**AI Chatbot Documentation**

This Document addresses the overview of chatbot, product development and future work. For more technical details, please refer to API reference in ‘docs’ folder & README.md.

General Overview

Rasa is an open source framework which provides sufficient control in building a bot. The bot understands the intent of user, learns from it, respond intelligently and performs actions if required and all efficient learning mechanism.

Rasa comes with two components:

**1- RASA NLU --** it is a library for natural language understanding (NLU). this component does the classification of the intent and extracts the entity from the user input and helps the bit to understand what the user is saying.

**2- RASA CORE --** its a framework with machine learning based dialogue management which takes the structured input from NLU and predicts the next best action using a probabilistic model like LSTM neural network.

Before moving forward, let's see some definitions:  
  
**Intent --** Intentit is something that the user is aiming for. For example if the user says “what is my balance” the intent can be classified as to balance info.

**Entity --** Entity is to extract useful information from the user input. From the example above

“what is my balance” the entities extracted would be number verification i.e confirm the number against which the user wants balance information.

**Stories --** Stories define the sample interaction between the user and chatbot in terms of intent and action taken by the bot. Like in the example above bot got the intent of finding user balance and entities user number verification but still, sometimes next action is called from the bot.

**Actions --** Actions are basically the operations performed by the bot either asking for some more details to get all the entities or integrating with some APIs or querying the database to get or save some information.

Now let's see how the bot sees the data. For this we need to teach our bot to understand our messages first. To extract structured data we need to train the NLU model with our input in a simple text format. We do this by defining the intents and provide some ways in which the user expresses.

**NLU training file --** it contains some user inputs that is tegh training data along with the mapping of the intents and entities present in each of them. The bot’s NLU capabilities becomes better with providing more varying samples.

**Stories file --** this file contains the user and bot’s sample interactions which they will have. For each story Rasa Core creates a probable model of interaction.

**Domain file --** this contains all the intents, entries,actions, templates and some more information. The template which I mentioned above is a sample bot reply which can be used as actions.

Product Specifications

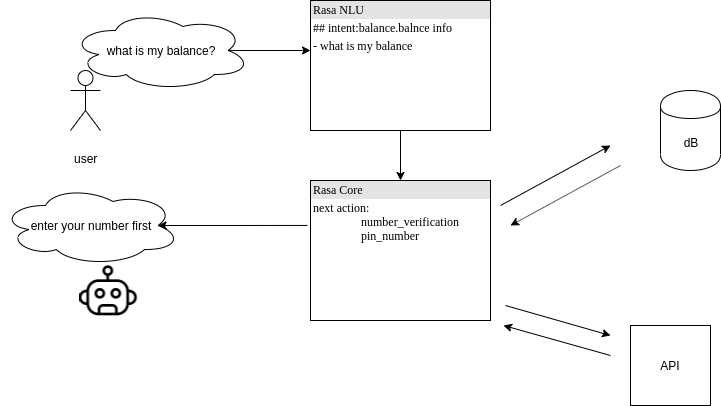
**Product overview**

An AI based conversational system with language understanding and dialogue management at its core which provides sufficient administration control in managing prebuilt conversational agents, updating the data, providing statistics for ongoing chat sessions with virtual as well as human agents. This system comes with other additional features including: personalized content (e.g. offers/ packages recommendation), aduitable logs & integration with existing chat platforms.

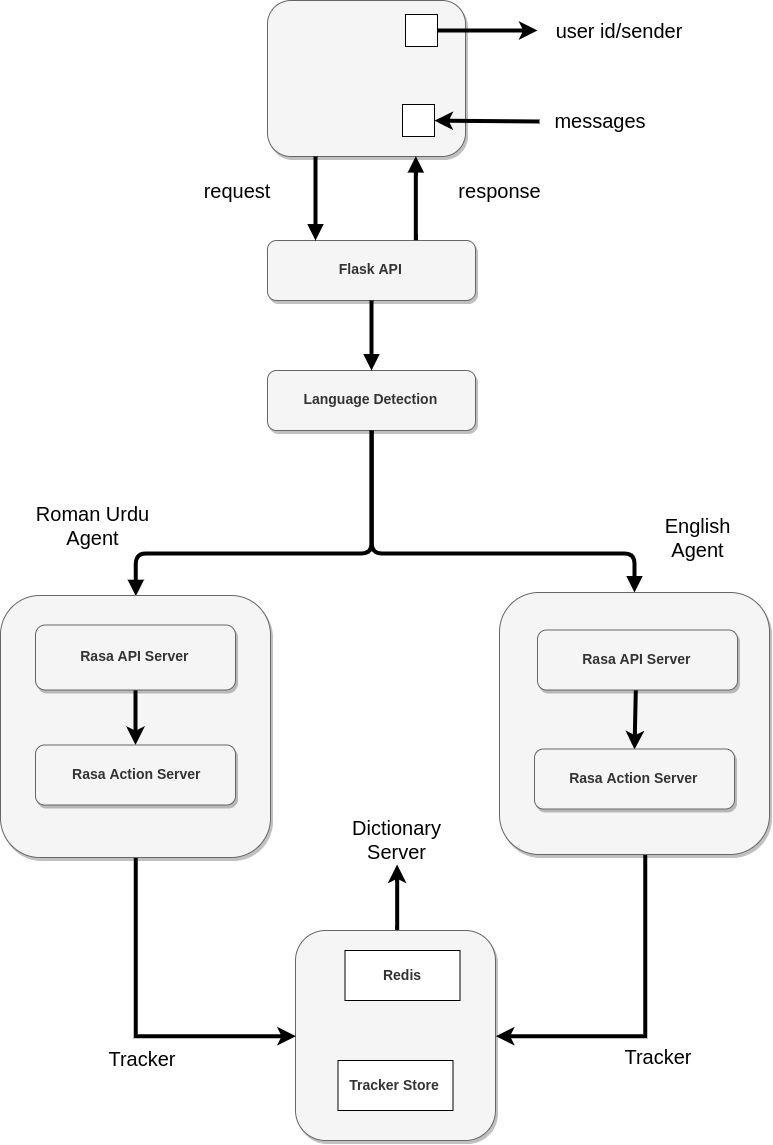
The following table explains the key modules available in product:

|  |  |
| --- | --- |
| **Module** | **Description** |
| Chatbot | Core chatbot engine with a pair of tools, Natural Language Understanding and user dialogue management. |
| Admin Portal | Admin panel to review stats, managing bots & updating data etc. Admin panel provides you control over the whole system. Authorized users can manage existing chatbot agents, updating agents data, look up tables, viewing the on-going live chat sessions with both AI agent and human agents (during handoff). |
| Continuous Development | Continuous development caters the problem of updating the agents with new conversational data, gathered during human-to-human conversation, in a semi-supervised fashion. |

# **Chatbot** Chatbot’s architecture is modular by design. Dialogue management is used in conjunction with NLU services for complete conversational flow. The bot understands the intent of user, learns from it, respond intelligently and performs required actions.

  
**Architecture:**

Following is the working diagram of whole chatbot in a bigger picture. When the user asks a question to a bot for example as mentioned in the above that when the user asks “what is my current balance in the account”. This user request is sent to the flask API which further goes to the language detection phase which detects whether the question asked by user belongs to Roman Urdu or English language. In both cases the respective API Server and the Rasa Action Server is connected. Redis is used as back-end dictionary server to keep a record of each user the tracker is used.



Following table lists down available features in this module:

|  |  |
| --- | --- |
| **Key Features** | **Description** |
| Intent Prediction | This module is responsible for natural language understanding (NLU). This component does the classification of the intent and extracts the entity from the user input and helps to understand what the user is saying. |
| Dialogue Management | This part does machine learning based dialogue management which takes the structured input from NLU and predicts the next best action using probabilistic models.  Dialogue state is saved in a tracker object. There is one tracker object per conversation session, and this is the only stateful component in the system. A tracker stores slots, as well as a log of all the 2 events that led to that state and have occurred within a conversation. The state of a conversation can be reconstructed by replaying all of the events. |
| User Authentication/ Authorization | Users authentication can be done through:   1. User login details from existing user base credentials (username/ password) 2. PIN code authentication through otp-sms service. |
| Spell Check | Misspell and deliberate short words cause a lot of issues in intent prediction. This feature will handle:   1. Sounds like variants for both Roman Urdu and english will get matched with original words. For example ‘blnc’ is a deliberate misspelled short form of word ‘balance’. 2. Misspell corrections will be only done for english language. Since there are a lot of variants of each word in roman urdu that come from different accent and classes of consumers and they are all considered correct. For example: ‘kia’, ‘kya’,’kea’ etc. 3. Proactively suggesting words to user during typing. This is an optional feature and will require to design language generation models for both Roman Urdu & English. |
| Small Talk | Small talk with user other than business related intents. Small talk can be extended , but in initial version it’ll include:   1. Greeting 2. How are you 3. Need help 4. Is there anyone 5. Agent.helpful 6. Agent.happy/good/excited 7. User.happy/good/excited 8. Appraisal 9. Dialog sorry 10. Goodbye 11. Good evening 12. Good morning 13. Good night 14. Nice to meet you/ see you/ talk to you |
| FAQs Knowledge Base [optional] | Knowledge Connectors will allow you to bulk add data from your enterprise to your agent, including FAQs and knowledge-base articles.  e.g Mapping to existing knowledge base for FAQs [for instance if 1000+ FAQs already listed somewhere for general queries regarding services]. |
| Personalized Content | This module can be used to bring in personalized content to users. This module will be connected to existing data source for fetching available services/ offers.  These services/ packages will be recommended to users based on:   1. Personalized based on user interests(he likes monthly or weekly or yearly subscriptions for example) 2. Context based (what has the conversation been about - balance, mobile data, sms etc.) |
| Business Logic Prompts | Re-confirmation from user before executing a business logic on behalf of user. For example, subscribing to a service/ balance transfer.  User will be authorized for business operations using sms-otp service. |
| Low Confidence Prompts | If confidence for user intent drops below a certain threshold, instead of activating wrong action the bot will help user with suggested actions once to handle fallback gracefully. |
| Integrations | Integrations with the on premises system:   1. Private cloud 2. Slack 3. Messenger 4. WhatApp |

# 

# **Admin Portal**

This tool is the only place to view and manage chatbot agents and other functionalities. Features in admin portal can be extended in future, but these are the features that will be exported with first release.

|  |  |
| --- | --- |
| **Key Features** | **Description** |
| Route to Support if Chatbot Fails | Whenever the bot fails to understand the user intention/ query a couple of times consecutively, control will be transferred to a human agent. This threshold can be changed depending on needs. For now it's set to 3 consecutive fallbacks. Human agent can transfer the control back to the bot after resolving the issues. |
| Logs | This services will write auditable logs to help you answer the questions, "Who did what, where, and when?". Types of logged information can be extended to admin activity logs, data access logs, event logs other than simple conversational logs. |
| Live Conversations | Authorized users can view live conversation sessions with bot and/or human agents. |
| Analytics | 1. Total number of conversations 2. Average message per conversation 3. Max conversation messages 4. Confidence in conversations    1. Failed    2. Passed 5. Intents wise statistics    1. breakdown of the above numbers by intent  (most common intent, least common intent etc) |

# **Continuous Development**

# To be implemented

This is the feedback and training process.

# 

# **Data Security**

1. On-premises
2. No external apis
3. Private cloud by customer

**Resources**

RASA, FastText

# **Technical Specifications**

1. Web based admin portal
2. Web based client window
3. Android/iOS support