Documentation for PyGyan.AI – AI chatbot for Pytorch

Developed by: Madhav Dhadwal

About PyGyan.AI

PyGyan.AI is an artificially intelligent chatbot that can respond to user questions based on the PyTorch documentation. The name PyGyan.AI is derived from two words: "Py" refers to Python (specifically Pytorch) and "Gyan" implies knowledge.

PyGyan.AI is totally created from scratch with the Rasa framework, which is one of the best open source framework to build chatbots. The chatbot's user interface was created using the Flask web framework. The data required to train the bot was scraped from the official Pytorch documentation website using the scrapy framework, which is a popular open source framework for web scraping/crawling.

Steps in creation of PyGyan.AI

1. The data involved to train an AI chatbot is the most important foundation for creating the chatbot. The quality of the training data provided to the chatbot determines how well it performs overall. If the data is noisy, the chatbot's responses will be inaccurate and irrelevant. So the first step in creating my Rasa chatbot is to acquire data. I used official Pytorch documentation as the data source for training my chatbot.

2. The second step was to scrape data from the Pytorch documentation page. To scrape the web, I utilized the scrapy web crawling framework. I saved the data in a.json file.

3. Then I transformed the.json file into a Rasa-compatible format. Also, I made ‘intents’, ‘examples’ and ‘responses’ out of the data.

4. Now, after the preprocessing of data as mentioned in step 3, I prepared the ‘nlu.yml’ file of the Rasa project. This file contains the training examples for the NLU (Natural Language Understanding) model.

5. After preparing the nlu.yml file, I defined the rules i.e specific conditions and actions to be executed when those conditions are met in the ‘rules.yml’ file.

6. After defining rules, I prepared the stories.yml file. This file contains story examples that represent different conversational paths or scenarios between the user and the chatbot. These

stories are used to train the dialogue management model in Rasa.

7. Then, I prepared the most important file i.e the domain.yml file of the model, Generally it includes the intents, entities, actions, templates for responses, slots, and other configuration settings

8. I also configured the project by specifying the pipeline for NLU, the policies for dialogue management, and other training settings in the config.yml file.

9. After all this, I defined the functions in the main python script i.e actions.py (I Specified the actions that the chatbot can take in response to user input)

10. To train the final model I used the command ‘rasa train’

11. To run the chatbot on terminal, I used the commands ‘rasa shell’ and ‘rasa run actions’ simultaneously in two terminals.

12. To make a user interface for the chatbot, I integrated it with flask. I made the frontend in ‘index.html’ and backend in the ‘app.py’ file.

13. To run the chatbot on my flask application, I ran [rasa run --enable-api --cors "*"] and [rasa run actions] simultaneously in two terminals and ran the app.py python script using the ‘run’ button.

Structure of PyGyan.AI

1) actions folder: This folder contains the custom actions for your chatbot. It includes an actions.py file where you can define the logic and behavior of your custom actions. You can add more Python files for additional actions as your project grows.

2) data folder: This folder stores the training data for your chatbot. It includes some main files:
   - nlu.yml: This file contains the training examples for the NLU (Natural Language Understanding) model.
   - rules.yml: This file includes rule-based conversation rules. Rules define specific conditions and actions to be executed when those conditions are met.
   - stories.yml: This file in your default project contains story examples that represent different conversational paths or scenarios between the user and the chatbot. These stories are used to train the dialogue management model in Rasa.models folder: This folder stores the trained models of your chatbot. After training, Rasa will save the trained models (including NLU and dialogue management models) in this folder.

3) tests folder: This folder is used for testing your chatbot.

4) config.yml: This file contains the configuration settings for your Rasa project. It specifies the pipeline for NLU, the policies for dialogue management, and other training settings.

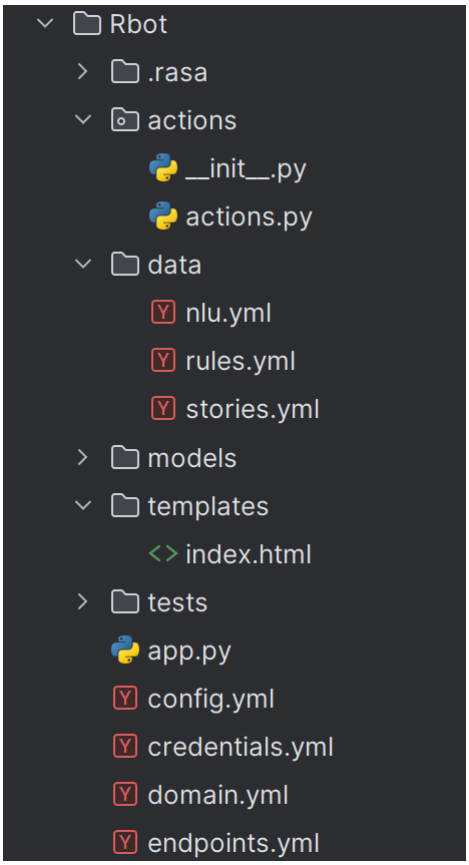
5) credentials.yml: This file is used to configure the credentials for external services or APIs that your chatbot may need to interact with, such as databases, APIs, or channels like Slack.

6) domain.yml: This file defines the domain of your chatbot. It includes the intents, entities, actions, templates for responses, slots, and other configuration settings. You define the structure of the conversation in this file.

7) endpoints.yml: This file defines the endpoints for your chatbot, including the webhook URL for custom actions and any external services that your chatbot needs to connect to.

8) templates folder: This folder contains the index.html file which is responsible for the frontend of User Interface of Chatbot.

9) app.py: This file contains the backend code which integrates flask web application with my Rasa model.



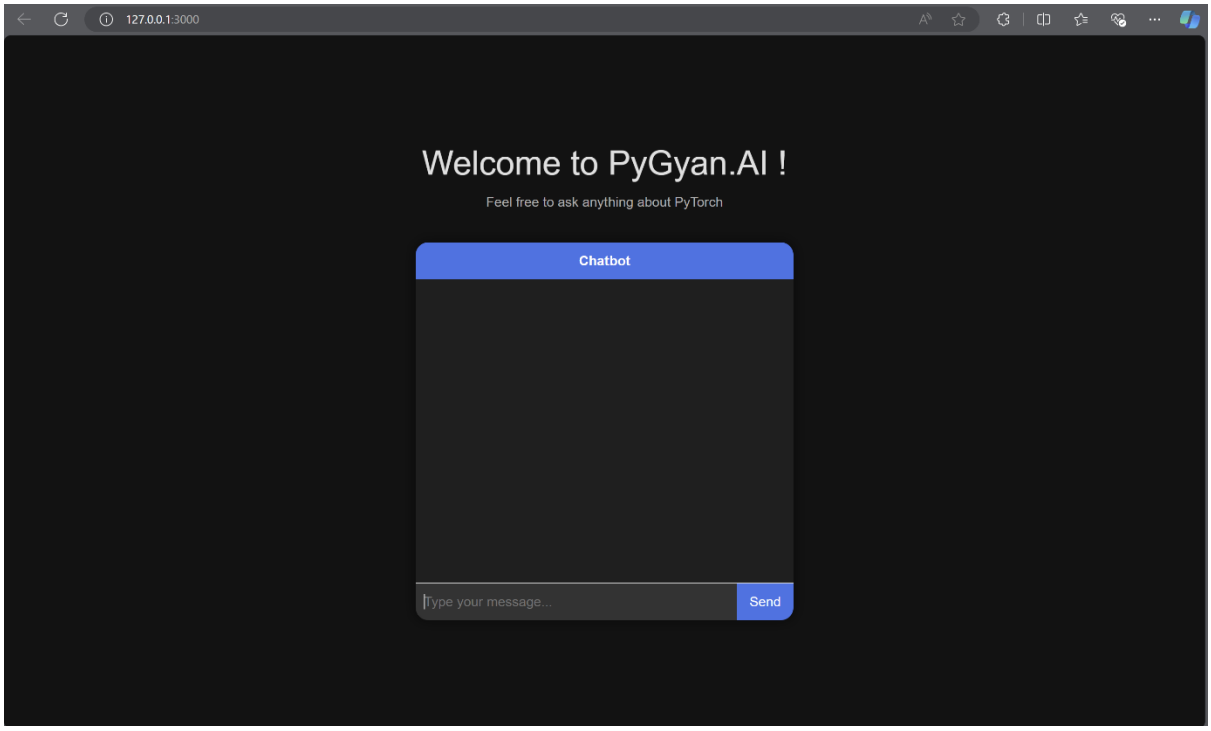
Please note :

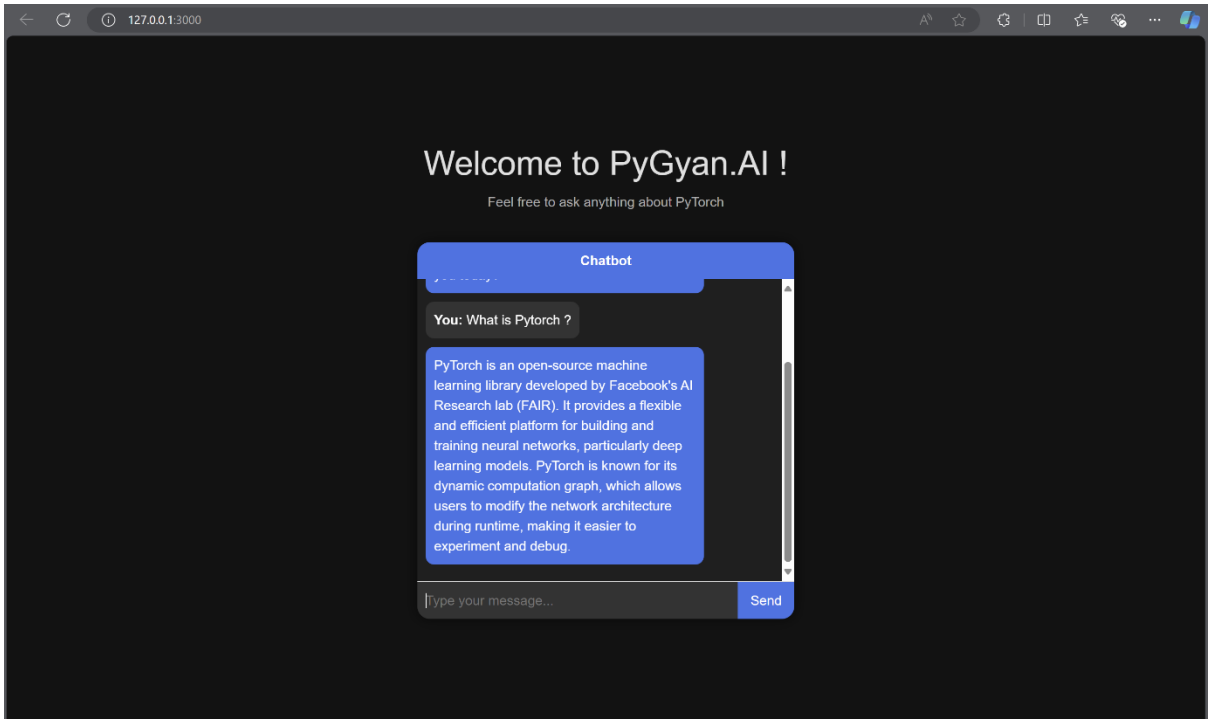
I have attached the folder which contains all the files of the project (including the source code).

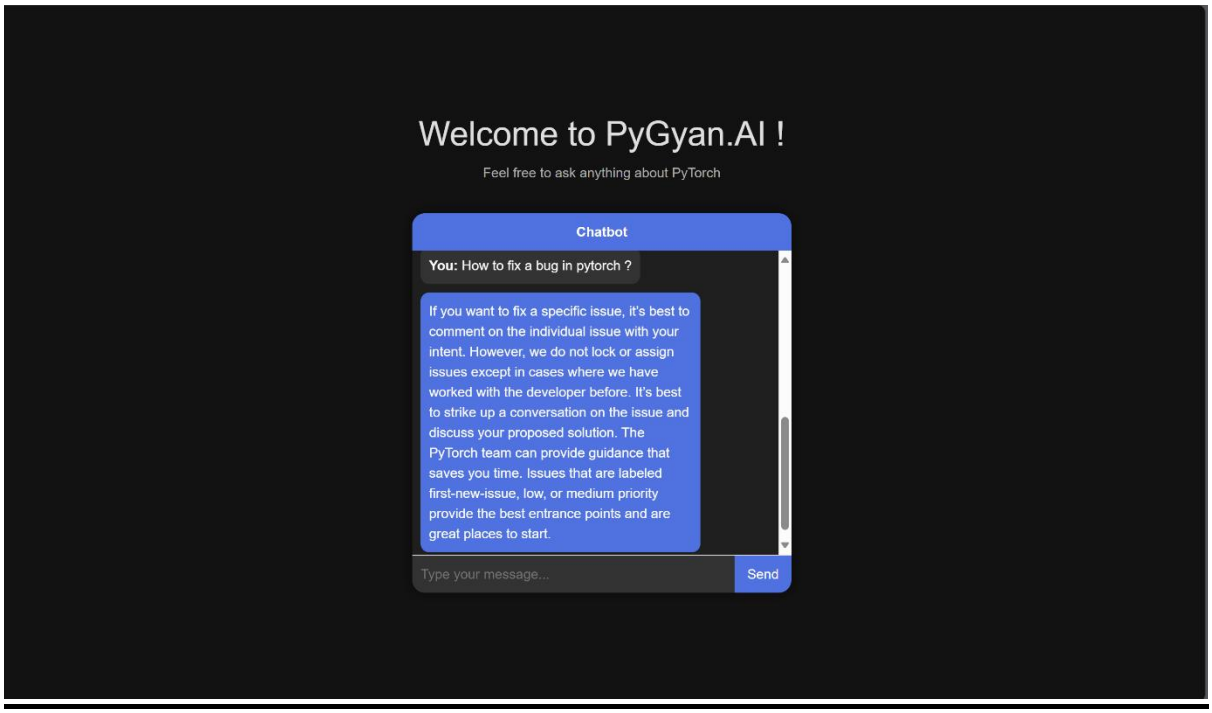
I have also made a short screen recording video of working of my Chatbot in real time, please find it in the link below :

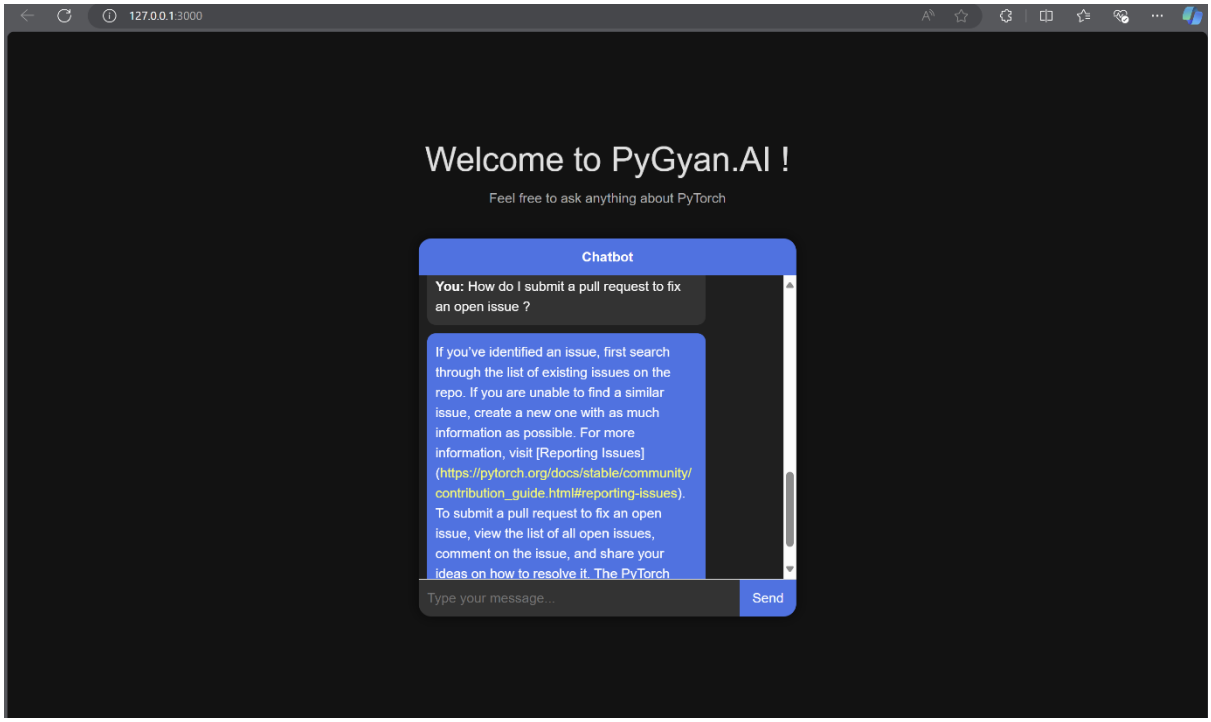
https://drive.google.com/file/d/1VbhX93Fk7rXmI0TmCPrlJny9sBGc5u1l/view?usp=sharing

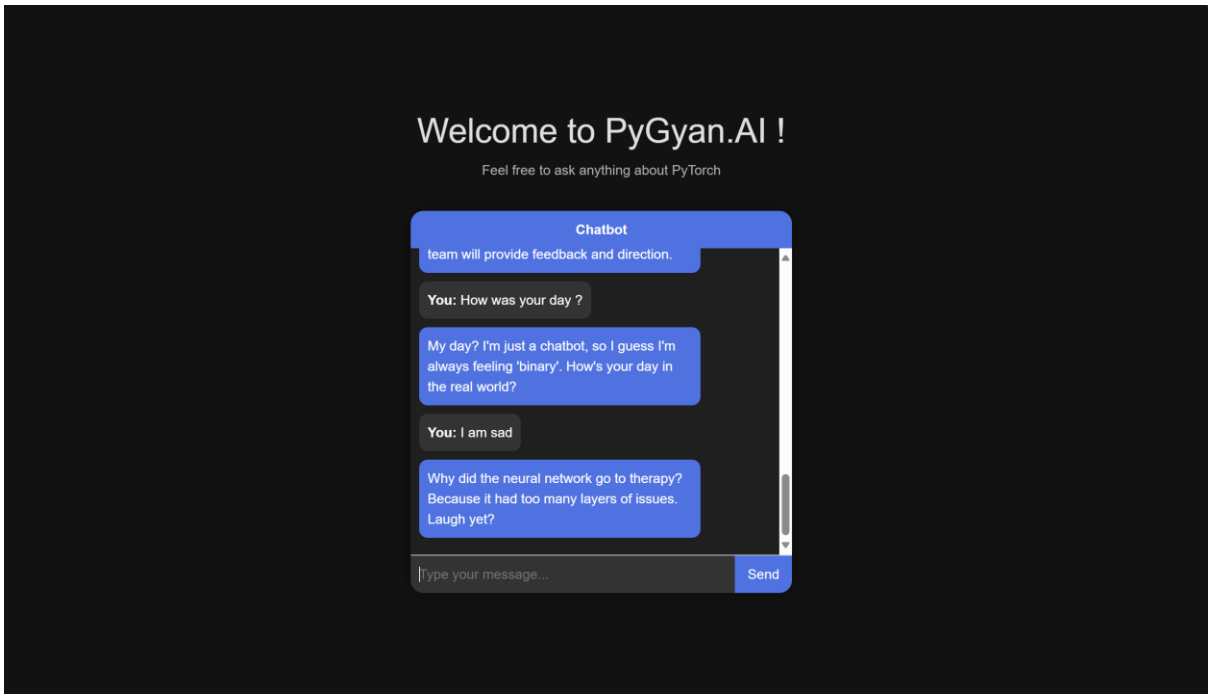
Screenshots of PyGyan.AI in action



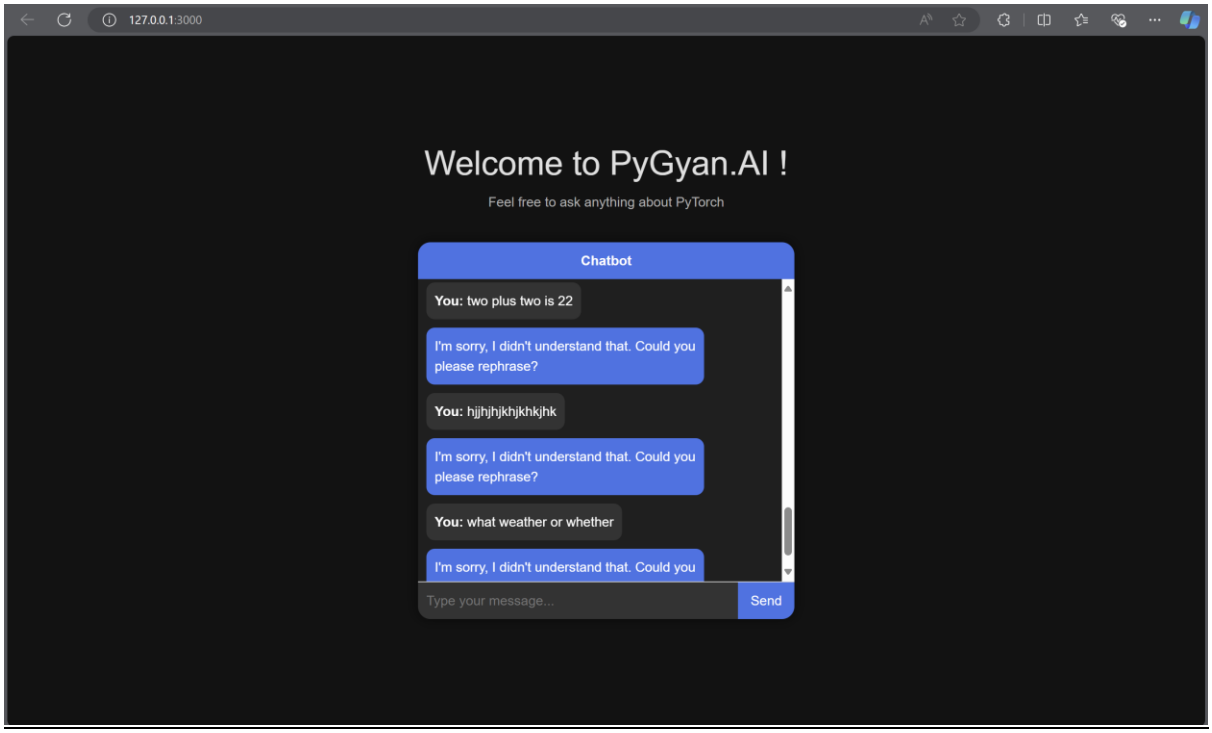

127.0.0.1:3000
Welcome to PyGyan.AI !
Feel free to ask anything about PyTorch
Chatbot
You: What is Pytorch ?
PyTorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It provides a flexible and efficient platform for building and training neural networks, particularly deep learning models. PyTorch is known for its dynamic computation graph, which allows users to modify the network architecture during runtime, making it easier to experiment and debug.
Type your message...
Send


Welcome to PyGyan.AI !
Feel free to ask anything about PyTorch
Chatbot
You: How to fix a bug in pytorch ?
If you want to fix a specific issue, it's best to comment on the individual issue with your intent. However, we do not lock or assign issues except in cases where we have worked with the developer before. It's best to strike up a conversation on the issue and discuss your proposed solution. The PyTorch team can provide guidance that saves you time. Issues that are labeled first-new-issue, low, or medium priority provide the best entrance points and are great places to start.
Type your message...
Send





Can handle day-to-day conversations



Can handle unknown queries

Thank you,

Madhav Dhadwal

+91 9817828122

madhavdhadwal@gmail.com