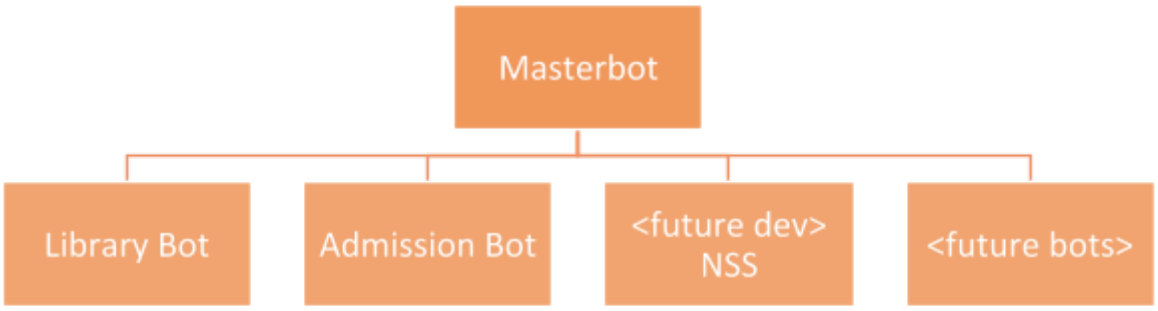
Technical Guideline for implementing Multibot Architecture using Dialogflow and Kommunicate – slau356v

Version 1.0, dated 27 Dec 2019

Written by Viet

This guideline **suggests** a framework for maintaining and expanding the current multibot system. At the moment, the structure follows the tree topology:



It is a complex system implemented to overcome the existing issue of 2000 intents quota from Dialogflow. By introducing separate bots for different groups of content, it is also easier for respective content owners to govern and maintain their own data without getting affected by other changes.

The content of this guideline includes:

1. Conversational design for multibot system
2. Details of the current system implementation - Dialogflow
3. Technical setup step-by-step guide for creating a new child bot
4. Kommunicate Setup

This is created purely as a guideline, hence there is no need to adhere to it strictly. Nonetheless, it is still a very good idea to follow as closely as possible to the suggested framework to maintain the entire system and avoid potential issues as we enter the system of multi bots, each of which is maintained by different content owners.

I. Conversational design for multibot system

The current multibot architecture defines these distinctive features to allow meaningful interaction among the bots:

1. Child bot has NTU domain-specific intents that no other bot should have.
2. During a conversation, should the user ask a non-domain specific question, the child bot will redirect the user to the masterbot.
3. The masterbot has a set of routing intents that can meaningfully direct the user to the correct place
4. Child bots cannot communicate with each other directly. It must be redirected to masterbot (thus the tree topology).

Point 1 is totally up to the content side of things.

Point 4 is for scalability and reducing new bot setup overhead.

Point 2 is elaborated below:

Below are the main templates of bot interactions that are being implemented in the system. Any changes to this underlying conversational design necessitates a change in the technical implementation.

Template 1: Fallback in Childbot due to Non Domain Specific Questions

During a conversation with childbot, the user asks something that is not the expertise of the bot

User: [says something childbot cannot answer because it is outside of childbot's domain] - *this is the queryText[1]

Child Bot: What you are asking may be beyond my knowledge. Would you like to repeat your query one more time? If you are asking non-[childbot] specific questions, I can refer you back to the Master bot! <button: redirect to master bot>

User: *click on button to redirect*

Master Bot: I am sorry that my colleague could not answer your queries. Which of the following categories do you think your issue might fall under?

<button: library><button: admission><*button for each of the other child bot*>

User: *click on button to another childbot*

Master Bot: *redirect (*silent – no message will be displayed)*

Child Bot 2: Looks like I have been called upon to answer your question. Would you mind repeating your query one more time?

This approach makes use of buttons so that the user only has to repeat their failed query once. Future improvements to this can be made by passing the queryText from one bot to another. This will be

[1] Following Dialogflow lingo

frequently seen if the user asks about multiple topics spanning knowledge of different bots in one single conversation.

Template 2: Fallback in Childbot due to Domain Specific Questions

During a conversation with childbot, the user asks something that is the expertise of the bot, yet bot cannot answer due to poor intent matching/typo/etc.

User: [says something childbot is supposed to know]

Child Bot: What you are asking may be beyond my knowledge. Would you like to repeat your query one more time? If you are asking non-[childbot] specific questions, I can refer you back to the Master bot! <button: redirect to master bot>

User: [retype queryText]

Child Bot: [still don't understand] I'm so sorry that I could not help you, would you like to contact one of our staff?

<button for tel no. of content owner>

This template follows the original double fallback mechanism of Lyon.

Template 3: Fallback in Childbot due to Shared Questions

During a conversation with childbot, the user asks something that is the expertise of the bot, yet bot cannot answer due to poor intent matching/typo/etc.

User: [says something childbot is supposed to know]

Child Bot: What you are asking may be beyond my knowledge. Would you like to repeat your query one more time? If you are asking non-[childbot] specific questions, I can refer you back to the Master bot! <button: redirect to master bot>

User: [retype queryText]

Child Bot: I'm so sorry that I could not help you, would you like to contact one of our staff?

<button for tel no. of content owner>

Master Bot: I am sorry that my colleague could not answer your queries. Which of the following categories do you think your issue might fall under?

<button: library><button: admission><*button for each of the other child bot*>

User: [retype queryText]

Masterbot: *gives answer if it is a shared content among the bots.*

This has not been implemented yet due to content issues.

Point 3:

This is still a work in progress. My current implementation is to compile the most notable training phrases of the childbot to know which bot to reroute to, along with the use of buttons.

Now, there are several ways to expand this:

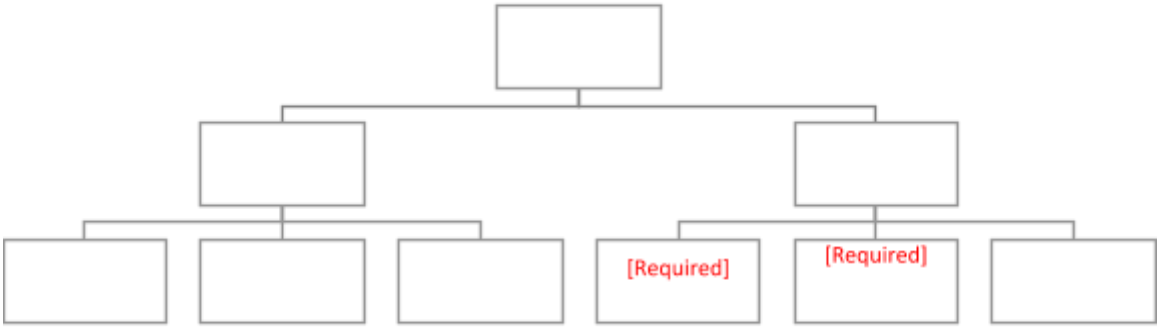
1. Use UAT to collect most common queries in childbot and put them in the ForwardTo[ChildBot] intent in Master Bot.
2. Put childbot Entities in Masterbot to reduce the number of training phrases in the ForwardTo[ChildBot] intents.

Regarding point #2, this is highly unreliable for two reasons: (1) many childbots might use the same entities but under different entities name. This will cause confusion for master bot upon imports, and (2) entities are very powerful and should not be used unless you are **very clear** about the distinction among different entities.

My suggestion is to rely more on training phrases, and use entities only for extremely specific keywords like Library, Admissions, etc. The worry I currently have is that it is very hard to debug entities problems once we use it (see Viet's Dialogflow Manual on this issue)

II. Technical implementation - Dialogflow

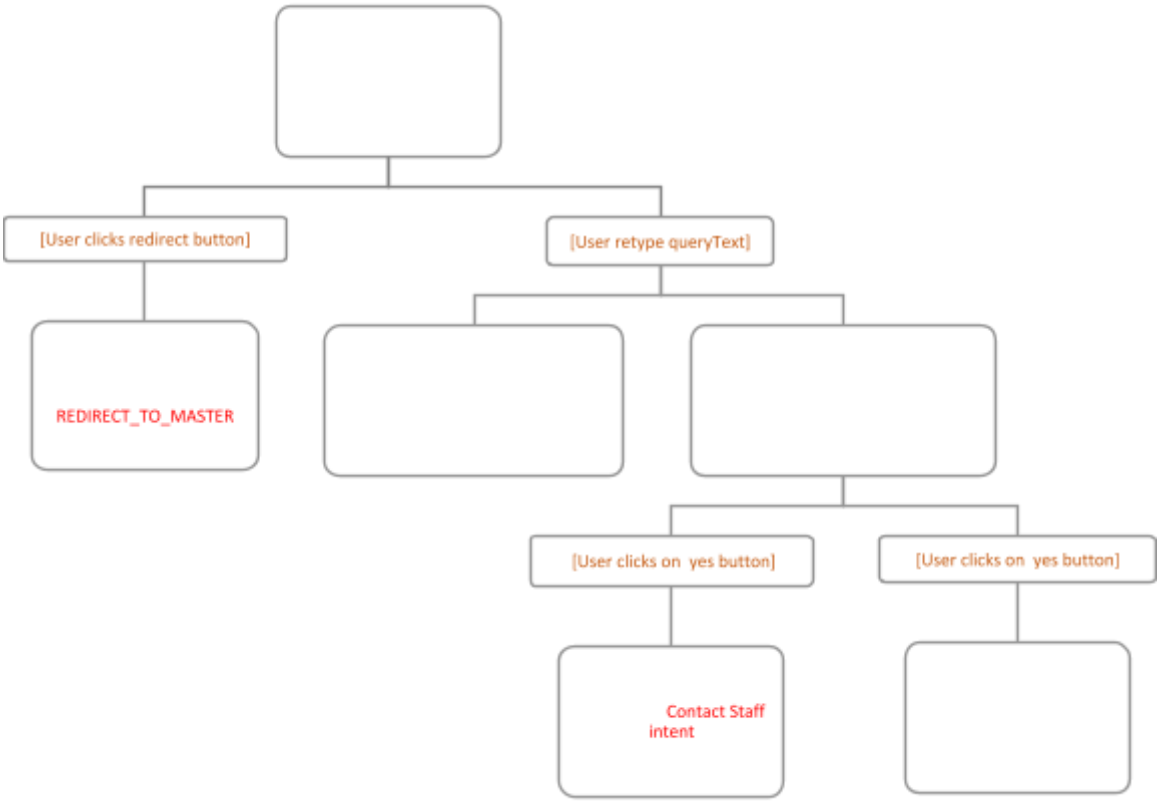
Childbot implementation hierarchy:



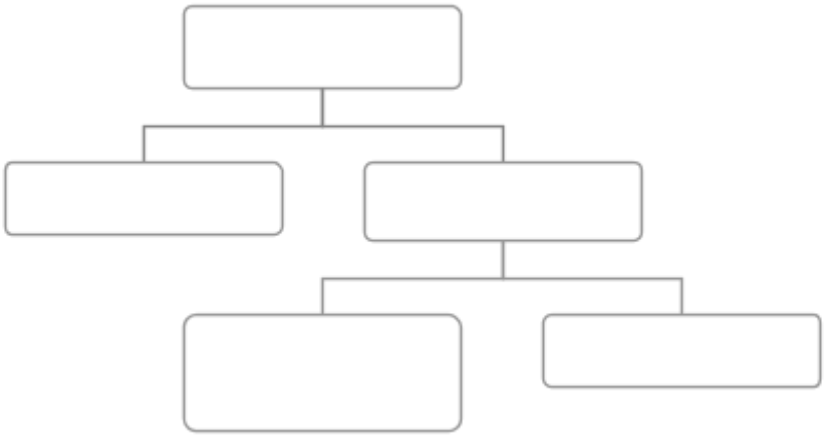
Detailed explanation of each intent and their purposes:

1. REDIRECT_TO_MASTER: this intent is used when the child bot needs to hand over the conversation to masterbot, usually when met with a non domain specific question
2. REDIRECT_FROM_MASTER: this intent is triggered when the child bot is called upon by the masterbot to handle a conversation that *another* bot cannot handle. I create this intent to make it more user-friendly: we could have used Welcome intent, but understanding that the user is facing an issue with another bot, we decided to give a more customized response to meet their needs.
3. Contact Staff: This intent is required for fallback (explained below). It should have a training phrase: "I would like to contact a staff"
4. Customized Welcome: just to provide a more personal touch to how each bot greets the user when they come to the website.
5. Fallback :

When the user says something the bot do not understand



Internally it is represented in dialogflow intents as such:



III. Step-by-step Dialogflow set up guide

For the childbot, please ensure that these 5 intents are inside and properly configured as specified below

Intent name	Event	Training Phrase	Response
REDIRECT_FROM_MASTER	REDIRECT_FROM_MASTER	No training phrases	It seems like I have been called to answer your query. Would you be so kind as to repeat your earlier query?
REDIRECT_TO_MASTER	No event	I would like to be redirected to master bot	KM payload to redirect to master bot
Contact Staff	No event	I would like to contact a staff	KM Payload for url with formal "tel: phone no"
Welcome	Default	Default	Specialised according to each bot persona
Defaul Fallback Intent	Default	Default	What you are asking may be beyond my knowledge. If you are asking a question not related to Admission, I can refer you back to the Master bot! Otherwise, if you are asking questions related to Admission, please repeat one more time. <KM payload for button>
Defaul Fallback Intent -fallback (Default fallback followup intent)	Default	Default	I still don't understand what you said. Would you like to contact one of our staff <yes button> <no button>
Defaul Fallback Intent -fallback-no (Default no followup intent)	Default	Default	I'm so sorry for not being able to help with your request ☹ Do you still have any other [BOT NAME] related questions?

KM Payload to redirect

```
{
 "platform": "kommunicate",
 "message": "Forwarding your request to a teammate who is expert in this",
 "metadata": {
  "KM_ASSIGN_TO": [BOT_ID to be redirected to],
  "event": {
   "name": [the event name to be triggered to the bot],
   "data": {}
```

```
   }
  }
}
```

KM payload for buttons:

```
{
  "message": [text to be displayed before buttons],
  "platform": "kommunicate",
  "metadata": {
    "contentType": "300",
    "templateId": "6",
    "payload": [
      {
        "title": [What is shown on the button],
        "message": [The message that is sent by clicking on the button]
      }
    ]
  }
}
```

For the Masterbot

It is already configured so there should not be any changes until a new bot is added.

Please be note that for maximum efficacy in routing, Master bot needs to be updated everytime ANY childbot has a major update to its content/ new bot is added

Intent name	Event	Training Phrase	Response
REDIRECT_TO_MASTER			
ForwardToAdmission	FORWARD2ADMISSION*	**	KM Payload to redirect (see previous section on childbot)
ForwardToLibrary	FORWARD2LIBRARY*	**	KM Payload to redirect (see previous section on childbot)
ForwardToXXX (for any future bots)	FORWARD2XXX	**	KM Payload to redirect (see previous section on childbot)

*For the Event, although it is not compulsory, it is still a good move to enable this in case we have to resort to use the second method for Kommunicate (see below)

** This is still in progress and is dependent on the content of the childbot. Pick out the most frequently asked queries in each bot. The quota for number of training phrases is 2000. Avoid the use of entities as much as possible, especially the built in entities.

IV. Kommunicate Setup

Things should be more or less straightforward on this side of things

1. The easy way:

Embed this code (the one that allows different bot to be on different websites) on each website

<See attached file DiffWebsiteDiffBot.html>

You can configure it to only open on click as well – see Kommunicate's docs

Make sure the variables botID and appID is configured correctly.

2. The hard way (in case of emergency, use as backup only)

The procedure is below:

- Embed every website with code configured to masterbot botID
- Create different event that redirect from masterbot to each bot (e.g give the intent ForwardToLibrary and Event named ForwardToLibrary – as suggested in the previous section)
- Inside the embedded js of each website, include code that only open when the user click on the icon
- <see RevisedEventTriggerToDiffBot.html>
- Change the value event name on line 159 to FORWARD2[Botname]
- Ensure appID is correct and Conversational Rules in the Kommunicate dashbot is enabled to allow that bot to handle all conversations.

What this does is that on every website, master bot is embedded but when the user clicks open the chat widget, masterbot immediately redirect to the respective bot.