Chat Bot For SRET

Yaeshwanth Urumaiya (E0122028)
BTech AIML (1st Year)

Prof. Dr. Vanitha Magesh
(Internship Guide)

The Abstract

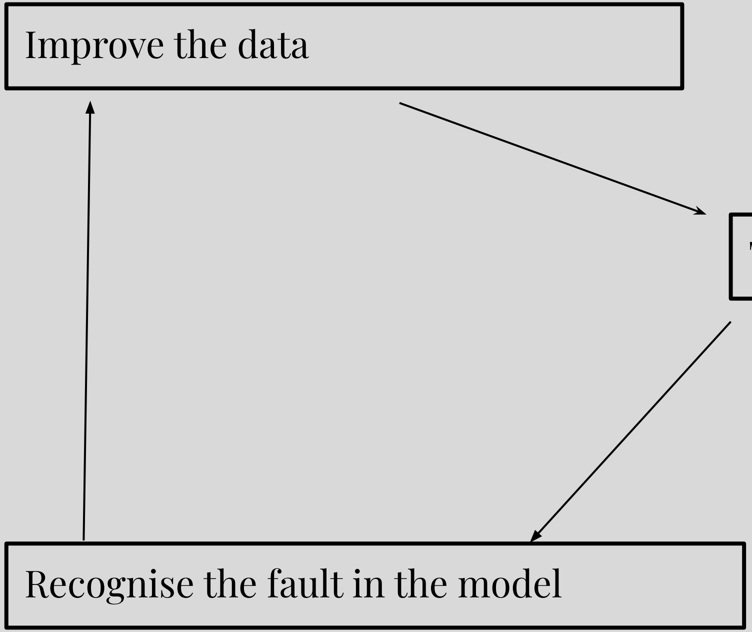
This project demonstrate a chatbot created for our college to enhance communication and give visitors to our website an engaging experience. The RASA Framework's chatbot makes use of powerful natural language processing and artificial intelligence tools to efficiently interpret and address user inquiries. Through the chatbot, users can access info about admissions, courses, and more

Problem Statment

Chat Bot

The need for an effective and user-friendly communication platform for our college is something that this project attempts to solve. Traditional information access techniques, such browsing websites or speaking with college staff members, often lead to delays and inefficiencies. To address this issue, I suggest creating a chatbot to serve as a virtual assistant.

Working with Rasa is pretty simple.



Improve the data

Train the model

Recognise the fault in the model

Work-Flow

It is pretty easy. The entire chatbot is programmed into a small widget, which can be easily applied to any website.

Scope and Scalability

Github Repo Usage

With the front-end of the widget and then the model working, what else should we add?

The actual JavaScript to connect the frontend to the Terminal based Chatbot responses.

But with the help of RASA Framework's documentation and Resources, we can use the pre-defined Script which does this for us automatically.

But that file only supports Rasa 1.1X - 2.8X. As we were using Rasa 3.1X, I opted for a GitHub Repo which updated the official resource Script to work alongside the latest version of Rasa

```
onst quickReplies = `<div class="quickReplies">${chips}</di
(quickReplies).appendTo(".chats").fadeIn(1000);
crollToBottomOfChat();

onst slider = document.querySelector(".quickReplies");
et isDown = false;
et startX;
et scrollLeft;

lider.addEventListener("mousedown", (e) => {
 isDown = true;
slider.classList.add("active");
tartX = e.pageX - slider.offsetLeft;
crollLeft = slider.scrollLeft;

r.addEventListener("mouseleave", () => {
own = false;
er.classList.remove("active");

addEventListener("mouseup", () => {
 = false;
classList.remove("active");

EventListener("mousemove", (e) => {
wn) return;
Default();
 x = e.pageX - slider
const walk = (x
slider.sc
```

Is it good?

Yes, It responds to the user in a very decent rate of intent recognition and the replies as just as you would expect from an production level project.