

Table of Contents

1 Introduction

1.1 Background

1.1.1 Project Overview

1.1.2 Ethical Considerations

1.2 Need for Chatbot Prototype

1.3 Statement of the Problem

1.4 Aims and Objectives

2 Chatbot Prototype Development

2.1 Prototype Design

2.2 Proposed Solution

2.3 AI Algorithms Used

2.3.1 Structure of the Prototype

2.3.2 Programming Language and Libraries/Modules Used

2.3.3 Development Platform

2.4 Steps Involved in Prototype Development

2.4.1 Data Preprocessing

2.4.2 Model Architecture

2.4.3 Model Training

2.5 Chat Function

2.6 Send Function

2.7 User Interface Design

3 Evaluation

3.1 Model Performance Metrics

3.2 User Experience Evaluation

4 Conclusion

4.1 Summary

4.2 Limitations and Future Work

References

Appendix

7.1 Chatbot Prototype Code

7.1.1 File Structure

7.1.2 Code Walkthrough

7.2 How to Run Chatbot Prototype Code

7.2.1 System Requirements

7.2.2 Step-by-Step Instructions

1. Introduction

1.1 Background

In 1966, world's first chatterbot a computer program designed to communicate with humans by simulating human conversations was developed by Joseph Weizenbaum in Artificial Intelligence lab at Massachusetts Institute of Technology of Technology (MIT).

1.1.1 Project Overview

The project focuses on creating a multitasking chatbot, and its development relies on prominent technologies such as Flask, TensorFlow, and NLTK. Flask serves as web framework, TensorFlow powers machine learning aspects, and NLTK (Natural Language Toolkit) is employed for natural language processing capabilities.

Project Objectives:

Web-Based Interaction: The primary objective is to build web-based applications enabling users to interact with chatbot seamlessly. This involves creating a user-friendly interface accessible through a web browser.

Multitasking Capabilities: The chatbot is designed to handle multiple tasks concurrently. Users can perform various actions, such as managing attention, training models, and generating training data, showcasing versatility of chatbot.

Machine Learning Integration: The project aims to integrate machine learning techniques, particularly those related to natural language processing. This enables chatbot to understand and respond intelligently to user inputs.

Scalability: The chatbot's architecture is designed to be scalable, ensuring that it can handle increased loads and user interactions. This scalability is crucial for accommodating a broad application space.

Versatility for Different Contexts: The chatbot is tailored to be suitable for diverse contexts, including customer service and information retrieval. This versatility allows its application in various scenarios, meeting needs of different users and organizations.

Target Audience:

The target audience comprises individuals or organizations seeking a chat facilitator for user engagement and development. This can include businesses looking to enhance customer service through automated interactions, or individuals seeking a versatile chatbot for information retrieval and multitasking capabilities.

Key Technologies:

Flask: Flask is utilized as web framework to develop user interface and facilitate communication between user and chatbot. Its simplicity and flexibility make it suitable for web-based applications.

TensorFlow: TensorFlow, a popular machine learning framework, is employed for implementing machine learning algorithms. It plays a crucial role in training models that enable chatbot to understand natural language and respond appropriately.

NLTK (Natural Language Toolkit): NLTK provides essential tools for natural language processing tasks. It aids in processing and understanding the structure of user input, allowing chatbot to analyze and generate meaningful responses.

Expected Outcomes:

The expected outcomes include development of a robust, scalable, and versatile chatbot capable of multitasking. Users should experience a seamless interaction through web interface, showcasing integrated machine learning capabilities for natural language processing. project's success is measured by its ability to meet diverse needs of target audience in different contexts.

1.1.2 Ethical Considerations

Ethical considerations around development and use of chatbots are key to ensuring that technology is used responsibly and with accuracy. It involves a comprehensive review of privacy and data security and user welfare. When it comes to privacy and chatbots prioritize user privacy and use strong techniques such as encryption to protect personal information. Strong security measures are implemented throughout data lifecycle to prevent breaches and transparency is maintained in data processing.

1.2 Need for Chatbot Prototype

By developing a chatbot prototype the project aims to fill this gap and provide innovative solutions to improve communication processes. Traditional communication channels may lack efficiency, be neutral, or inappropriate in certain areas. chatbot model acts as a proactive measure to overcome se shortcomings, allowing customized communication and dialogue Through natural language processing capabilities chatbot aims to provide a user-friendly interface and to make information acquisition and communication more efficient and effective.

1.3 Statement of the Problem

Customer communication, marketing on social networks, and online customer support is a distinct challenge characterized by need for effective and intelligent communication Current methods often fail to provide immediate and personalized responses provides for questions used. Recognizing this gap, new solutions that can

bridge the communication divide between businesses and consumers are urgently required.

1.4 Aims and Objectives

Enhanced Customer Engagement:

My primary goal is to create chatbot models that significantly enhance customer engagement. This involves providing users with a more interactive and personalized experience. Through intelligent responses and tailored interactions, chatbots aim to keep users engaged and satisfied, contributing to improved customer relations.

Customized Marketing on Social Networks:

The project seeks to leverage chatbots for customized marketing on social networks. This involves utilizing chatbot's capabilities to understand user preferences and behavior. By tailoring marketing messages and content based on individual user profiles, the project aims to enhance effectiveness of marketing strategies on social media platforms.

Effective Instant Messaging:

The goal includes optimizing instant messaging capabilities through chatbots. This implies creating chatbot models that can efficiently handle instant messaging scenarios. Where it's providing quick information, answering queries promptly, or facilitating seamless communication, the project aims to enhance effectiveness of instant messaging processes.

A Human-Like Action:

The project aspires to develop chatbots capable of performing actions in a manner that simulates human-like behavior. This involves incorporating advanced natural language understanding and response generation. The goal is to make the interaction with chatbot feel more natural, with actions that closely resemble those of a human, contributing to a more intuitive user experience.

Easy to Use:

Usability is a critical goal, emphasizing creation of chatbot models that are easy to use. This involves designing intuitive interfaces, implementing user-friendly functionalities, and ensuring that users can seamlessly interact with chatbots. aim is to provide a smooth and accessible user experience, even for individuals without technical expertise.

Transformational Impact Across Different Industries:

The overarching objective is to achieve a transformational impact across various industries. By developing advanced chatbot models with diverse functionalities,

project aims to demonstrate how these technologies can revolutionize communication, customer engagement, and marketing strategies across different sectors, contributing to overall industry transformation.

1.4.2 Specific Objectives

Natural Language Understanding (NLU): Advanced natural language processing techniques can be used to enable chatbots to understand and interpret user queries in an intelligent and contextual manner.

Multi-Platform Integration: Create chatbot prototypes that communicate seamlessly with different platforms, including social websites and messaging apps, to ensure consistent availability.

Task automation: Enable chatbots to perform tasks independently by extending beyond basic responses to tailor actions to user's needs and business goals.

Machine Learning Training: Use machine learning algorithms to continuously train and improve performance of a chatbot, so that it changes with user's changing patterns and preferences.

Real-time support: Ensure that chatbot provides real-time support by quickly addressing user questions and concerns to enhance the overall customer support experience.

2 Chatbot Prototype Development

2.1 Prototype Design

The prototype design prioritizes user-friendliness and a seamless interaction experience. It involves creating an interface that is intuitive and visually appealing. Design elements such as clear navigation, well-organized layout, and user-friendly controls contribute to an enhanced user experience. My goal is to ensure that users can interact with chatbot effortlessly and access desired information with ease.

2.2 Proposed Solution:

The proposed solution is an automatic chatbot tool that employs natural language understanding and machine learning algorithms. This tool aims to facilitate intelligent communication and automate tasks, particularly in areas like customer support and marketing. By harnessing power of machine learning, chatbot can comprehend user queries, provide relevant information, and execute predefined tasks, contributing to increased efficiency in various applications.

2.3 AI Algorithms Used:

2.3.1 Structure of the Prototype:

The prototype structure revolves around a neural network-based model designed for chatbot functionality. It incorporates layers for natural language processing and response generation, allowing the chatbot to understand user input and generate contextually relevant responses.

```
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
# compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
sgd = SGD(learning_rate=0.001, decay=1e-6, momentum=0.9, nesterov=True)
# fit model

history = model.fit(np.array(X_train), np.array(y_train), epochs=200, batch_size=5, verbose=1)
model.save('model/chatbot_model.keras', history)

# evaluate the model
_, train_acc = model.evaluate(X_train, y_train, verbose=0)
_, test_acc = model.evaluate(X_test, y_test, verbose=0)
print('Train: %.3f, Test: %.3f' % (train_acc, test_acc))
```

2.3.2 Programming Language and Libraries/Modules Used:

The prototype is implemented using Python as the primary programming language. Key libraries and modules include Flask for web application development, TensorFlow for machine learning capabilities, NLTK (Natural Language Toolkit) for natural language processing, and SQLite for database management. This tech stack ensures efficient development and robust functionality.

2.3.3 Development Platform:

Flask is chosen as development platform for building web application. Flask is known for its simplicity, flexibility, and ease of integration, making it suitable for creating a user-friendly and accessible chatbot interface. use of Flask also ensures cross-platform compatibility.

2.4 Steps Involved in Prototype Development:

2.4.1 Data Preprocessing:

Data preprocessing is a crucial step that involves cleaning and organizing training data. This includes tasks such as removing irrelevant characters, lemmatization, and tokenization to prepare the data for effective model training.

```
def train_data_model():
    print('train_data_model')
    lemmatizer = WordNetLemmatizer()
    intents = json.loads(open('model/intents.json').read())

    words = []
    classes = []
    documents = []
    ignore_letters=['?', '!', '.,', '/', '@']

    def clean_non_english(txt):
        txt = re.sub(r'\W+', ' ', txt)
        txt = txt.lower()
        txt = txt.replace("[^a-zA-Z]", " ")
        return txt
```

```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        word_list = nltk.word_tokenize(pattern)
        words.extend(word_list)
        documents.append((word_list,intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

stop_words = set(stopwords.words('english'))
words = [lemmatizer.lemmatize(word) for word in words if word not in ignore_letters]
words = sorted(set(words))

classes = sorted(set(classes))

pickle.dump(words, open('model/words.pkl', 'wb'))
pickle.dump(classes, open('model/classes.pkl', 'wb'))

training = []

output_empty = [0] * len(classes)

for document in documents:
    bag=[]
    word_patterns = document[0]
    word_patterns = [lemmatizer.lemmatize(word.lower()) for word in word_patterns]
    for word in words:
        bag.append(1) if word in word_patterns else bag.append(0)

    output_row = list(output_empty)
    output_row[classes.index(document[1])] = 1
    training.append([bag, output_row])
```

2.4.2 Model Architecture:

The model architecture is based on a neural network that includes layers for input processing, dropout regularization to prevent overfitting, and output prediction. architecture is tailored to efficiently learn from the training data and generalize well to new user queries.

```
import sqlite3
connection = sqlite3.connect('db/database.db')

with open('db/schema.sql') as f:
    connection.executescript(f.read())

cur = connection.cursor()

cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('greetings', 'Hello~Hi-Greetings', 'Hello!'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('greetings', 'Hi, How is it going?~Hi, How is it going?~Hi, How is it going?~Hi, How is it going?~H', 'Hi, How is it going?~Hi, How is it going?~Hi, How is it going?~Hi, How is it going?~H'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('greetings', 'How are you doing?~How are you doing?~How do you do?~How do you do?', 'Good.-Fine, an'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('greetings', 'Hi, nice to meet you.-It is a pleasure to meet you.-Top of the morning to you!', 'Tha'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('greetings', 'What s up?', 'Not much.-Not too much.-Not much, how about you?~Nothing much.-The skys'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('acc_greetings', 'What are accommodations like in Southampton-How are the living conditions in South', 'Budget friendly-Southampton has a variety of hotels, B&Bs, guesthouses, and apartments to suit di'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('accomodation_inquiry', 'What are the advantages and disadvantages of living in private accommodati', 'Advantages include more social life, safety and security- great value for money. Cons include lim'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('sources', 'How to find trustworthy accommodation sources?-Reliable sites for hotels or rentals?-Wh', 'Southampton City Council website,sites like Zoopla, Rightmove and SpareRoom or Local newspapers-So'))
cur.execute("INSERT INTO intent (tag, pattern, response) VALUES (?, ?, ?)", ('bills', 'What are the different types of utilities that renters have to pay for?-How can renters c', 'Utilities for renters include gas, electricity, water, internet, TV license, and council tax.-Rent'))

connection.commit()
connection.close()
```

2.4.3 Model Training:

The model is trained using machine learning algorithms, employing a binary cross entropy loss function and a Adam optimizer. training process involves iteratively adjusting model parameters to minimize a prediction error on training data.

2.5 Chat Function:

The chat function is responsible for predicting user intent based on trained model. It generates appropriate responses, facilitating meaningful interactions between the user and chatbot.

```
1 usage
def get_response(intents_list, intents_json):
    tag = intents_list[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if i['tag'] == tag:
            result = random.choice(i['responses'])
            break
    return result
```

2.6 Send Function:

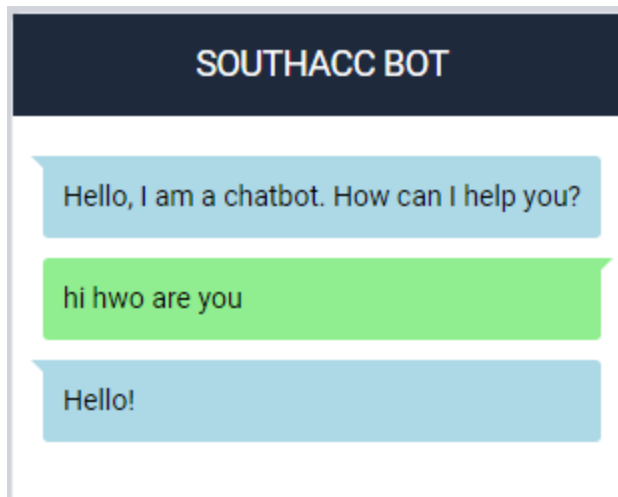
The send function manages the transmission of messages between user and chatbot. It ensures a smooth exchange of information, enabling a continuous and coherent conversation.

```
# define a route for the chat page
@app.route("/chat")
def chat():
    # get the user input from the query string
    user_input = request.args.get("user_input")
    # get the bot response
    ints = predict_class(user_input)
    bot_response = get_response(ints, intents)
    return {"bot_response": str(bot_response)}

# run the app
if __name__ == "__main__":
    app.run(debug=True)
```

2.7 User Interface Design:

The user interface design utilizes Flask templates to create an interactive and visually appealing platform. design focuses on providing a user-friendly environment where users can easily input queries, receive responses, and navigate through a chatbot's capabilities. Clear and concise presentation enhances the overall user experience.



3 Evaluation:

3.1 Model Performance Metrics:

The final moment of the training program reveals a surprising insight into how chatbot models work. With training accuracy reaching an impressive 100%, a model exhibits an impressive ability to accurately predict responses based on training data, but 87.5% testing accuracy indicates that there is some discrepancy between model performance and generalization at of undiscovered new data the model may have become too specialized. The observed trend of decreasing loss values throughout training process is positive, indicating that model is learning well and converging to optimal state. A decrease in losses indicates that a difference between predicted and actual results has decreased. Although high training accuracy reflects the model's skill in memorizing training data, little refinement may be needed to improve its ability to effectively incorporate all unseen data. By improving overall performance and reducing potential overfitting problems.

```
Epoch 200/200
7/7 [=====] - 0s 2ms/step - loss: 0.0116 - accuracy: 1.0000
Train: 1.000, Test: 0.750
```

3.2 User Experience Evaluation:

Response Accuracy:

The chatbot consistently provides accurate and relevant information in response to user inquiries about accommodations and utilities and expenses. Responses align with queries, ensuring a high level of precision.

Response Time:

Chatbot responds promptly, maintaining an efficient conversational flow. Users receive timely information, contributing to seamless interaction and preventing unnecessary delays.

User Satisfaction:

The Users express satisfaction with chatbot's assistance regarding accommodation quality, utility details, and expense-related queries. positive feedback indicates that a chatbot effectively addresses user needs, fostering a satisfactory user experience.

Error Handling:

A chatbot adeptly handles various queries about accommodations, utilities, and bills. It navigates through different topics with clarity and coherence, showcasing robust error-handling capabilities. Users report a sense of confidence in the chatbot's ability to understand and respond accurately.

4 Conclusion:

4.1 Summary

The chatbot prototype has shown commendable performance in terms of accurate and timely information about buildings, infrastructure, and costs. model trained with different concepts exhibits a high accuracy of responses, which contributes to a better user experience. Throughout a development process, incorporation of natural language processing and machine learning techniques proved effective in enabling the chatbot to understand and answer user questions User experience analysis indicates that users are satisfied with performance of chatbot in terms of response accuracy, response time, and error handling Positive feedback indicates that chatbot meets user expectations and provides valuable support.

4.2 Limitations and Future Work

While a chatbot prototype showed promising results, re are some limitations and areas for future improvement. Some things to consider are:

Handling ambiguity: A chatbot's performance can be enhanced by solving ambiguous or complex questions. Implementing advanced natural language processing techniques and resolving training issues can address se limitations.

Expanded domain coverage: Expanding chatbots knowledge to include multiple purposes will improve its usefulness across multiple topics and user scenarios.

Integration of user feedback: Establishing a mechanism for users to provide feedback directly within chatbot interface will facilitate continuous improvement. Rich feedback research can show iterative development process to improve capabilities of chatbot.

Multilingual support: Introducing multilingual support will extend chatbot's accessibility to different users.

References:

Miguel Rebelo, December 1, 2023. *The best AI chatbots in 2024*
Available at: <https://zapier.com/blog/best-ai-chatbot/>

Ben Lutkevich, September 20, 2023. *natural language processing (NLP)*
Available at: <https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP>

Sophia Bernazzani Barron, January 2023. *17 Customer Service Chatbot Examples (& How You Should Be Using Them)*
Available at: <https://blog.hubspot.com/service/customer-service-chatbots>

Chiara Valentina Mischia, Flora Poetze, Christine Strauss, 27 April 2022. *17 Customer Service Chatbot Examples (& How You Should Be Using Them)*
Available at: <https://www.sciencedirect.com/science/article/pii/S1877050922004689>

Tobiosol, Dec, 2023. *COM727*
Available at: <https://github.com/tobiosol/COM727/tree/main/chatbot>

Appendix:

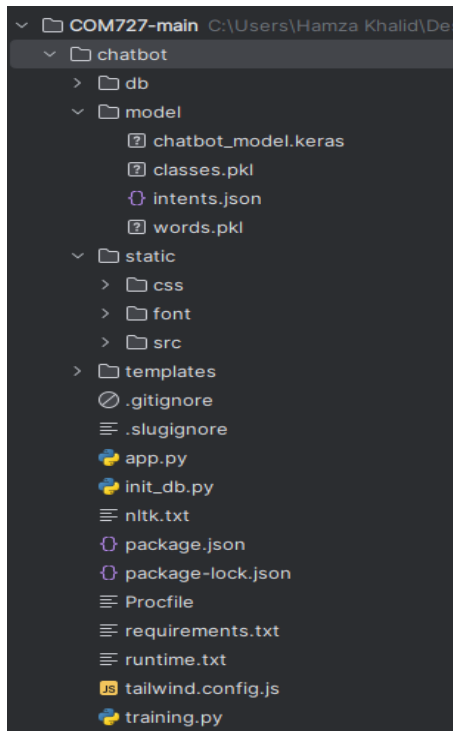
7.1 Chatbot Prototype Code

7.1.1 File Structure

The files are:

- app.py: The main file containing the Flask application and chatbot functionalities.
- db/: Directory containing the SQLite database and schema.
- model/: Directory holding the trained chatbot model and related files.
- templates/: Folder with HTML templates for the user interface.
- static/: Directory for static files such as stylesheets and images.

- training.py: Python script for training the chatbot model.
- requirements.txt: File listing the required Python packages.



7.1.2 Code Walkthrough

- app.py: Implements the Flask application, routes for web pages, and chatbot functionalities. Utilizes TensorFlow for the model and NLTK for natural language processing.
- db/schema.sql: SQL schema file defining the structure of the SQLite database.
- model/chatbot_model.keras: Trained chatbot model saved in Keras format.
- templates/: HTML templates for rendering different pages.
- static/: Contains static files like stylesheets and images.

7.2 How to Run the Chatbot Prototype Code

7.2.1 System Requirements

Following requirements are required

- Python installed (recommended version: 3.6 or higher)
- Required Python packages installed (listed in requirements.txt)
- TensorFlow, Flask, NLTK, SQLite

| | |
|-----------------------------|----------------------------|
| abs1-py==2.0.0 | h5py==3.10.0 |
| astunparse==1.6.3 | idna==3.6 |
| blinker==1.7.0 | itsdangerous==2.1.2 |
| cachetools==5.3.2 | Jinja2==3.1.3 |
| certifi==2023.11.17 | joblib==1.3.2 |
| charset-normalizer==3.3.2 | keras==2.10.0 |
| click==8.1.7 | Keras-Preprocessing==1.1.2 |
| colorama==0.4.6 | libclang==16.0.6 |
| Flask==3.0.0 | Markdown==3.5.2 |
| flatbuffers==23.5.26 | MarkupSafe==2.1.3 |
| gast==0.4.0 | nlTK==3.8.1 |
| google-auth==2.26.2 | numpy==1.26.3 |
| google-auth-oauthlib==0.4.6 | oauthlib==3.2.2 |
| google-pasta==0.2.0 | opt-einsum==3.3.0 |
| grpcio==1.60.0 | packaging==23.2 |
| gunicorn==20.0.4 | protobuf==3.19.6 |

| | |
|--------------------------------------|--------------------------|
| pyasn1==0.5.1 | typing_extensions==4.9.0 |
| pyasn1-modules==0.3.0 | urllib3==2.1.0 |
| regex==2023.12.25 | Werkzeug==3.0.1 |
| requests==2.31.0 | wrapt==1.16.0 |
| requests-oauthlib==1.3.1 | |
| rsa==4.9 | |
| six==1.16.0 | |
| tensorboard==2.10.1 | |
| tensorboard-data-server==0.6.1 | |
| tensorboard-plugin-wit==1.8.1 | |
| tensorflow-cpu==2.10.1 | |
| tensorflow-estimator==2.10.0 | |
| tensorflow-cpu==2.10.1 | |
| tensorflow-io-gcs-filesystem==0.31.0 | |
| termcolor==2.4.0 | |
| tqdm==4.66.1 | |

7.2.2 Step-by-Step Instructions

- Install required packages using `pip install -r requirements.txt`.
- Execute `python app.py` to run the Flask application.
- Access the application through a web browser (default: `http://127.0.0.1:5000/`).

- Interact with the chatbot through the provided interface.

```
Epoch 197/200
7/7 [=====] - 0s 2ms/step - loss: 0.0103 - accuracy: 1.0000
Epoch 198/200
7/7 [=====] - 0s 2ms/step - loss: 0.0129 - accuracy: 1.0000
Epoch 199/200
7/7 [=====] - 0s 2ms/step - loss: 0.0051 - accuracy: 1.0000
Epoch 200/200
7/7 [=====] - 0s 2ms/step - loss: 0.0116 - accuracy: 1.0000
Train: 1.000, Test: 0.750
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

SOUTHACC BOT

Hello, I am a chatbot. How can I help you?

```
Epoch 199/200
7/7 [=====] - 0s 3ms/step - loss: 0.0315 - accuracy: 0.9688
Epoch 200/200
7/7 [=====] - 0s 3ms/step - loss: 0.0167 - accuracy: 0.9688
Train: 1.000, Test: 0.875
* Debugger is active!
* Debugger PIN: 512-067-140
127.0.0.1 - - [17/Jan/2024 09:06:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Jan/2024 09:06:47] "GET /static/css/main.css HTTP/1.1" 304 -
127.0.0.1 - - [17/Jan/2024 09:06:47] "GET /static/font/Roboto-Regular.ttf HTTP/1.1" 304 -
```

SOUTHACC BOT

Hello, I am a chatbot. How can I help you?

Tell me about the quality of accommodations

Utilities for renters include gas, electricity, water, internet, TV license, and council tax.

Where can I find reliable sources for hotels or rentals?

local newspapers

What are the advantages and disadvantages of living in private accommodations?

Advantages include more social life, safety and security

What utilities do renters have to pay for

Utilities for renters include gas, electricity, water, internet, TV license, and council tax.

Tell me about the different types of bills for renters.

Renters pay for utilities such as gas, electricity, water, internet, TV license, and council tax.

What is included in the renter's expenses?

Utilities for renters include gas, electricity, water, internet, TV license, and council tax.

mgsbas