E-COMMERCE CHATBOT

USING LSTM

CHANDRASEKARAN P

422521104008

chandrasekaran4096@gmail.com

University college of Engineering
Villupuram

OUTLINE

1. PROBLEM STATEMENT
2. PROPOSED SYSTEM/SOLUTION
3. SYSTEM REQUIREMENT
4. ALGORITHM AND DEPLOYMENT
5. WHO ARE THE END-USERS?
6. RESULT
7. CONCLUSION
8. REFERENCES

PROBLEM STATEMENT

The problem in eCommerce customer support is high response times and operational inefficiencies. We implemented an automated chatbot using deep learning techniques to provide real-time, consistent, and cost-effective customer support. This solution aims to enhance user experience and streamline the support process.

PROPOSED SYSTEM/SOLUTION

CHATBOT ARCHITECTURE DESIGN:

Design a chatbot architecture using LSTM networks for natural language processing, complemented by text cleaning, tokenization, and intent classification to deliver precise and relevant responses.

DEEP LEARNING MODEL:

Utilizing LSTM networks to process and understand the sequential nature of natural language data effectively. This involves processing textual data and capturing long-term dependencies in sequences, facilitating the model's understanding and classification of user queries.

TEXT PREPROCESSING:

This involves removing unnecessary characters, converting text to lowercase, lemmatizing words, and removing stopwords to prepare the text data for model input.

HYPERPARAMETER TUNING:

Utilize Keras Tuner to automate the optimization of model parameters to improve performance and accuracy. This involves systematically exploring the hyperparameter space to find the optimal configuration for the model.

TOKENIZATION AND PADDING:

Implement tokenization and padding techniques to convert text data into numerical sequences and ensure a consistent input length for the model. This involves tokenizing textual data to convert words into numerical form and applying padding to ensure a uniform sequence length.

SYSTEM APPROACH

HARDWARE

1. CPU: A multicore processor to handle computational tasks efficiently, supporting the intensive processing demands of deep learning algorithms

2. RAM: Minimum of 8GB RAM to ensure smooth performance during model training and inference, accommodating the memory requirements of large datasets and complex computations.

3. Internet Connection: Stable and high-speed internet connection to facilitate data retrieval, model updates, and seamless interaction with external services and APIs.

SYSTEM APPROACH

SOFTWARE

1. Python: Programming language used for developing the chatbot application.
2. TensorFlow/Keras: Deep learning libraries used for building and training the LSTM model.
3. NLTK: Natural Language Toolkit used for text preprocessing and tokenization.
4. scikit-learn: Library for various machine learning tasks like data preprocessing and model evaluation.
5. Jupyter Notebook: Interactive computing environment used for prototyping, data analysis, and model development.
6. Optional: Django for web-based interface (if applicable)

ALGORITHM AND DEPLOYMENT

1.DATA PREPARATION:

- Retrieved the dataset from Kaggle to serve as the basis for training and testing the chatbot model

2. TEXT PREPROCESSING:

- Cleaned and organized raw text data using NLTK for text processing.
- Removed punctuation and stopwords to ensure clean input.
- Lemmatized words to reduce them to their base form
- Tokenized text to convert sentences into a list of words for analysis

ALGORITHM AND DEPLOYMENT

3.DEEP LEARNING MODEL: NETWORKS

- Implemented LSTM networks to process and understand natural language data
- Employed a dense layer with 208 units followed by a softmax output layer
- Utilized the Adam optimizer with a learning rate set to 0.01
- Configured the model with two LSTM layers, each having 110 units

4.HYPERPARAMETER TUNING: KERAS TUNER

- Utilized Keras Tuner to automate the optimization of model parameters
- Tuned the number of units in LSTM layers ranging from 50 to 150.
- Explored the number of dense layers in the model from 1 to 20.

ALGORITHM AND DEPLOYMENT

- Adjusted learning rates between 0.01, 0.001, and 0.0001 for optimal performance.

5. TOKENIZATION AND PADDING:

- Tokenized the text data using Keras Tokenizer to convert words into numerical sequences.
- Applied padding to ensure a uniform sequence length of 200 for model input.
- Set the vocabulary size to 2000 to capture the most frequent words in the dataset.
- Used an out-of-vocabulary token (OOV) to handle words not in the vocabulary during tokenization.

DEPLOYMENT

The chatbot code is hosted on GitHub by implementing the main code directly on a Jupyter Notebook. This approach allows for easy viewing and execution of the code.

1. Jupyter Notebook:

2. Django Web Application:

The chatbot is presented as a user-friendly web-based interface using the Django framework and hosted on GitHub. This method offers a seamless user experience and makes the chatbot accessible via a web browser.

WHO ARE THE END USERS?

Online Shoppers: Individuals looking for product information, recommendations, or assistance with their shopping experience

Customer Support Teams: Staff members responsible for handling customer queries and providing timely and accurate responses.

Website Visitors: Potential customers exploring the eCommerce website whomay have questions or require assistance.

Technical Support: IT professionals or developers involved in maintaining and updating the chatbot's functionality and performance

RESULT

jupyter chatbot Last Checkpoint: 4 days ago

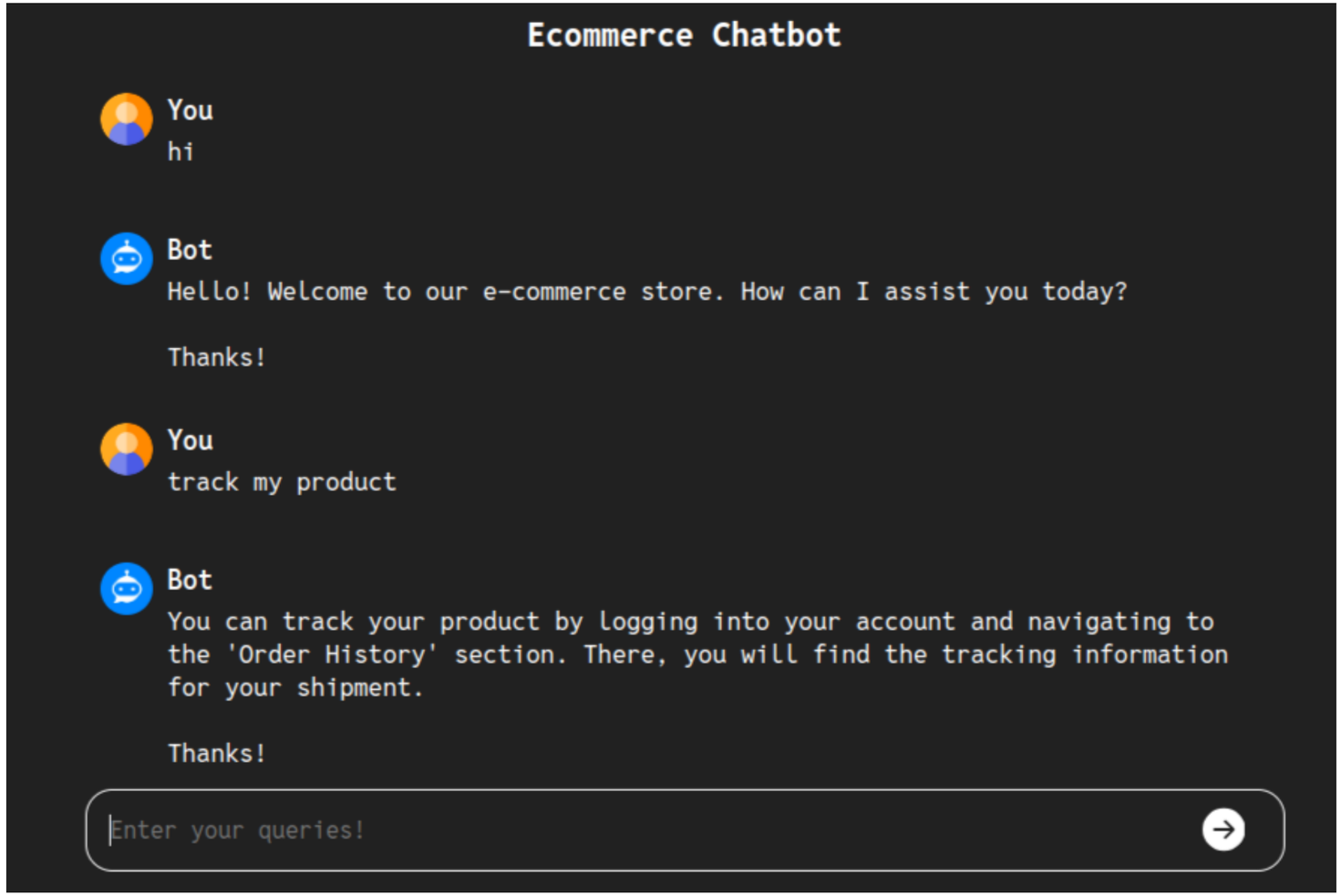
File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
[nltk_data]   Package wordnet is already up-to-date!
You:  hi
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you trai
n or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Bot: Hello! Welcome to our e-commerce store. How can I assist you today? (Score: 0.9859431 )
You:  track my product
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you trai
n or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 447ms/step
Bot: You can track your product by logging into your account and navigating to the 'Order History' section. There, you will find the tracki
ng information for your shipment. (Score: 0.9072466 )
You:  how to leave product review
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you trai
n or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 463ms/step
Bot: To leave a product review, navigate to the product page on our website and click on the 'Write a Review' button. You can share your fe
edback and rating based on your experience with the product. (Score: 0.9918269 )
You:  what is our return policy
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you trai
n or evaluate the model.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 477ms/step
Bot: Our return policy allows you to return products within 30 days of purchase for a full refund, provided they are in their original cond
ition and packaging. Please refer to our Returns page for detailed instructions. (Score: 0.9582239 )
You:  bye
Bot: Goodbye!
```

Executed on jupyter

RESULT



Implemented on Django

CONCLUSION

The chatbot code is hosted on GitHub by implementing the main code directly on a Jupyter Notebook. This approach allows for easy viewing and execution of the code.Our eCommerce chatbot leverages advanced AI to deliver personalized customer support, enhancing the shopping experience and providing a competitive edge in the digital marketplace. As AI technologies continue to advance, our chatbot is poised to evolve and offer even more innovative solutions to meet the changing demands of the eCommerce industry.

REFERENCES

- **NumPy:**
  https://numpy.org/
- **TensorFlow:**
  https://www.tensorflow.org/
- **Keras:**
  https://keras.io
- **scikit-learn (sklearn):**
  https://scikit-learn.org/stable/
- **Keras Tuner**
  https://keras.io/keras_tuner/
- **Django:**
  https://www.djangoproject.com