Order Buddy

Aanchal Anil Samdariya: samdariya.a@husky.neu.edu
Apoorva Kasoju: kasoju.a@husky.neu.edu
Karan Gulur Muralidhar: gulurmuralidhar.k@husky.neu.edu
Krishna Sai Rohith Kanuparti: kanuparti.k@husky.neu.edu

November 4, 2019

I. Summary

Traditional way of placing an order involves repetitive manual labour, be it giving/taking orders face to face at an eatery or using mobile/web applications for ordering. Leveraging increase in popularity of hands free assistant[1] such as Google Home and Alexa, we enable conversational agent to work as a Speech-to-text Order Assistant.
The user will be given the ability to navigate through all phases of order management[3](i.e., placing an order, modifying the existing order and cancel an item/order), in effect, have a conversation and be able to place an order hands free.
The following could be an example of the kind of conversation possible with Order Buddy:

OB: Hey there! What do you want to order?
Cust: I want a Mocha Latte.
OB: What size?
Cust: Large
OB: Got it, one large Mocha Latte. Do you want to order anything else?
Cust: One small Coffee please.
OB: Got it, one small Coffee. Do you want to order anything else?
Cust: No, that's it.
OB: Order for 1 large Mocha Latte and 1 small Coffee is Confirmed, have a good day!

The history of all orders will be stored for future reference. A subset of the menu from Starbucks will be used as the dataset for proof of concept and later aim to expand to any menu.

II. Methods

To better explain the different aspects of this project the following are the broad sections to which we sub divide methods section as follows

i. Dialogflow Components

Dialogflow[4] is a Google API which supports speech to text transition and has the following components used for the project:

- **Agent** A natural language understanding module that handles conversations with end-users. It serves as top-level container for data and settings, including intents, entities, integration and fulfillment. Dialogflow provides pre-built agent which is specific to use-cases such as booking reservations.
- **Entity** Intent parameter with type and possible values to extract data from an end-user's expression, including system entities that match date, time, color.
- **Intent** Intent categorizes an end-user's intention. Intents consists of training phase, actions, parameters, response, contexts and events. When a user's utterance is matched to one of training examples, corresponding intent is matched along with extracted parameters and triggers associated action/text response.

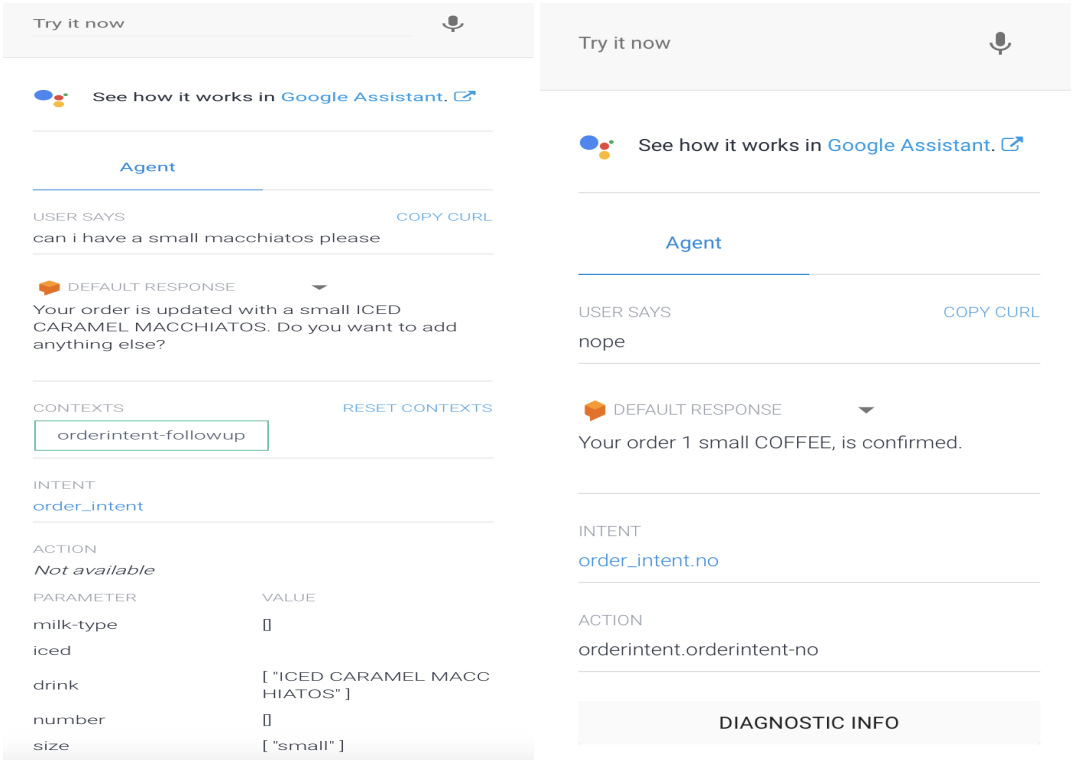

Figure 1: *On left:Order Intent; On right:Order Intent Followup - No*

  - **Follow-up Intent** A follow-up intent is a child of its associated parent intent and contexts get automatically configured for this pair. When a follow-up intent is created, its input context and output context for parent intent is set with same name. A follow-up intent is only matched when the parent intent is matched in the previous conversational turn.

- **Context** Context helps to understand user's utterance by controlling the order of intent matching based on input-output context mapping.
  - **Input Context** Input context helps to match intent only if the end-user expression is a close match and if the context is active.
  - **Output Context** Output context helps to activate a context if it's not already active or to maintain the context after the intent is matched.

- **Fulfillment** Allows real-time, dynamic responses and triggers action for user's utterances by connecting your service to the agent for integration. To achieve this, intent should have fulfillment enabled else intent's default response is returned.
- **Google Cloud Firestore** Cloud Firestore[5] is a cloud-hosted, No-SQL database that allows for mobile, web, and server development. Provides flexible and scalable storage in the form of collection-document structure which can contain different data types, sub-collections etc.

ii. Intent Definitions

- **Order Intent:** This intent handles the ordering intent as seen in **Figure 1 (*left*)**. Adds an item to the current outstanding order of the user.
  - **Sample Utterances:** Can I have a large coffee?; please add a flat white

  - **Input Context:** None expected.
  - **Output Context:** *order-intent-followup*
  - **Context lifetime:** 1 conversation
  - **Response:** Done! do you need anything else?
  - **Fulfillment:** Add the item to Firestore collection

- **Order Intent followup - Yes:** This is a follow up intent which follows after the order intent, if the user replies *Yes* to the question *Do you want to add anything else to the order* then the Dialogflow agent resolves to this intent.
  - **Sample Utterances:** yes ; yes, a large coffee;
  - **Input Context:** *order-intent-followup*
  - **Output Context:** None
  - **Context lifetime:** NA
  - **Response:** what do you want to add?
  - **Fulfillment:** None

- **Order Intent followup - No:** As seen in **Fig. 1 (*right*)**,this is a follow up intent which follows after the order intent, if the user replies *No* to the question *Do you want to add anything else to the order* then the Dialogflow agent resolves to this intent.
  - **Sample Utterances:** No; No I do not want to add anything;
  - **Input Context:** *order-intent-followup*
  - **Output Context:** None
  - **Context lifetime:** NA
  - **Response:** Done your *ORDER* is confirmed
  - **Fulfillment:** Confirm and Delete the order from *current order* collections and added to *history* collection.

- **Complete Order Intent:** This intent is triggered when the user voices a utterance that conveys an intent to place the order as shown in **Fig. 3 (*left*)** and its follow-up in **Figure 3 (*right*)**.
  - **Sample Utterances:** confirm order; complete order;
  - **Input Context:** None
  - **Output Context:** *complete-order-followup*
  - **Context lifetime:** 1 conversation
  - **Response:** Are you sure?
  - **Fulfillment:** None.

- **Cancel Order Intent:** This intent is triggered when the user voices a utterance that conveys an intent to cancel the order as illustrated in **Fig. 2 (*left*)**.
  - **Sample Utterances:** abort; exit; bye
  - **Input Context:** None
  - **Output Context:** *cancel-order-followup*
  - **Context lifetime:** 1 conversation
  - **Response:** Are you sure?
  - **Fulfillment:** None.

- **Cancel Order Intent Followup - Yes:** This is a follow up intent which follows after the cancel order intent. If the user replies *Yes* to the question *Are you sure you want to cancel the order?*, the Dialogflow agent resolves to this intent.
  - **Sample Utterances:** Yes; yup;
  - **Input Context:** *cancel-order-followup*
  - **Output Context:** None.
  - **Context lifetime:** NA
  - **Response:** Done your order has been cancelled.
  - **Fulfillment:** Order is deleted from Firestore.

- **Cancel Order Intent Followup - No:** This is a follow up intent which follows after the cancel order intent. If the user replies *No* to the question *Are you sure you want to cancel the order?*, the Dialogflow agent resolves to this intent.
  - **Sample Utterances:** No; Do Not;
  - **Input Context:** *cancel-order-followup*
  - **Output Context:** None.
  - **Context lifetime:** NA
  - **Response:** Do you want to add anything else?
  - **Fulfillment:** None

- **Cancel Item Intent:** When issuing a cancel item, user through his utterance can choose to issue the cancel intent in the following ways:
  - **With no drink parameter in the utterance** *Ex: Cancel; remove*

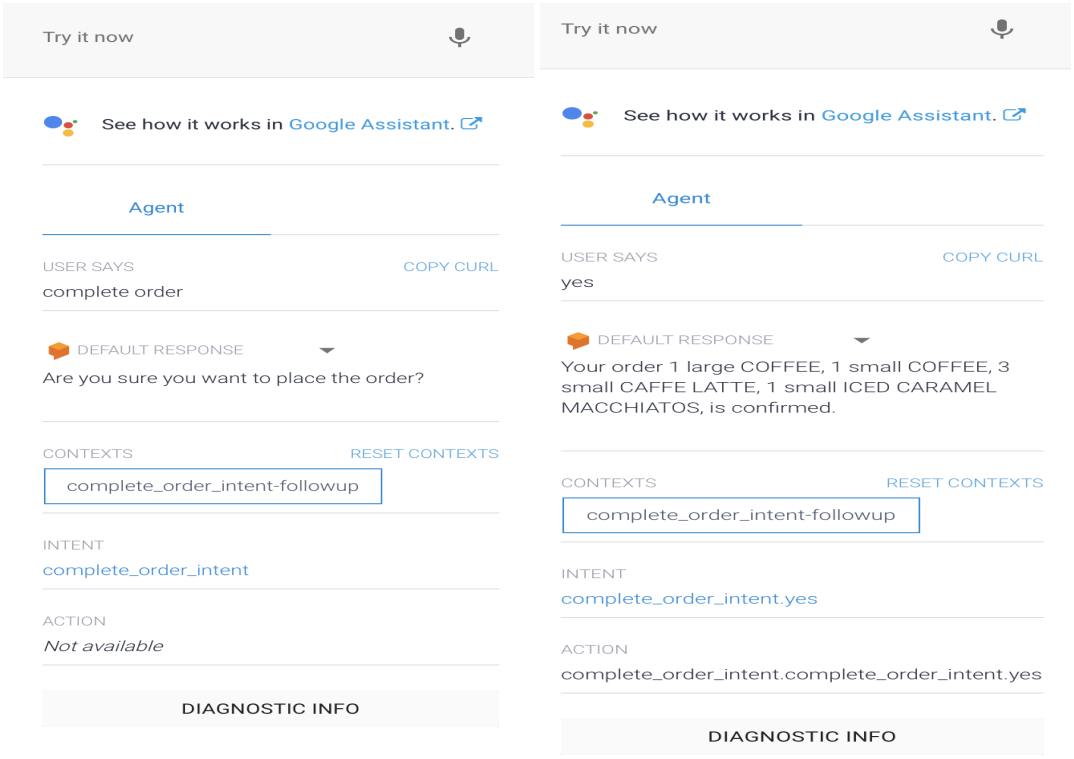

Figure 2: *On left:Complete Order Intent; On right:Complete Order Intent Followup - Yes*

  - **Input Context:** None
  - **Output Context:** *Cancel-item-followup.*
  - **Context lifetime:** 1 conversation
  - **Response:** *Say 1 to delete your last item*
  - **Fulfillment:** Retrieve the drinks to be used for deletion.

- **With a invalid drink parameter**, i.e the drink entity is recognized but has not been ordered by the user yet. *Ex: Cancel flat white; remove coffee*
  - **Input Context:** None
  - **Output Context:** *Cancel-item-followup.*
  - **Context lifetime:** 1 conversation
  - **Response:** *This drink is not in your order please say cancel followed a drink in your order*
  - **Fulfillment:** Retrieve the drinks present in the order to be used for response.

- **With a valid drink parameter** *Ex: Cancel flat white; remove coffee*
  - **Input Context:** None
  - **Output Context:** *Cancel-item-followup.*
  - **Context lifetime:** 1 conversation
  - **Response:** Lists all the drinks in the given category to be deleted by the user.
  - **Fulfillment:** Retrieve the drinks present in the order to be used for response.

- **Cancel Item Intent Followup - Continue:** As seen in **Figure 2 (*right*)** ,this is a follow up intent which follows after the cancel order intent. If the user replies *Item-number* to the question *Please say 1 for large coffee, 3 for small flat white*, the Dialogflow agent resolves to this intent.
  - **Sample Utterances:** 13; item number 3;

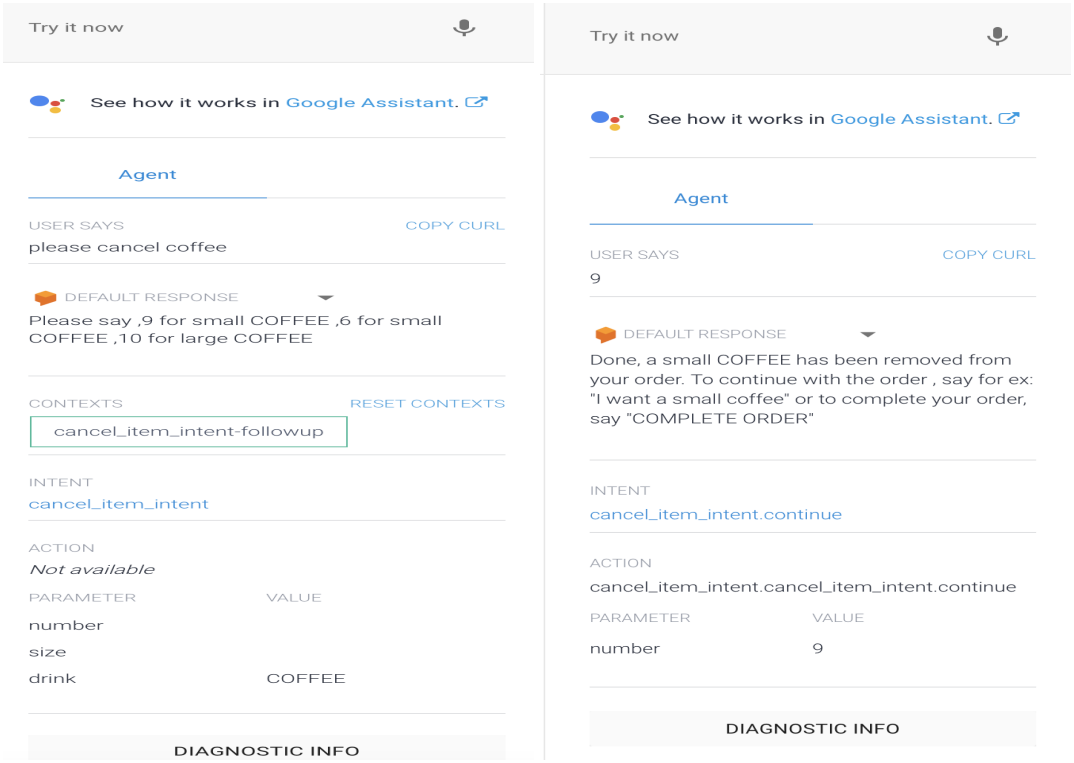

Figure 3: *On left:Cancel Item Intent; On right:Cancel Item Continue Intent*

  - **Input Context:** *cancel-item-followup*
  - **Output Context:** None.
  - **Context lifetime:** NA
  - **Response:** If the item is present it is deleted or we repeat the drinks list to the user
  - **Fulfillment:** Item is deleted if present in the order or user is prompted to choose from the list of drinks again.

– **Cancel Item Intent Followup - No:** This is a follow up intent which follows after the cancel order intent. If the user replies *No* to the question *You sure you want to delete an item? Please say 1 for large coffee, 3 for small flat white*, the Dialogflow agent resolves to this intent.

  - **Sample Utterances:** No;
  - **Input Context:** *cancel-item-followup*
  - **Output Context:** None.
  - **Context lifetime:** NA
  - **Response:** We repeat the drinks list to the user.
  - **Fulfillment:**Retrieve the drinks ordered by the user to choose.

III. Results

A sample conversation in **Fig. 4** shows the transition of intents based on user's utterances and context mapping, along with responses. As seen in the figure mentioned, a typical user wishes to have a small coffee in first figure of the flow. Upon receiving user's voice input, it is converted to text. The intent is identified and agent looks for required entities to fulfill the order and hence prompts the user to specify, i.e., size(second figure in the flow). Once the slot-filling(required entities) is accomplished, the Firestore database is updated with the current order information for the user as part of

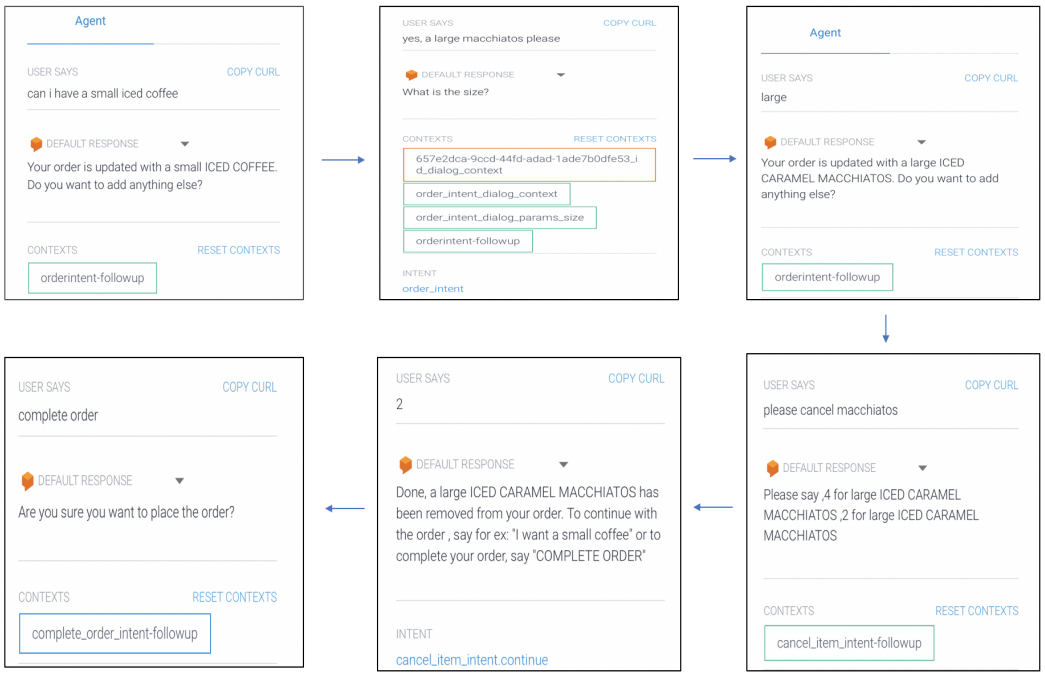

Figure 4: *A Sample Conversation*

the business logic and the user is notified about the same (third figure in the flow), including in response if he wants to add anything more. Here, the user expresses his intent to cancel one of items(fourth figure in the flow), for which the follow-up intent is triggered and the agent responds by providing options for the user to choose for cancel. After the user provides the option, the agent confirms the cancellation and prompts user to choose to continue or complete the order(fifth figure in the flow). If the user wishes to complete the order, the agent prompts to confirm to place the order and a positive response results in placing/completing the order(sixth figure).

IV. Discussion

For the next phase, we plan to work on following tasks:

- **More training and Intents** We plan to train by collect user's utterances from surveys along with other data-sets to make our model more robust. We also want to enable customization to orders, that provides ability to the user to be able to change a part/complete order anytime during the order life-cycle. They can customize already ordered items as well according to their preferences.
- **Multiple Items and Restaurants** In addition to including the menus from different restaurants and proper entities, we would be implementing a provision of accepting user's utterance with list of items required instead of just a single item at a time.
- **Multi-user Support and Integration** Extend current implementation to enable multiple users to interact with our OrderBuddy via different popular platforms such as Google Assistant, Facebook Messenger and Mobile.

V. Statement of Contributions

- **Aanchal Anil Samadariya** Design of Intents, Fulfillment and Dialogflow.
- **Apoorva Kasoju** Design of Intents,

Google Cloud Firestore and Initial Research.

- **Karan Gulur Mularidhar** Design of Intents, Dialogflow and Initial Research.
- **Rohith KrishnaSai Kanuparthi** Design of Intents, Google Cloud Firestore and Fulfillment.

References

[1] Google Cloud Speech to Text: *https://cloud.google.com/speech-to-text/*.

[2] Google Cloud Natural Language: *https://cloud.google.com/natural-language/*.

[3] Homa B. Hashemi, and Amir Asiaee, Reiner Kraft
Query Intent Detection using Convolutional Neural Networks Intelligent Systems Program,University of Pittsburg and Yahoo Inc.,Sunnyvale,CA

[4] Google Cloud Dialogflow: *https://cloud.google.com/Dialogflow/docs/reference/rest/v2-overview*.

[5] Google Cloud Firestore: *https://firebase.google.com/docs/Firestore/quickstart*.

[6] Brenes, David Gayo-Avello, Daniel and Pérez-González, Kilian
Survey and evaluation of query intent detection methods Proceedings of Workshop on Web Search Click Data, WSCD'09. 10.1145/1507509.1507510

VI. Appendix

Sample Code for Order Intent

```
def order_intent(self):
    user_id = self.request.userid
    parameters = self.request.parameters
    Firestore_timestamp = Firestore.SERVER_TIMESTAMP
    # Assumption – Only one item per request.
    drink_name = parameters.get('drink')[0]
    drink_size = parameters.get('size')[0]
    # INSERT TO DB (If collection not present, it get's created)
    document_exists = self.Firestore_client.collection(u'current_order').document(user_
    if document_exists:
        doc_ref = self.Firestore_client.collection(u'current_order').document(user_id)
        doc_ref_dict = doc_ref.get().to_dict()
        current_item_count = doc_ref_dict.get(u'current_item_count')
        item_number = current_item_count + 1
        drinks_dict = doc_ref.get().to_dict().get(u'drinks')
        if drinks_dict is None:
            drinks_dict = {}
        if drink_name in drinks_dict:
            drinks_dict.get(drink_name)[str(item_number)] = {u'size': drink_size}
        else:
            drinks_dict[drink_name] = {}
            drinks_dict[drink_name][str(item_number)] = {u'size': drink_size}
        doc_ref.update({
            u'current_item_count': item_number,
            u'order_timestamp':Firestore_timestamp,
            u'drinks': drinks_dict})
    else:
        current_item_count = 0
        item_number = current_item_count + 1
        doc_ref = self.Firestore_client.collection(u'current_order').document(user_id)
        doc_ref.set({
            u'current_item_count': item_number,
            u'order_timestamp':Firestore_timestamp,
            u'drinks': {
                drink_name: {str(item_number): {u'size': drink_size}}}
        })
    response = {'fulfillmentText': 'Your order is updated with a ' + drink_size + ' ' +
    return response
```