**Creation of Joybot**

**Terminology and Approach:**

This chatbot is developed using RASA (Receive, Appreciate, Summarize, Ask) Framework and NLP (Natural Language Processing).

Rasa is a python framework that helps us to build any kind of Chatbot easily. It is based on NLU (Natural Language Processing) which offers the possibility to understand what the user wants.

The Chatbot is composed by some specifics terms that we will discover:

* **Intents**: It represents intentions that the user wants to express when he submits his message to the Chatbot. Inside the intents, we provide some phrases that the user may ask and we also provide some responses that the Chatbot must use to answer to the user. It’s the work of the Developer here.
* **Entities**: Entities are keywords that represent some specific dates that the Chatbot may use to perform the discussion with the user. Entities are used to extract some values inside the user input (message).

Now, we know how Chatbot works, we can switch to the Rasa Framework. Rasa is split into two python libraries like Rasa NLU and Rasa Core. The first one is the Natural Language Understanding and uses the intents to understand what the user wants and entities to extract some specific values to make the conversation more interesting, after Rasa NLU has understood what the user wants and extracted some values, it sends it to the next one. Rasa Core receives the data sent by Rasa NLU and processes it to find the correct answer that it should send to the user as output, for having a response, it will look for the responses that the Developer provided to him inside the intents.

**Creating FAQs and Stories:**

* Initial Step was to make FAQs, we used one-intent FAQ and made numerous FAQs (faq/causes\_mental\_illness) in NLU.yml with answers in domain.yml (utter\_faq/causes\_mental\_illness). Finally used rule.yml to define the functionality (whenever faq is called by user, bot must perfrom utter\_faq)
* For creating the conversation, we used stories.yml, where we defined different approaches possible by the user. This functionality includes various options, buttons, replies etc.

The project consists of the following files:

* **nlu.yml**- This file contains the list of intents and their possible sample text. These intents are used to train our NLU model.
* **domain.yml**- This file lists all intents, the bot’s responses with their text, and also the action that can be performed by the bot. Note, any action or response we define has to be listed in the domain.yml
* **stories.yml**- This file gives the bot an idea of how the conversation should flow. These stories are defined in the form of markdown language. We delete some predefined stories and replace them with those conversations which are relevant for our bot.
* **rules.yml** - This file contains certain rules that bot should respond whenever user gives a specific intent. For e.g. Our bot returns emergency contacts whenver user inputs words like ``emergency``, ``help me``, ``code red`` etc.

**Chatting Stories**

**User**

**Chatbot**

* **Hi**
* **Greet and ask name**
* **Nikhil**
* **Gender? Buttons: 3 Options**
* **Select Gender**
* **Ask how are you?**

* **Good:**
* **Glad to hear that. Ends the conversation**

* **If Sad / not good:**
* **Ask reason (Buttons: 4 options):**
* **Give reason**
* **Consolidate and ask for a more precise reason.**
* **Give a more precise reason.**
* **Consolidate by suggesting good thoughts and ask if that helped**
  + **If yes:**
  + **Glad to hear that. Ends the conversation**

* **If no:**
* **Ask more specific reasons (Buttons:3 options)**
* **Select the reason.**
* **Give a reply as per the reason selected and ask if that helped.**
* **Reply Yes (as this is end of all conversations)**

**Rules**

* **Bot will reply to FAQs whenever asked.**
* **Bot will give emergency contacts whenever asked**
* **Bot will explain about its functionality whenever user calls out ``joybot``**
* **Bot will say goodbye whenever the user says goodbye.**
* **Bot will introduce itself whenever user asks ``who are you``**

**Formatting Stories**

**Instead of having various Stories (each for each scenario). We tried to set ``checkpoints`` and ``or`` statements.**

**Checkpoints save the question asked and let us continue from that question in the next story. Usually used when two stories have very similar paths.**

**Or helps in situations where intents in the story could be different but the action connected are the same.**

**Using this we created following Stories:**

* **Talking starts (checkpoint start-open)**
* **If user feels good (checkpoint open-closed)**
* **User not feeling good (checkpoint open-open)**
* **From specific problem to feeling good (checkpoint open-close)**
* **From specific problem to not happy (checkpoint open-open)**
* **Still not happy to more helpful resources - Websites (checkpoint open-close)**
* **Still not happy to more helpful resources - Contacts (checkpoint open-close)**
* **Still not happy to more good habits (checkpoint open-close)**
* **When the process ends (checkpoint closed-end)**