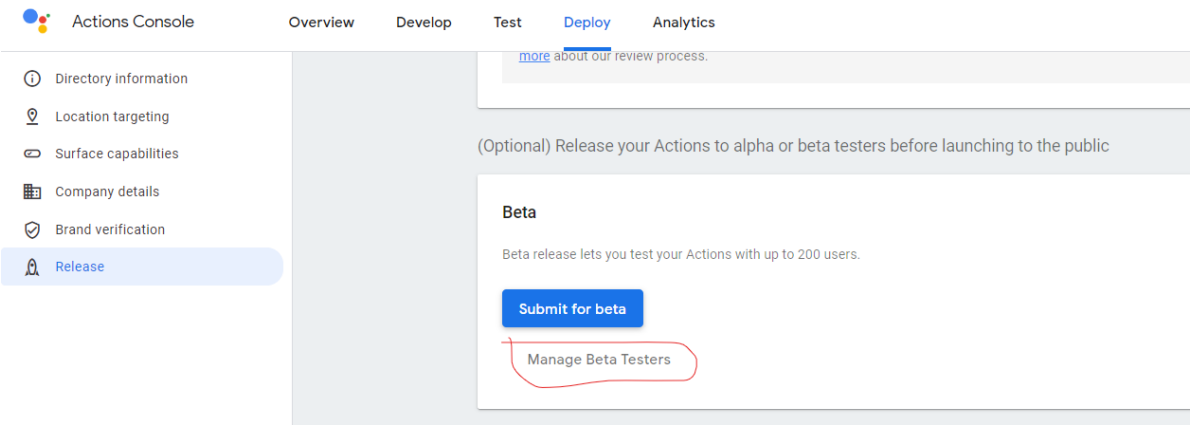
2. Select the dialogflow project you just created:

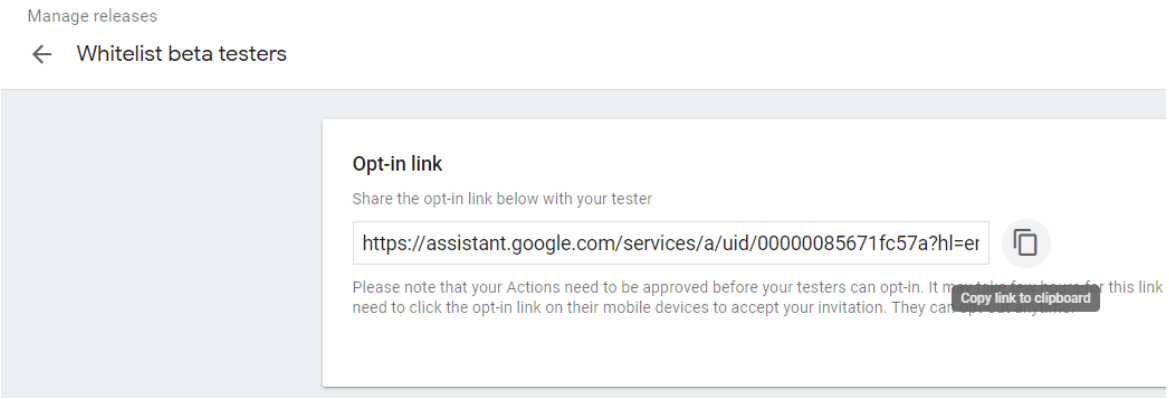


3. Follow the steps on https://developers.google.com/actions/console/publishing, fill in some information in order to release your project to Alpha/Beta Environments. Half-ways, you could test out your actions in simulator in "Test" panel:

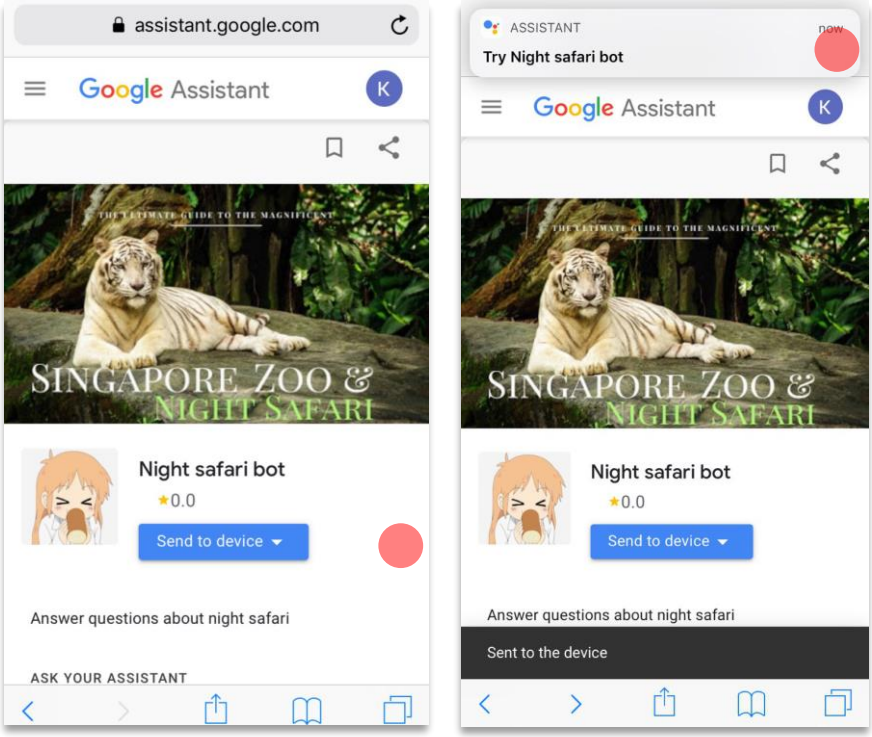


4. Finally, submit a beta/alpha release under Deploy -> Release -> Beta/alpha -> manage Beta/alpha testers. Then copy the opt-in link:

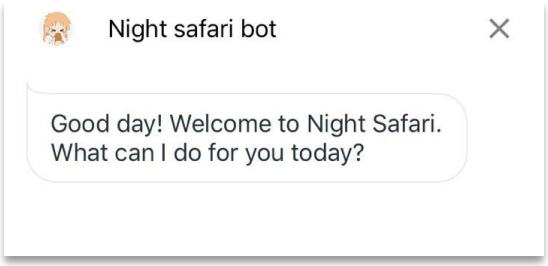




5. Onen the link in your mobile browser, log in to your google account.
6. Click “send to device”, select your current device, then redirect to google assistant app.



7. By now, you should be already inside the night safari bot, with the avatar and chatbot name you previously set in step 3. If not, check if you have signed in as the same account in dialogflow and invocate the google assistant with your invocation phrase you set in step 3, e.g. “talk to night safari bot”.



8. Enjoy your conversation on google assistant :)

Business Scenarios

Although our intent and entities configuration combine all kinds of scenarios into one intent, for clarity, here we categorize the cases.

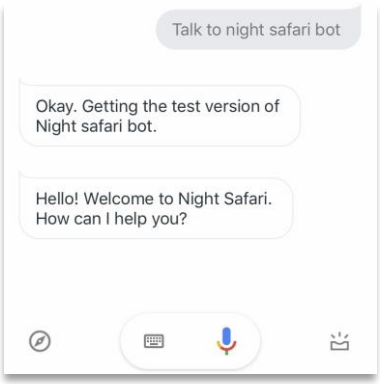
Queries about listed information on Night Safari webpages:

Interrogative type	Zoo(general)	Program	Event	Activity	Animal	Zone	Show	Dine/shop
What	Descrption	Similar to "zoo" column						
When	Open/close time							
How much	Admission rate of price group							
Where	Location of the zoo							
Direction	Google map direction	Not available						

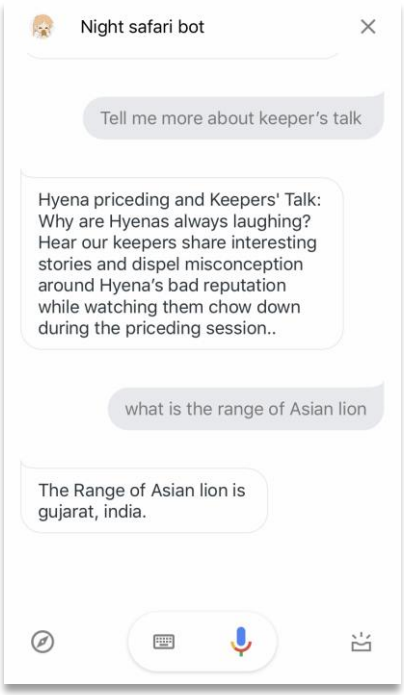
Interaction demonstration (on Google Assistant app)

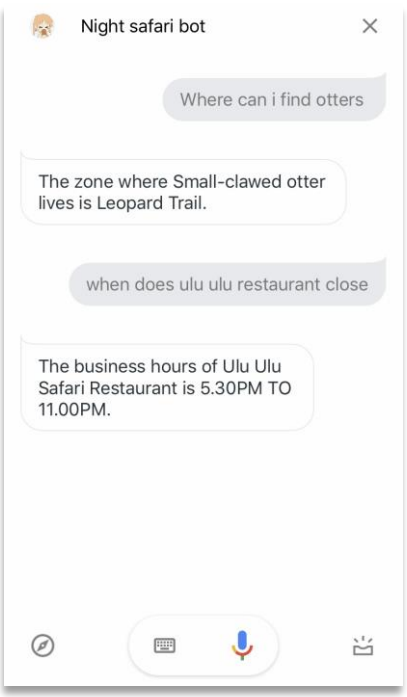
Facilitated by google action api, our users can not only read from our text responses, but also interact with some of them, enhancing user experience.

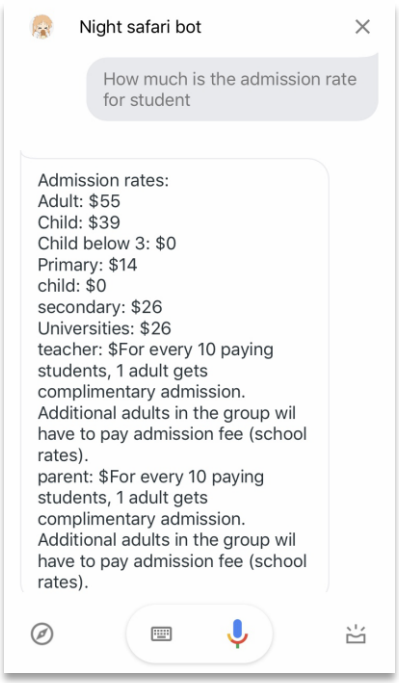
Trigger the google assistant app



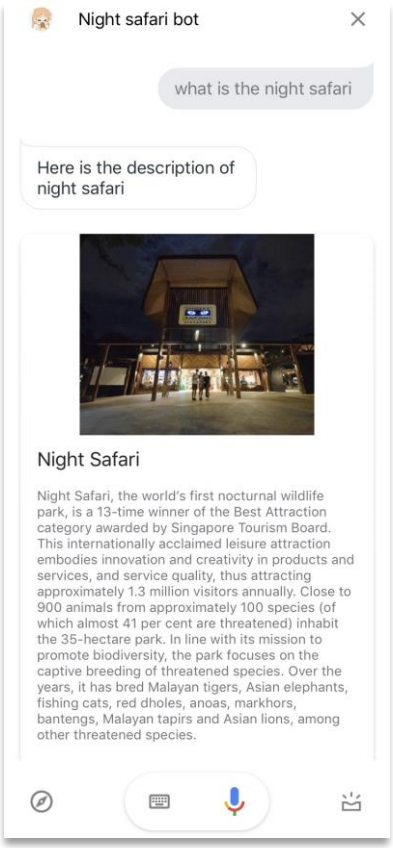
Specific questions about event/animal, etc.

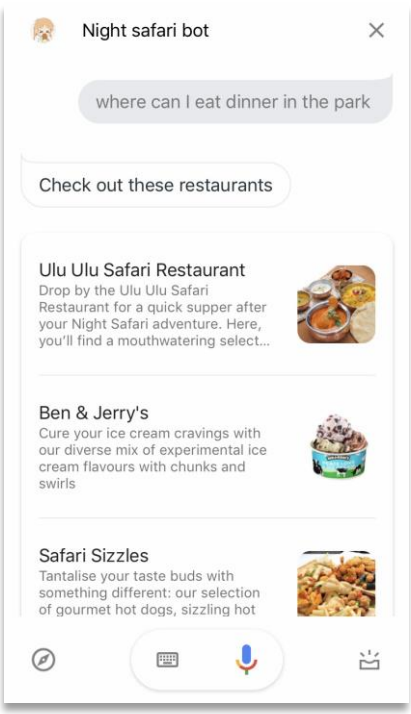


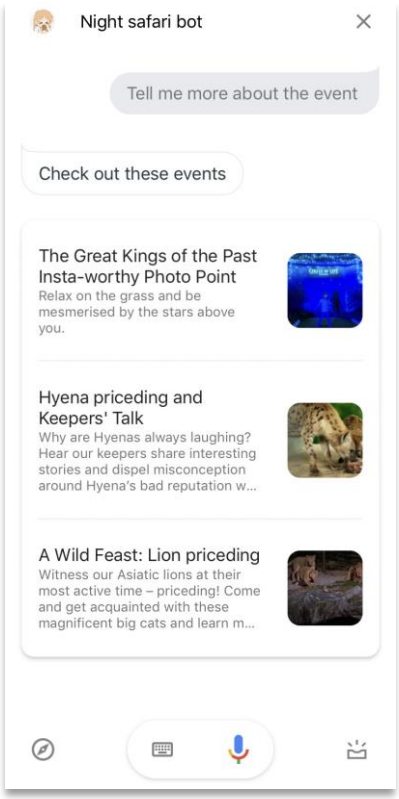




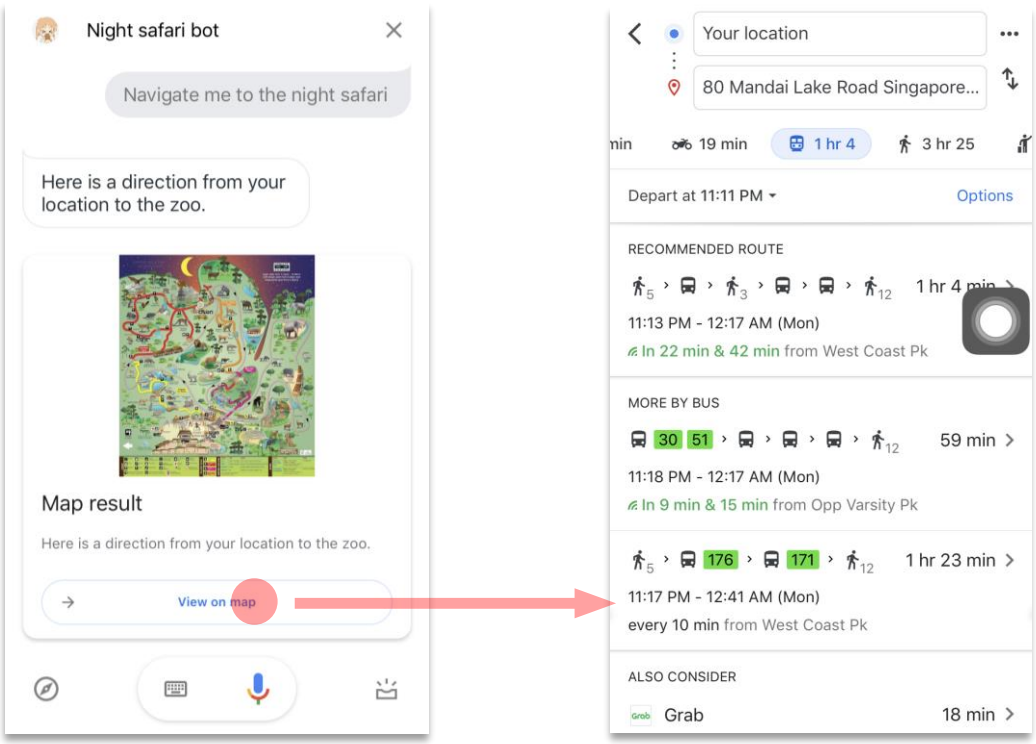
General questions







Navigation



*If you wish to switch your frontend to web demo or other integrations that don't support card display of google actions, you don't have to modify any configuration or restart at all. Backend script can detect your request environment and decide the correct display format at runtime.