PROJECT REPORT

on

CSE Department Chatbot

(CSE III Semester Mini project)

2024 - 2025

Submitted to:

Mr. Aakash Chauhan

(CC-CSE-K1-III-Sem)

Submitted by:

Ms. Anushka Negi

Roll. No: 2318473

CSE-K1-III-Sem

Session: 2024-2025

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"CSE Department Chatbot"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering (AI-ML) in the Department of Computer Science and Engineering of the Graphic Era Hill University, Dehradun.

Name: Anushka Negi University Roll no: 2318473

The above mentioned student shall be working under the supervision of the undersigned on the **"CSE DEPARTMENT CHATBOT"**

Supervisor **Head of the Department**

Examination

Name of the Examiners: **Signature with Date**

1.

2.

ACKNOWLEDGEMENT

I extend my heartfelt thanks to my parents for their constant support and encouragement. I also wish to thank them for providing me the opportunity to reach this far in studies.

Special thanks to my project Co-ordinator Mr. Aakash Chauhan for his patience, support and encouragement throughout the completion of this project and having faith in me. I am also grateful to all my professors and peers who directly or indirectly supported and helped me during this work.

Ms. Anushka Negi

Roll No: 2318473

CSE-K1-III-Sem

Session: 2024-2025

GEHU, Dehradun

TABLE OF CONTENTS

LIST OF SYMBOLS, ABBREVIATIONS

1. **NLP: Natural Language Processing**
    - Explanation: A branch of AI that focuses on enabling machines to understand and process human language.
2. **AI: Artificial Intelligence**
    - Explanation: Simulation of human intelligence in machines capable of performing tasks requiring human-like thinking.
3. **ML: Machine Learning**
    - Explanation: A subset of AI focused on systems that learn from data to make predictions or decisions.
4. **JSON: JavaScript Object Notation**
    - Explanation: A lightweight data-interchange format used for storing structured data.
5. **BoW: Bag of Words**
    - Explanation: A method of text representation in which text is converted into a set of words and their occurrences.
6. **ReLU: Rectified Linear Unit**
    - Explanation: An activation function used in neural networks to introduce non-linearity.
7. **Adam Optimizer: Adaptive Moment Estimation Optimizer**
    - Explanation: A popular optimization algorithm for training machine learning models.
8. **Cross-Entropy Loss:**
    - Explanation: A loss function used for classification problems to measure the difference between predicted and true labels.
9. **Epoch:**
    - Explanation: A single pass through the entire dataset during training.

CHAPTER 1

INTRODUCTION

1.1 ABOUT PROJECT

The CSE Department Chatbot is an Artificial Intelligence-based virtual assistant designed to enhance and simplify communication within the department. It provides instant responses to queries related to various aspects of the department, such as faculty details, placements, syllabus, lab facilities, events, and more.

This chatbot demonstrates the effective use of Natural Language Processing (NLP) and Machine Learning (ML) techniques to create an interactive tool. By automating the process of answering frequently asked questions, the chatbot saves time and reduces the workload on administrative staff. It is designed to integrate seamlessly into departmental operations, offering students and faculty a user-friendly platform for obtaining information.

The chatbot employs a structured dataset consisting of intents, patterns, and responses. Through a robust neural network model, it processes user input and generates appropriate responses. The project highlights the practical applications of NLP in building intelligent conversational systems.

1.2 TECHNOLOGIES USED

The development of the CSE Department Chatbot required a combination of robust programming tools, libraries, and frameworks:

Programming Language

- **Python:** Python was chosen due to its simplicity, versatility, and support for NLP and Machine Learning libraries. Its concise syntax and readability make it ideal for developing AI applications.

Libraries and Frameworks

1. **PyTorch:**
    - Used for building and training the neural network.
    - Offers dynamic computation graphs and easy debugging tools.
2. **NLTK (Natural Language Toolkit):**

    - Provides utilities for tokenizing sentences, stemming words, and text processing.
    - Essential for implementing preprocessing steps in NLP tasks.
3. **NumPy:**
    - Facilitates numerical computations, including array manipulations and mathematical operations.
    - Integral to the implementation of the Bag of Words model.

Dataset Format

- **JSON:** The dataset containing intents, patterns, and responses was structured in a JSON file. This format ensures ease of data retrieval and modification during development.

CHAPTER 2

PROJECT DEVELOPMENT

2.1 REQUIREMENT ANALYSIS

Hardware Requirements

1. Processor: A dual-core or higher processor ensures smooth computations.
2. RAM: Minimum of 4GB for efficient processing.
3. Storage: At least 20GB of free disk space to store libraries and datasets.

Software Requirements

1. Operating System: Windows 10, macOS, or Linux.
2. Python 3.8+: The primary programming language used for development.
3. Development Tools: Jupyter Notebook and Visual Studio Code for code writing and debugging.
4. Libraries: PyTorch, NLTK, and NumPy for model development and text processing.

2.2 SOFTWARE SPECIFICATIONS

The chatbot's core backend functionalities include:

1. **Data Preprocessing:**
    - Tokenization: Breaking sentences into words.
    - Stemming: Reducing words to their root forms.
    - Removing punctuation and irrelevant characters.
2. **Feature Representation:**
    - Using the Bag of Words (BoW) model to numerically represent text data.
3. **Model Training:**
    - A feedforward neural network processes the data and predicts intents.
4. **Response Generation:**
    - Selecting and returning the most relevant response from the dataset.

2.3 SYSTEM ARCHITECTURE AND WORKFLOW

The chatbot system comprises the following steps:

1. **User Input:** A query in natural language is entered by the user.

2. **Preprocessing:** The query undergoes tokenization, stemming, and numerical transformation using the Bag of Words model.
3. **Neural Network:**
    - The processed input is fed into a trained model.
    - The model predicts the corresponding intent.
4. **Response Selection:** The system retrieves the appropriate response associated with the predicted intent.

CHAPTER 3

IMPLEMENTATION

3.1 DATA PREPROCESSING

The preprocessing steps ensure consistent and meaningful input for the neural network:

1. Tokenization: Splitting input sentences into individual words.
2. Stemming: Reducing words to their base forms (e.g., "running" → "run").
3. Bag of Words Representation: Converting processed words into a vectorized format, enabling the neural network to interpret the data.

3.2 MODEL TRAINING AND TESTING

Model Architecture:

1. Input Layer: Matches the size of the vocabulary.
2. Hidden Layer: Eight neurons using ReLU activation.
3. Output Layer: Corresponds to the number of unique intents.

Training Details:

- Optimizer: Adam optimizer for effective learning.
- Loss Function: Cross-entropy loss for minimizing prediction errors.
- Epochs: Trained over 1000 iterations to achieve optimal performance.

3.3 SYSTEM FEATURES

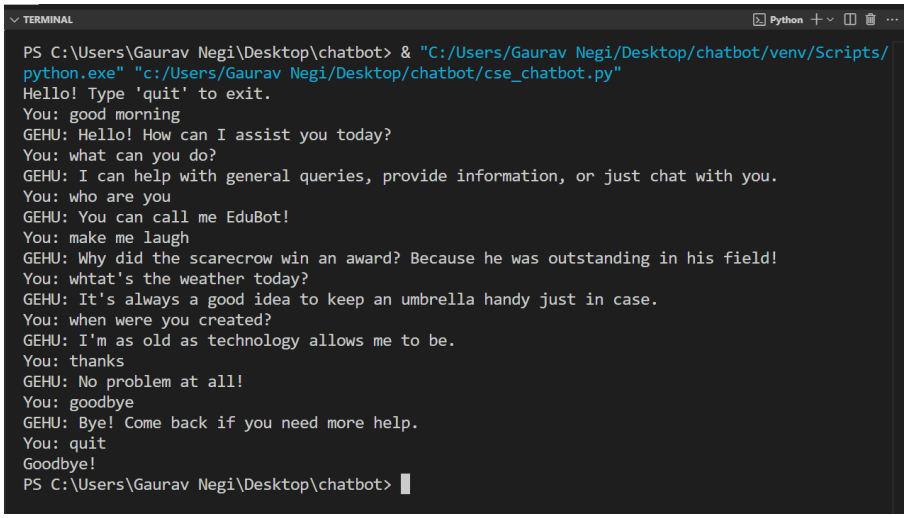
The chatbot offers the following features:

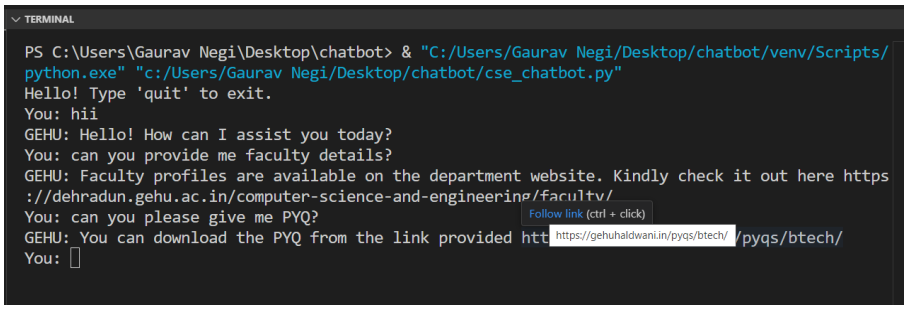
1. **Accuracy:** High precision in understanding and responding to user queries.
2. **Interactivity:** Real-time response generation.
3. **Scalability:** Easy to expand with additional intents and responses.
4. **User-Friendly Design:** Accessible and intuitive interface.

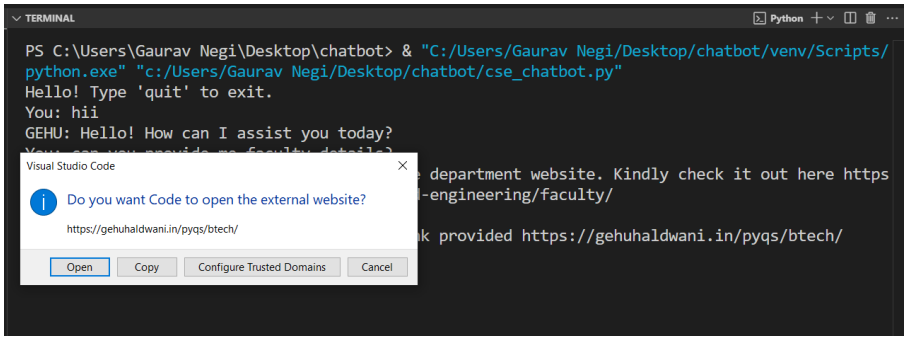
CHAPTER 4

RESULTS

4.1 SNAPSHOTS OF PROJECT







4.2 DATASET OVERVIEW

The dataset comprises intents categorized by tags, patterns, and responses. Example:

Intent: Greeting

- Patterns: Hi, Hello, Good morning.
- Responses: Hello! How can I assist you today?

CHAPTER 5

CONCLUSION

5.1 SUMMARY

The CSE Department Chatbot demonstrates the effective application of NLP and machine learning to automate query handling. It significantly enhances communication efficiency within the department.

5.2 FUTURE SCOPE

Future improvements will focus on enhancing voice interaction, multi-language support, and contextual understanding.

1. Incorporating more advanced NLP models like LSTM or transformers for better accuracy.
2. Integrating real-time data fetching for more dynamic responses.
3. Enhancing the chatbot's interactivity with personalized queries and user feedback.
4. Next Steps: Deployment in the student portal or as a web app for easy access.
5. Adding voice interaction to improve accessibility.
6. Expanding the dataset to cover a broader range of queries.
7. Supporting multiple languages for diverse users.
8. Deploying the chatbot on platforms like WhatsApp, Telegram, or departmental websites.

APPENDIX (Code)

1. nltk_utils.py

```python
import numpy as np
import nltk
from nltk.stem.porter import PorterStemmer
stemmer = PorterStemmer()

def tokenize(sentence):
    return nltk.word_tokenize(sentence)

def stem(word):
    return stemmer.stem(word.lower())

def bag_of_words(tokenized_sentence, all_words):
    tokenized_sentence = [stem(w) for w in tokenized_sentence]
    bag = np.zeros(len(all_words), dtype = np.float32)
    for idx, w in enumerate(all_words):
        if w in tokenized_sentence:
            bag[idx] = 1.0
    return bag

```

2. training.py

```python
import json
import numpy as np
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from nltk_utils import tokenize, stem, bag_of_words

with open('cse_chatbot_dataset.json', 'r') as f:
    intents = json.load(f)

# Prepare data
all_words = []
tags = []
xy = []
for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        words = tokenize(pattern)
        all_words.extend(words)
        xy.append((words, tag))
# Stem and sort
ignore_words = ['?', '!', '.', ',']
all_words = [stem(w) for w in all_words if w not in ignore_words]
all_words = sorted(set(all_words))
tags = sorted(set(tags))
# Create training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    label = tags.index(tag)
    y_train.append(label)
X_train = np.array(X_train)
y_train = np.array(y_train)
```

training.py > ...

```python
# Neural Network Model
class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]
    def __len__(self):
        return self.n_samples
# Hyperparameters
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
batch_size = 8
learning_rate = 0.001
num_epochs = 1000
# Dataset and Dataloader
dataset = ChatDataset()
train_loader = DataLoader(dataset=dataset, batch_size=batch_size, shuffle=True)

```

training.py > ...

```python

# Model
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x = torch.relu(self.l1(x))
        x = self.l2(x)
        return x

model = NeuralNet(input_size, hidden_size, output_size)
# Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
# Training loop
for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = torch.tensor(words, dtype=torch.float32)
        labels = torch.tensor(labels, dtype=torch.long)
        # Forward pass
        outputs = model(words)
        loss = criterion(outputs, labels)
        # Backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    if (epoch + 1) % 100 == 0:
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')
print(f"final loss, loss={loss.item():.4f}")
# Save model
data = {
    "model_state": model.state_dict(),
    "input_size": input_size,
    "hidden_size": hidden_size,
    "output_size": output_size,
    "all_words": all_words,
    "tags": tags
}
torch.save(data, "cse_chatbot.pth")
print("Model training complete. File saved as 'cse_chatbot.pth'")

```

3. cse_chatbot.py

cse_chatbot.py > ...

```python
import random
import json
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from nltk_utils import bag_of_words, tokenize
import numpy as np
import warnings
warnings.filterwarnings('ignore')
data = torch.load("cse_chatbot.pth", weights_only=True)
input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]

class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(NeuralNet, self).__init__()
        self.l1 = torch.nn.Linear(input_size, hidden_size)
        self.l2 = torch.nn.Linear(hidden_size, output_size)
    def forward(self, x):
        x = torch.relu(self.l1(x))
        x = self.l2(x)
        return x

model = NeuralNet(input_size, hidden_size, output_size)
model.load_state_dict(data["model_state"])
model.eval()
```

cse_chatbot.py > ...

```python

all_words = data["all_words"]
tags = data["tags"]
with open('cse_chatbot_dataset.json', 'r') as f:
    intents = json.load(f)

chatbot_name = "GEHU"
print("Hello! Type 'quit' to exit.")
while True:
    try:
        sentence = input('You: ')
        if sentence.lower() == "quit":
            print("Goodbye!")
            break
        tokenized = tokenize(sentence)
        bag = bag_of_words(tokenized, all_words)
        bag = bag.reshape(1, bag.shape[0])
        bag = torch.from_numpy(bag)
        output = model(bag)
        _, predicted = torch.max(output, dim=1)
        tag = tags[predicted.item()]
        probs = torch.softmax(output, dim=1)
        prob = probs[0][predicted.item()]
        if prob.item() > 0.75:
            for intent in intents["intents"]:
                if tag == intent["tag"]:
                    response = random.choice(intent['responses'])
                    print(f"{chatbot_name}: {response}")
        else:
            print(f"{chatbot_name}: I didn't understand that.")
    except KeyboardInterrupt:
        print("\nGoodbye!")
        break

```

REFERENCES

1. PyTorch Documentation Welcome to PyTorch Tutorials — PyTorch Tutorials 2.5.0+cu124 documentation
2. NLTK Documentation NLTK :: Natural Language Toolkit
3. Online Resources for Chatbot Development
4. Brownlee, Jason. “Deep Learning for Natural Language Processing.”
5. Raschka, Sebastian. “Python Machine Learning.”
6. Chollet, François. “Deep Learning with Python.”
7. Research Papers on Chatbots and NLP Techniques