VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : **B.Tech(CSE- AIR)**

Course Title : **Fundamentals of Artificial Intelligence**

Course Code : **CSE2039**

Slot : **F1**

Title: Task Scheduler Bot

Team Members:

SWAATHY K | 20BRS1083
RAMYA T | 20BRS1109
JEYAHARI RAJASEKAR | 20BRS1211
DEVIKA RAJASEKAR | 20BRS1214

Faculty: Dr. R. Jothi

Sign:

Date: 27/7/22

TASK SCHEDULER CHATBOT

Swaathy K 20BRS1083, Jeyahari Rajasekar 20BRS1211,
Devika Rajasekar 20BRS1214, Ramya T 20BRS1109

School of computer science and engineering, Vellore Institute of Technology

Vellore Institute of Technology,
Vandalur – Kelambakkam Road Chennai,
Tamil Nadu – 600 127

ramya.2020a@vitstudent.ac.in

jeyahari.rajasekar2020@vitstudent.ac.in

swaathy.k2020@vitstudent.ac.in

devika.rajasekar2020@vitstudent.ac.in

Abstract

A chatbot is frequently considered as one of the most efficient strategies for artificial intelligence-based communication between humans and machines. It is a software programme that uses natural language processing (NLP) and deep learning techniques to conduct an online chat conversation via text. In the form of a GUI, it offers direct communication with a live human agent. By asking a few questions, our AI chatbot learns about the event details that the user wants to add to their Google Calendar. It analyses the user's request to schedule an event and retrieves the necessary database entries. The deep learning techniques employed in this chatbot are responsible for effectively comprehending the user intents and avoiding any errors. The chatbot responds to the user's query request with the most relevant response once the user's intent has been determined. The user is then given all of the information regarding the event that has been made so that they can be prepared for it and reminded of it. Our chatbot is written in Python using the Keras library and the Tkinter graphical user interface.

Keywords—artificial intelligence; chatbot; natural language processing; deep learning; Keras; GUI; Tkinter

I. Introduction

Time management is one of the crucial parts of an individual’s life and it is also an aspect that a lot of people struggle with. Whether it’s in a professional’s life or a student’s life, staying ahead of deadlines and organizing tasks can have a massive positive impact on one's productivity. Some also struggle with remembering events and preparing for the same. To aid us in overcoming all of these shortcomings, a task scheduling AI chat bot can be very useful. A chatbot is an artificially intelligent software agent that is capable of communicating and performing actions similar to a human. Chatbots are used in various fields and for various purposes such as for customer interaction, marketing on social network sites and instantly replying to the client etc. The task scheduler chat bot, named ‘Dewyt’, lets the user schedule tasks in their google calendar and stay on track with their tasks. One can simply text Dewyt whenever they want to schedule a task or just remembered an important deadline. The user needs to let Dewyt know about the specifics of the task such as the duration, the start and the end, the date it is scheduled on etc. Once the specifications are given the task will be automatically synced to the google calendar by Dewyt. In case of a clash of events or an overlapping of tasks, Dewyt lets the user know about the existing task and suggests rescheduling the task. Dewyt bot is developed with the aim to create a productivity tool that can be used by anyone who needs help with organizing and managing their time in the best way possible.

II. Methodology

Natural language processing (NLP) combines natural language comprehension (NLU) with natural language generation (NLG) (N. NLP is one of the categories of AI that allows chatbots to interpret and respond to user messages. AI is the science of making machines do human-like tasks. When it comes to creating chatbots, natural language processing (NLP) is quite crucial.

Natural Language Toolkit is primarily designed for constructing and using symbolic and statistical natural language processing in Python. It is a collection of libraries that includes several functions that we found useful, such as tokenization, parsing, text classification, stemming of words, and reasoning of semantics.

In Python, we utilised Keras, a neural network API library. We chose it because it works well with a variety of technologies, including R, PlaidML, TensorFlow, and Microsoft Cognitive Toolkit. Future advances would be simple to produce.

We made use of Tkinter to develop our GUI (Graphical User Interface).

Tensorflow is a machine learning software library framework that leverages dataflow graphs and differentiates programming over a variety of tasks to develop models. We employed it for many of its applications, including classification, understanding, prediction, and creation, because it can be used to generate large-scale development applications, including neural networks.

The RNN (Recurrent Neural Network) algorithm we used to develop this chatbot is called "LSTM," and it is used to identify the category to which the user's message belongs, and it will deliver an arbitrary response from a record of responses using text classification.

"Long Short-Term Memory," one of the most powerful artificial neural networks and a well-known subset of RNN, is designed to recognise patterns in sequences that are present in numerical time series data emerging from government agencies, stock markets, as well as text, handwriting, and spoken word.

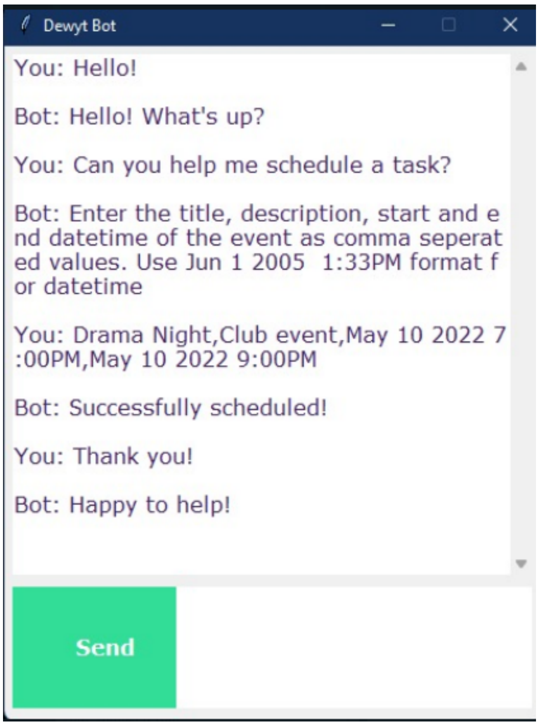
III. Experimental Analysis

First to begin the interaction with the bot, the user starts by entering any input that is considered as a greeting by the bot, such as ‘Hello’,’Good day’, etc, which is then met with a greeting. This is facilitated by the use of intents.json, which contains the possible input that might be entered by the user, grouped by categories. In the next reply the bot looks for keywords such as ‘schedule’ and ‘task’. If found, the bot replies with the input format that it wants the user to enter the task that is to be scheduled. Once the data for the task to be scheduled is entered, if there are no errors in the input and if there are no clashes with existing tasks, the bot immediately schedules the task in google calendar, through the google calendar API. The task reminder is then available in google calendar. If there happens to be a time clash between the task to be scheduled and an existing task that is to be completed, the bot requests the user to reschedule the task outside the time frame of the existing task with which it clashes. Once the user re enters the data for the task, the task is then scheduled and available in google calendar.

IV. Results with Screenshots

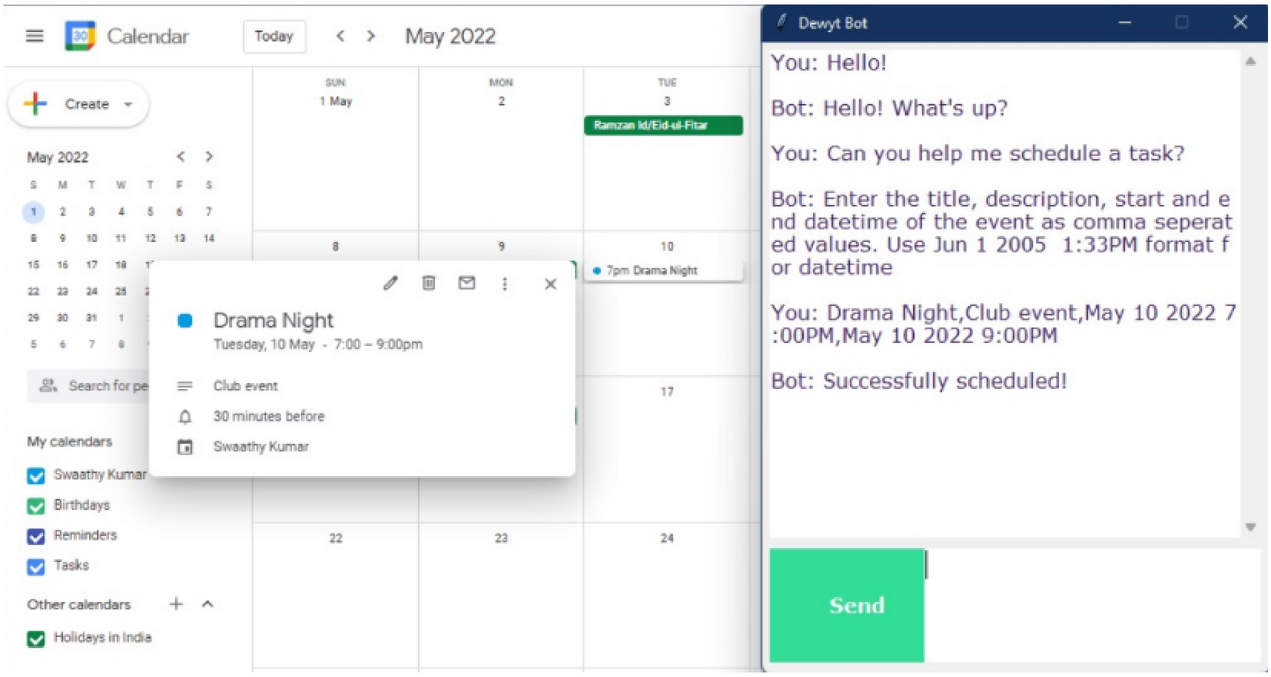
Source code : Github link

Scheduling a task:



```
You: Hello!

Bot: Hello! What's up?

You: Can you help me schedule a task?

Bot: Enter the title, description, start and e
nd datetime of the event as comma seperat
ed values. Use Jun 1 2005  1:33PM format f
or datetime

You: Drama Night,Club event,May 10 2022 7
:00PM,May 10 2022 9:00PM

Bot: Successfully scheduled!

You: Thank you!

Bot: Happy to help!
```

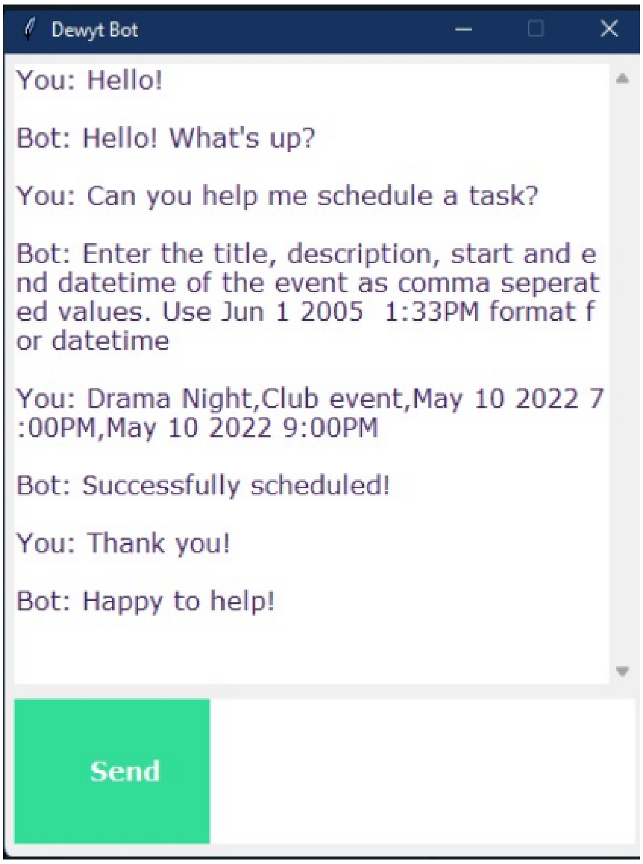
Task scheduled in Google Calendar:



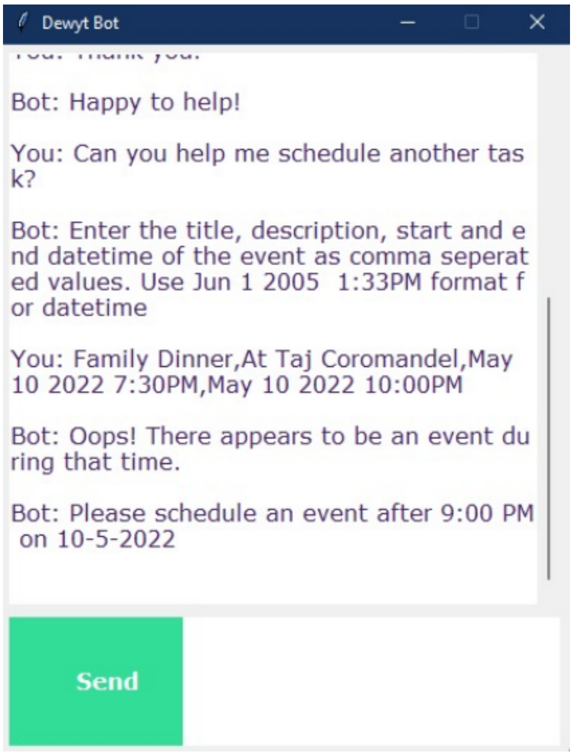
Drama Night
Tuesday, 10 May · 7:00 – 9:00pm
Club event
30 minutes before
Swaathy Kumar

```
You: Hello!

Bot: Hello! What's up?

You: Can you help me schedule a task?

Bot: Enter the title, description, start and e
nd datetime of the event as comma seperat
ed values. Use Jun 1 2005  1:33PM format f
or datetime

You: Drama Night,Club event,May 10 2022 7
:00PM,May 10 2022 9:00PM

Bot: Successfully scheduled!
```

Clashing of two tasks:

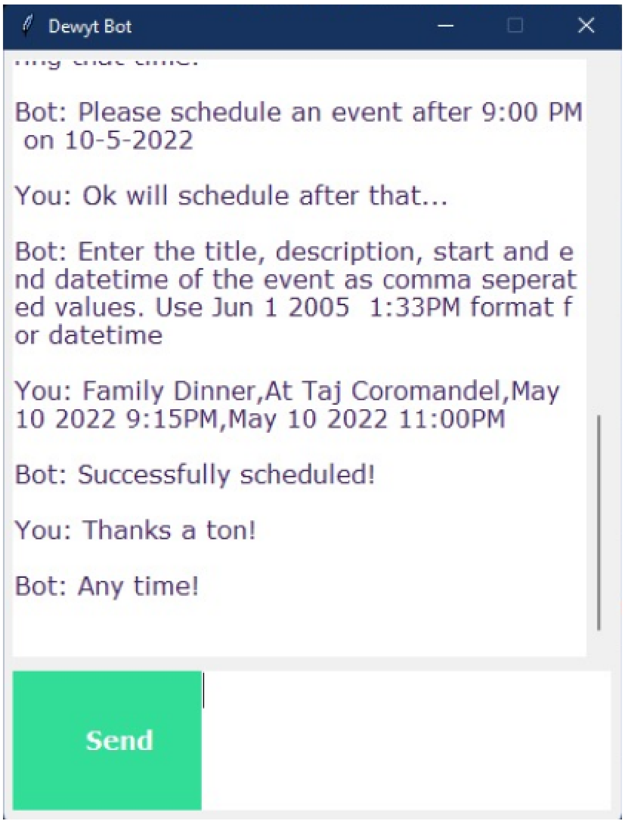


```
You: Hello!

Bot: Hello! What's up?

You: Can you help me schedule a task?

Bot: Enter the title, description, start and e
nd datetime of the event as comma seperat
ed values. Use Jun 1 2005  1:33PM format f
or datetime

You: Drama Night,Club event,May 10 2022 7
:00PM,May 10 2022 9:00PM

Bot: Successfully scheduled!

You: Thank you!

Bot: Happy to help!
```



```
You: Thank you!

Bot: Happy to help!

You: Can you help me schedule another tas
k?

Bot: Enter the title, description, start and e
nd datetime of the event as comma seperat
ed values. Use Jun 1 2005  1:33PM format f
or datetime

You: Family Dinner,At Taj Coromandel,May
10 2022 7:30PM,May 10 2022 10:00PM

Bot: Oops! There appears to be an event du
ring that time.

Bot: Please schedule an event after 9:00 PM
 on 10-5-2022
```

Rescheduling clashing task:



Bot: Please schedule an event after 9:00 PM on 10-5-2022

You: Ok will schedule after that...

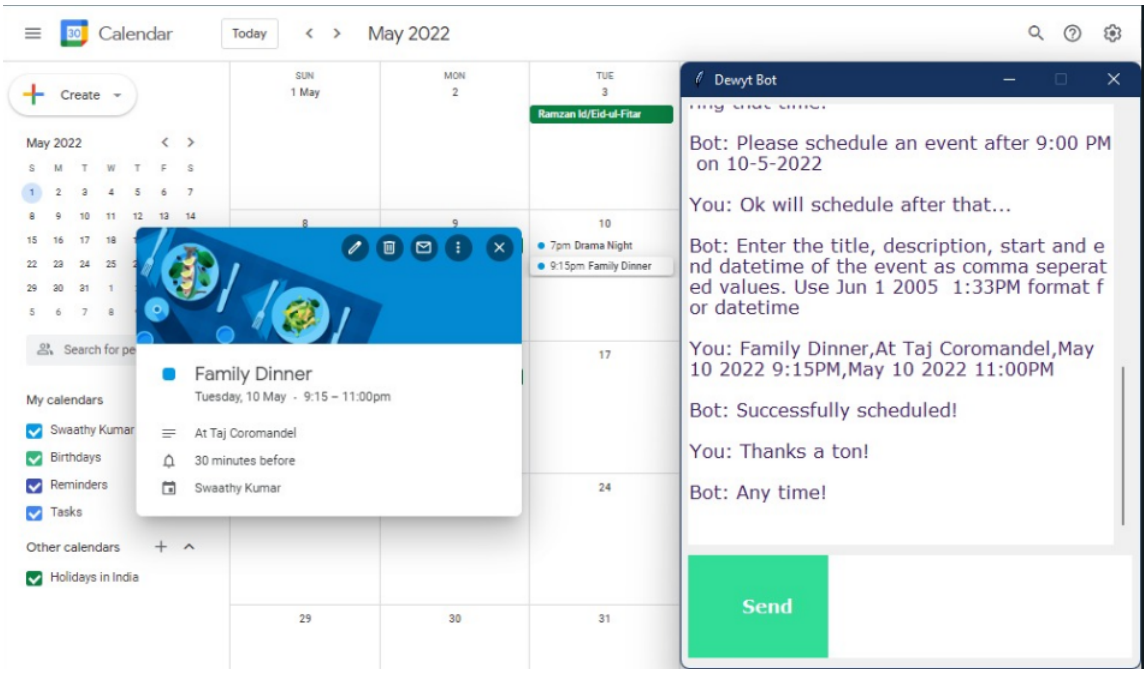
Bot: Enter the title, description, start and end datetime of the event as comma seperated values. Use Jun 1 2005 1:33PM format for datetime

You: Family Dinner,At Taj Coromandel,May 10 2022 9:15PM,May 10 2022 11:00PM

Bot: Successfully scheduled!

You: Thanks a ton!

Bot: Any time!



Family Dinner

Tuesday, 10 May · 9:15 – 11:00pm

At Taj Coromandel

30 minutes before

Swaathy Kumar

Bot: Please schedule an event after 9:00 PM on 10-5-2022

You: Ok will schedule after that...

Bot: Enter the title, description, start and end datetime of the event as comma seperated values. Use Jun 1 2005 1:33PM format for datetime

You: Family Dinner,At Taj Coromandel,May 10 2022 9:15PM,May 10 2022 11:00PM

Bot: Successfully scheduled!

You: Thanks a ton!

Bot: Any time!

V. Description Of Results & Observations

With the help of deep learning techniques such as Long short-term memory(LSTM) we were able to build this chatbot. The data collected in the form of intents, patterns and responses are used to train the chatbot. The libraries that are imported to create customer service chatbot are NLTK, TensorFlow, Keras, tflearn.

1. Intents.json – Contains a collection of tags with defined patterns and designated responses. This data is used as the training model by the chatbot.

2. train_chatbot.py – This file consists of a script that is used in building the model. By using deep learning techniques such as LSTM, it trains the model to categorise and identify the user's request.

3. words.pkl – Pickle file in which all distinct words are stored.

4. Classes.pkl – All the tag names that are used to classify when predicting the message are categorised and stored in this pickle file.

5. Chatbot_model.h5 – This model is trained which comprises a hierarchical data format and has stored weights and the architecture of our training model.

6. Chatgui.py – Script file that consists of the Graphical user interface of the chatbot.

Chatbot building process:

1. Importing the required libraries :

   First we open a new .py file called train_chatbot and import all the required modules and packages. We also imported the JSON package, since this package has parsing functionalities and will enable us to read JSON files.

2. Preprocessing of the data :

   Since we cannot directly use raw data, we used certain pre-processing techniques to make it easier to work with and also easier for the bot to understand. First , with the help of tokenization techniques we break the sentences into words. We use this in the intents file where we tokenize individually and store lists of patterns and responses.

The tags are placed in a list of classes to add all the intents associated with patterns. Then, we lemmatize each word, i.e , convert each word to their base form and remove the duplicates. This will save us the trouble of saving each form of a word. We then store these words in words.pkl, which contains the vocabulary that our bot will use, and store classification entities in classes.pkl, using pickle.dump().

3. Create training and testing data :

   We create training data which takes input and gives appropriate output responses.

4. Training the model :

   Our deep learning models architecture consists of 3 dense layers. The Keras sequential API is used for this. The first layer consists of 128 neurons, the second layer has 64 and the last layer has the same neurons as that of the number of classes. The remaining dropout layers are introduced to reduce the over-fitting of the model. We have used the SGD optimizer and fit the data to start the training process of the model. After the training of 200 epochs is finished, the model's accuracy reached 100%, then we saved the trained model using Keras model.save ("chatbot_model.h5") function

5. Interacting with the chatbot :

   Thus our bot is completed and is now ready to interact with users and schedule tasks. We have created pop up windows in “chatgui.py”, where we also extract the message entered by the user. This message is then received and goes through the various pre-processing techniques before it is inputted into the trained model. The model takes this input and predicts the tag of the message, based on which it will select an appropriate response from the list of available responses in the intents file.

VI. Conclusion

The developed Dewyt bot is therefore capable of intelligently understanding the user's query and processing it using NLP to give the desired result such as scheduling a task. The bot then uses the information it has obtained from the user to connect to the user's google calendar using the calendar's api. After the connection is made it schedules the given task in the mentioned deadline. The user is alerted if the bot discovers that the event clashes with an already existing event and in that case it suggests the user to reschedule it after a specific time duration.

VII. References

1. Sheikh, S. A., Tiwari, V., & Singhal, S. (2019, September). Generative model chatbot for human resources using deep learning. In 2019 International Conference on Data Science and Engineering (ICDSE) (pp. 126-132). IEEE.

2. Deepti Chopra, Jacob Perkins, and Nitin Hardeniya, "Natural Language Processing: Python and NLTK", pp.21–33,2016.

3. Mnasri, M. (2019). Recent advances in conversational NLP: Towards the standardization of Chatbot building. arXiv preprint arXiv:1903.09025.

4. C. Manning and H. Schutze, Foundations of Statistical Natural Language Processing, MIT press, 1999

5. T. Lalwani, S. Bhalotia, A. Pal, S. Bisen, and V. Rathod, "Implementation of a Chat Bot system using AI and NLP," International Journal of Innovative Research in Computer Science & Technology (IJIRCST), vol. 6, no. 3, pp. 26–30, 2018.

6. API documentation : Tensorflow core v2.8.0. TensorFlow. (n.d.). Retrieved March 15, 2022, from https://www.tensorflow.org/api_docs

7. NLTK. (n.d.). Retrieved March 16, 2022, from https://www.nltk.org/