Strategy Paper on Voice & Chatbots

Contents Index

1. Conversational AI :

Conversational AI in form of virtual assistants, chatbots and voicebots have gained popularity nowadays, as it is used as a means of augmenting the system to automate the task of answering user queries that are repetitive in nature and provides administration support to speed up task completion by relieving them of such task which can be answered by the system and in the process bring efficiency for the citizens.

The process of creating a chatbot follows a pattern similar to the development of a web page or a mobile app. It can be divided into Design, Building, Analytics and Maintenance.

Designing the corpus for chatbot is the most important work. Automating Question Answers (QA) is very largely dependent on a good corpus - for without documents containing the answer, there is little any QA system can do. It thus makes sense that larger collection sizes generally lend well to better QA performance. Nuggets of information are likely to be phrased in many different ways in differing contexts and documents which will help the bot to train better due to redundancy in the data collected.

The process of Building a chatbot can be divided into two main tasks: understanding the user's intent and producing the correct answer.

- The first task involves understanding the user input. In order to properly understand a user input in a free text form, a Natural Language Understanding Component or NLP Engine can be used, that is trained on varieties of sentences as input and intents as target.
- The second task may involve different approaches depending on the type of the response that the chatbot will generate and involves the Dialogue management component and Middleware to handle business logic and User Interface for the clients.

2. Development Background:

NIC TDPP (renamed as IVRS, Service Desk Chatbot & Voicebot, later merged with AI Resource Division) was given the mandate to develop a voice assisted IVRS to log tickets wrt the Network and associated issues in August 2016. A speech enabled application with failover in DTMF was demonstrated but was not found acceptable. Subsequently, DG (NIC) gave the mandate to the division to work on Natural Language Understanding/Processing (NLU/P) based voicebot for the upcoming NIC Service Desk. Since there was no proper

training available, NLP based voice -bot could not materialise, even though extensive work and research was done.

Due to this, text based bots platform were surveyed and **Wit.AI** was found to be one such platform which was also available free. Work related activities progressed, while this platform was found to lack handling of small-talk or answering vague non related questions and certain features were being deprecated which is a major requirement of any bot. After evaluating around 25 other bot platforms, **API.AI** was chosen as it satisfied most of the requirements and was also available as a free product.

The first version was made available by mid-January 2018 for testing and evaluation in a limited period of time which was later brought into 24x7 operations by end of January 2018. Further improvements were carried out and a second version with better NLP understanding and accuracy was launched in mid-April 2018. As any AI application needs continuous training, this was also nurtured to give correct response, initially from a dismal 30% to near 100% accuracy in most cases.

Besides working on the chat based bots, the division also developed a generic framework for the whole ecosystem, and got it audited and deployed for all text/voice and mobile voice based bot usage. A PoC for voice based bot and one on a mobile were also developed and demonstrated to the Hon'ble. Minister for E&IT when he inaugurated the CoE-AI in early January 2019. The architecture of the framework is detailed later in this paper.

From the data collected in the text bot for nearly one and half years, a Statistical Language Model was generated after sufficient cleaning and scrubbing of the data and used for Automatic Speech Recognition (ASR) of the spoken problem statement by the user. This was further integrated to the generic framework and used the same back-end API.AI (Dialogflow) engine to find the problem classification just like the text bot. A similar case was used for developing the Mobile app based bot, through which a similar experience is achieved either by voice/text to raise a ticket.

3. Generic Framework of the Bot:

A generic framework has been developed in-house for the chat /voice/mobile bot in a layered- modular-pluggable architecture. The framework also has a Queue manager, monitor and a live human agent handover module. All these are connected to a front-end JQuery UI which can be plugged into the web interface running in any of the development frameworks like PHP, JSP, Servlet, Drupal, ASP.Net, etc.. on MS Windows or any Linux flavours.

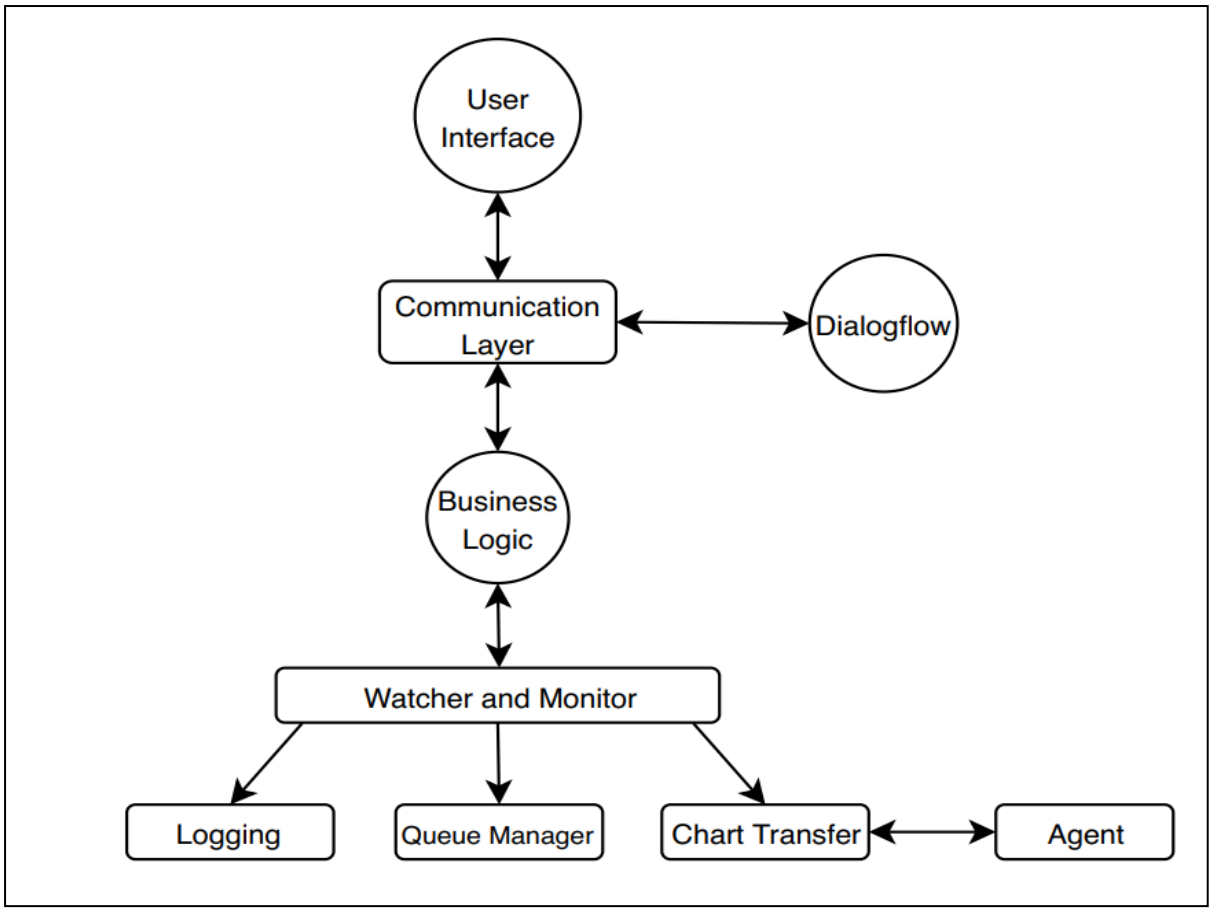

Figure 1 : VANI Conceptual Architecture

The Communication (CO) module does all the back and forth communication between the business module and the back-end AI engine using individual OAuth and session IDs for maintaining the privacy of each chat user.

The business logic layer is customised for individual chat/voicebot as the naming of Intents/action parameters differ from case to case. The business module hands over the conversation back and forth between the UI and the Communication (CO) module. On completion of the task assigned, the business module kicks in the associated layers, disconnecting from the CO module and may hand-over to the Queue manager or may do some other tasks like calling a user defined web service for disseminating some requested information or generate OTP, etc.

The Queue manager, as required, manages the queue for the transfer of chats to the human agents. The Agents, who are logged in are handed over the chats with the complete conversation history (the user and the bot conversation), which help the agent to smoothly continue the conversation from the point where it was handed over to the agent. The agents have individual applications installed on their respective desktops, where on getting a transfer, the screen is brought on top and notifies the agent with visual and audible warnings of chat transfers.

Using this generic framework, multiple Chat, Voice or mobile App based bots can interact with the back-end engine and also maintain their own individual sessions, keeping the privacy of each conversation and handling and meeting the business requirements, as shown below:

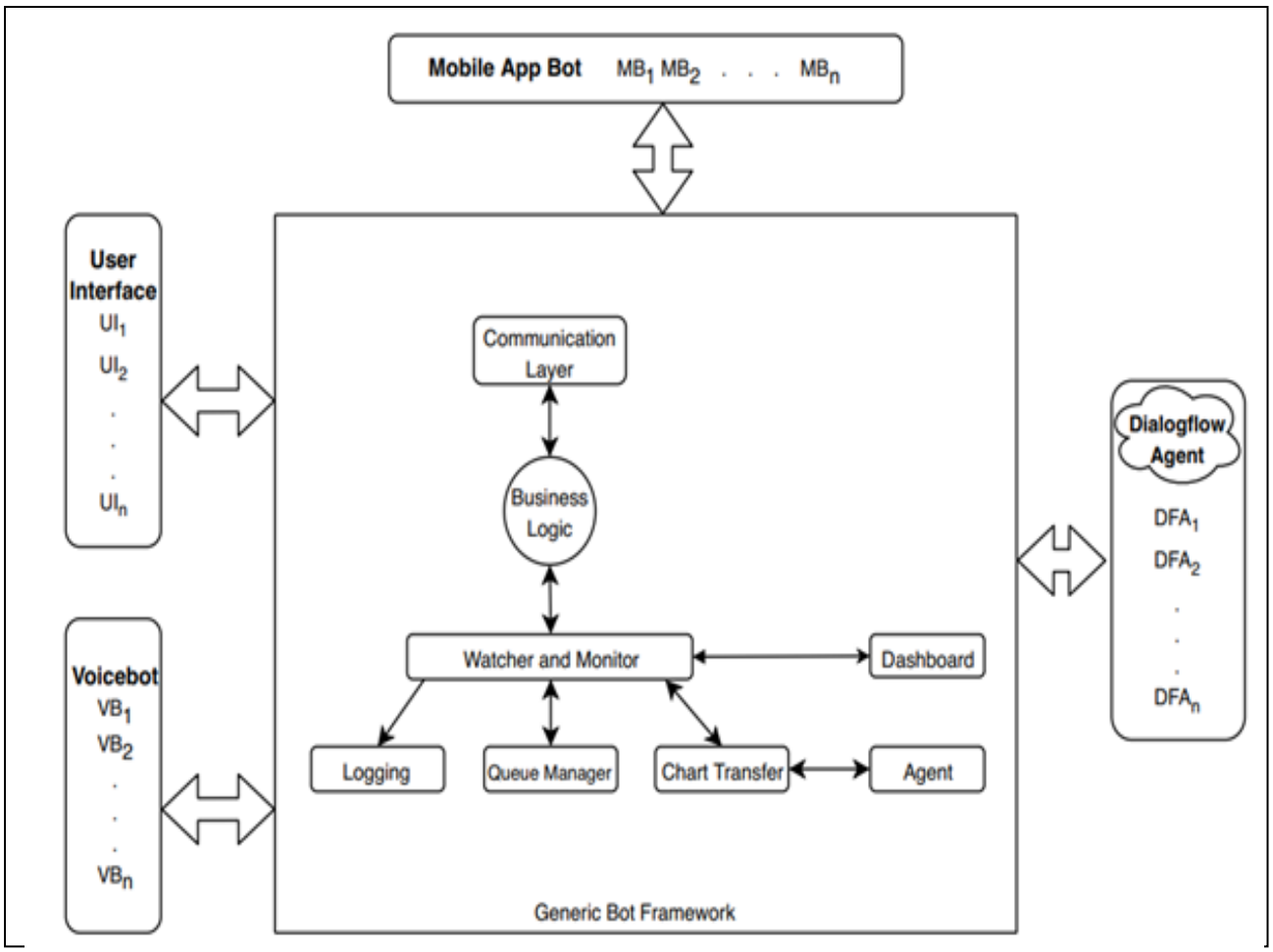

Figure 2 : VANI Deployment Architecture

4. Current Usage of the bot:

The chatbot for NIC Service Desk(NSD) has been enhanced with FAQ based L1 solutions to common problems for some of the domains where FAQ have been provided, like eOffice, VC, VPN, server side issues of Antivirus and scaled down version for eMail. The voice bot for NSD has been enhanced with better accuracy, approx. 95% or better, after loading all the domain data for the 2019 NSD, approx. 3.5 lakh records and the FAQ data.

Further cleaning of the NSD data is in progress for the years 2017-2018 for all the domains, the respective FAQs is being augmented with the problem statements available from NSD. With the available cleaned data from NSD, the SLM would be trained for even higher accuracy.

Chatbots have also been developed for the following Departments/Ministries other than the 15 domains covered for NIC Service Desk:

Sr. No	Department/ Ministry	Brief Functionality	Chat Bot Status	UAT Status
1	CONFONET, Consumer Affairs	Provides the case status for different state commissions /Benches and NCDRC	Primary requirements completed, new feature requests to be implemented in next release	Completed
2	Sarathi-Vaahan, Ministry of Transport	Provide information wrt licenses, fee & registration	Primary requirements completed	completed
3	eAwas – Chandigarh UT	Basic information wrt eAWAS	Primary requirements completed	Awaiting UAT
4	OTG Chennai	All info connected with OTG, like advisories on open source software, downloads, OS, etc. And ticketing of issues	AI part completed, integration with API and ticketing engine	To be done
5	DBT-PDS Puducherry	Info connected with PDS and DBT from FAQ as well as regular intents	Completed staging awaiting launch	UAT done
6	eVigilance – Chandigarh UT	General information, Application status and some analytical textual data	Under process	Not completed
7	Sahayak – HP Govt	Website Navigator with respect to the Govt of HP	Initialised, work started	

Figure 3 : Current BOT Development Status in VANI Framework

Current Infrastructure:

a) Chatbot - The framework, the associated DB and related services are currently working from the Meghraj Cloud, NDC, Shastri Park, Delhi. For scaling up the application framework, a load balanced setup needs to be setup in the Cloud, with failover/redundant servers at DR location/s.

The Chatbot currently uses the standard free version of Dialogflow APIs for classification of intents and usage of FAQs for providing L1 support, where applicable.

Till a suitable AI engine is developed, the Dialogflow APIs need to be used with the condition that no personal/ID related information is stored/trained on the engine. Further, if high volume chat transactions or additional features like Speech, both input/output need to be used, then enterprise version license needs to be procured. The enterprise version work on pay-as-you go payment model, with monthly payments, much like the post-paid telephone billing system.

b) Voicebot - The voicebot uses the **Nuance Voice Platform** for Automatically Recognising (ASR) of the spoken inputs by the caller using a Statistical Language Model (SLM) and associated W3C compliant VXML pages to deliver the answers using Text-To-Speech (TTS). NIC has procured the Nuance Voice Platform, hosted on premise, which doubles up as a normal IVRS with DTMF/Speech recognition inputs as well as a NLP/U based engine for natural speech inputs.

The perpetual licenses provide ASR for 14 Indian languages and TTS for Hindi & Indian Accent English. The current product is reaching End-Of-Life (EOL) on 30th September 2019. The newer version (Nuance Voice platform Speech Suite 11.04), using better NLP/U capabilities and providing capabilities for identifying/classifying the intents without any third party AI engine like Dialogflow, need to be upgraded from the existing one, which is available only after renewal of the maintenance and support.

****

5. Background for RASA(NLU) opensource framework :

Opensource RASA provides complete framework to design a context based chatbot, with conversation context being maintained in form of stories. Only if there is a change of context midway in user conversation, RASA may not be able to entertain the same.

RASA framework can be classified in three major parts:
1)**RASA NLU:** This part is used for identify intents and entity. NLU take two major inputs files :

a) Configuration file: It contains info regarding tokenization, model used for intent & entity identification, and model configuration like number of epochs etc.)
b) NLU model file: It contains intents, intent examples, entity corresponding to each intent, synonyms for entity, regex for entity identification.

2)**RASA Core:** This part is used for conversation. Core maintains the context and uses NLU module to find intents and entity in the user input. It take following inputs:

a) Story file: contains all stories which chatbot uses for conversation.
b) Domain file: it is like short summary of entire chatbot. It contains intent name, entity name, slots, responses and static response messages.
c) Configuration: related to the policies used for the learning the stories and fallback details.

Dialogue Model: This model is trained on stories we define, based on which the policy will take the action. There are two ways in which stories can be generated:

- **Supervised Learning:** In this type of learning we will create the stories by hand, writing them directly in a file. It is easy to write but in the case of complex use-cases, it is difficult to cover all scenarios.

- **Reinforcement Learning:** The user provides feedback on every decision taken by the policy. This is also known as interactive learning. This helps in including edge cases which are difficult to create by hand. You must be thinking about how it works? Every time when a policy chooses an action to take, it is asked from the user whether the chosen action is correct or not. If the action taken is wrong, you can correct the action on the fly and store the stories to train the model again.

3)**RASA SDK:** This part is used for performing user defined actions. RASA sdk runs the actions defined on separate server and execute it as event listener (triggered from the core side send request for respected action).

6. Challenges in current opensource RASA(NLU) framework :

1) Follow the concepts of story to follow the conversation. Disadvantages of the story method:
   a) Story is like a path followed for the conversation. So, it restricts the user to follow the restricted paths.
   b) Difficult for programmer to develop and manage all stories and paths.
   c) RASA is not able to switch between stories if we suddenly start another story (even other story is from the start).

2) Intent and Entity are not interdependent in rasa NLU model. This leads to cases when it identifies correct entities but wrong intents.

Implementation Architecture :

Image shown below is the high level architecture (workflow) of Core part:

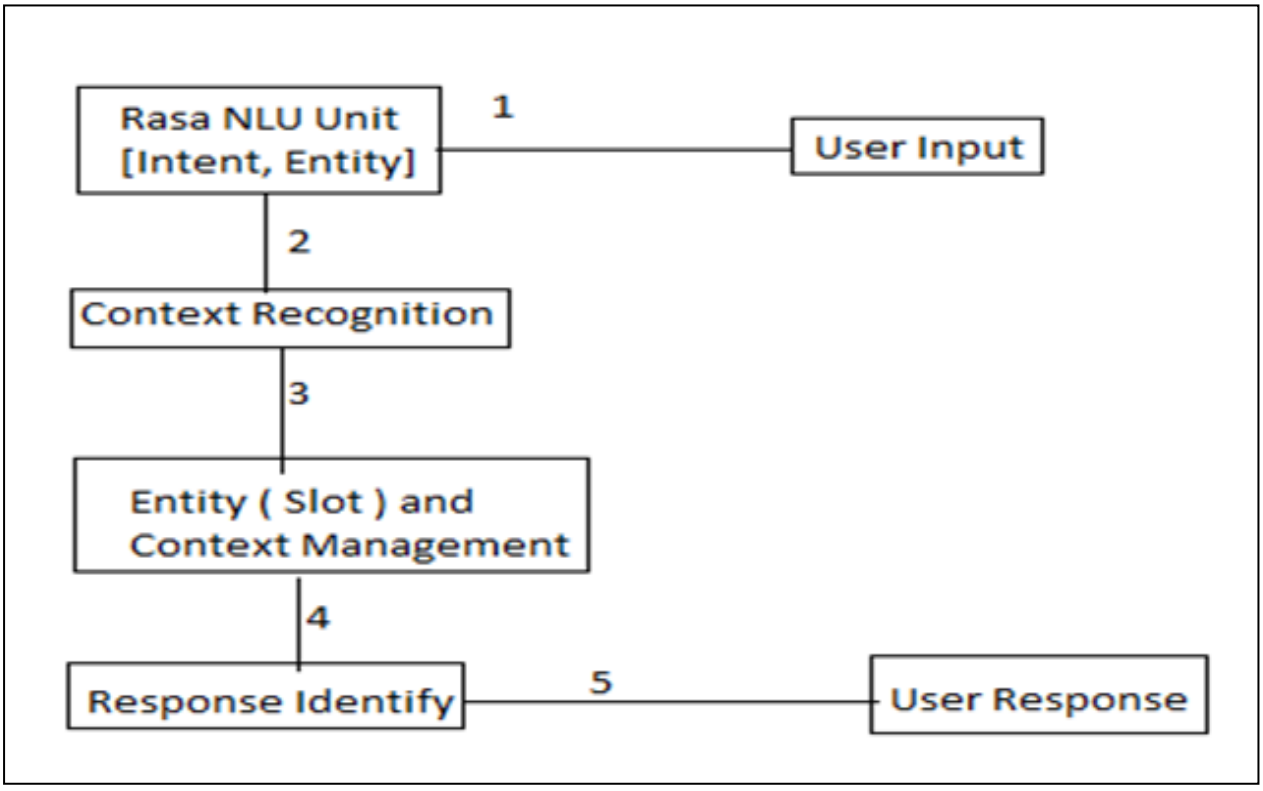

Figure 4 : RASA Conceptual Architecture

7. Development with architecture change for new model :

Proposed Solutions (Approaches):
1.Write short stories instead of end to end stories. In stories write only those actions together those must be performed.
2.Use concepts of checkpoints in stories to reduce effort. Implement new core from the scratch which is generic, scalable and reliable.

Problem with Proposed Solutions:

Problem with approach 1:
1. Still all above problem exists only their effects and limitation can be reduced and there is no fix figure to which this can be reduced as it is all implementer dependent.

Problem with approach 2:
1. Derive and define complete architecture for the platform. Since, core part defines behaviour of the chatbot.
2. A number of heuristic decision need to be taken for implementation.
3. Resultant core will be unreliable and take time to mature.

Modifications Required:

For Context Identification:
1. In respect of current rasa intent, there are few changes how to represent intent in NLU model file. Now, along with intent, provide context for intent respectively.
2. Context related information (like fallback, slots management for a context and context forwarding) need to be managed with in core part.

For Conversation:
1) Take user input and pre process it if needed. Find intent and entities from the input.
2) Change in input format for response corresponding to each intent.
3) Identify the response as per the intent.
4) Prepare the identified response and prepare it to send it as reply message to user. Currently, following type of response are considered:
    a) Static message
    b) External executable code
    c) Message with slot which can be filled from context data.

5) Extra information regarding context needed to be manage context, intent and entity.
6) Context and intent history need to be managed for validation of the system.

For changes of conversation few new files are generated by the system which contain consolidated information regarding context intent mapping, intent entity mapping, fallback information, response corresponding to a intent.

Context Identification:
Approach 1) Use single NLU model to identify all context

Approach 2) Use master slave like architecture. Use a global NLU model for identify context and then in particular context NLU model, identify intent of the input message.

Both approaches of context management are implemented. Following Points are observed:
1)When user frequently changes the context in conversation there is slight increase in response time in approach 2.

2)If two contexts have intent where they have similar input examples then wrong intent identification chances increase significantly in approach 2.
3)For implementing approach 2 for each context separate threshold need to be identified and managed.

****

Annexure – I

Vocabulary

Word	Definition
Intents	Things we expect the user to say.
Entities	These represent pieces of information extracted from what user said.
Templates	We define some template strings which our bot can say. The format for defining a template string is utter_<intent>. These are considered as actions which bot can take.
Actions	List of things bot can do and say. There are two types of actions we define one those which will only utter message (Templates) and others some customized actions where some required logic is defined.
Slots	These are user-defined variables which need to be tracked in a conversation. For e.g to buy term insurance, we need to keep track of what policy user selects and details of the user, so all these details will come under slots.