iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 8 **Issue:** IV **Month of publication:** April 2020

DOI: http://doi.org/10.22214/ijraset.2020.4027

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

AI based Conversational Bot to Assist Users in Planning and Achieving Academic Goals

Vinayak Iyer[1], Sahil Sheth[2], Harsh Shelar[3], Radha Shankarmani[4]
[1, 2, 3, 4]*Sardar Patel Institute of Technology, Mumbai, Maharashtra, India.*

Abstract: ***The development of conversational bots have increased, while in many cases its overall intention still remains loosely defined. Due to the innovative and relatively new technology, there is an opportunity to create meaningful experiences with conversational bots in a human's life. In this paper, we describe a task-oriented conversational bot aiming to provide a frictionless, and non-input intensive interface to assist the user in planning academic tasks by creating and generating academic schedules, and reminders.***

We developed a Flutter application that acts as the front end to the bot. The intent classification and slot detection is implemented using Dialog Flow and a GRU network. We consider the sentiment and confidence scores to predict user's intents using a content based recommendation system.

The schedule containing tasks is generated using a modified longest-task-first algorithm. We also capitalise the use of Plug and Play Language Model [3] (PPLM) for text prediction while the user types the input string. The recurrent neural network and Dialogflow communicate with the Flutter application through REST APIs.

KeyWords: Conversational bot, Intent Detection, Recurrent Neural Network, Dialog Flow, Schedule generation

I. INTRODUCTION

Day-to-day planning in order to achieve academic milestone(s) is an intensive task without assistance. Presently, there is a lack of dedicated and personalised tools to motivate and keep track of milestones, tailored according to the user's needs. Given the increasing popularity and prevalence of conversation systems [11], a tool helping in charting out a planner and help the user to achieve the targets can prove to be of great assistance.

The challenges in developing such a tool include building an accurate user intent and slot detection model, an accurate user profile to generate and recommend more relatable prompts to the user, an algorithm which schedules tasks preferred by the user, on top of a user friendly front end.

In this paper, we present a conversational bot implemented as a mobile application that learns about the user using a feedback mechanism, and assists users in developing and charting out a planner to enable them to achieve the targets. We propose a non-traditional task oriented conversation system using neural networks.

Incorporating a Plug and Play Language Model (PPLM) [3] in the conversational bot is crucial to make the interaction with the user frictionless and as natural as possible. The PPLM employs a pre-trained language model and with changes to the model parameters it can generate text with controlled attributes such as topic and sentiment. The user sentiment can be recognised and thus appropriate text can be predicted.

The tool aims to cater to students of all ages to help them keep a check of their progress towards a goal. The bot will learn about the user and adapt to his/her requirements in order to interact more efficiently and suggest appropriate tasks. The umbrella of the suggestions and prompts offered by the bot would be pertaining to the defined academic goals or target of the user. The bot predicts the intents of the user with respect to academic planning. The burden on the user to plan out his/her studies is reduced and the user can actually focus on studying rather than planning. It aims to be a seamless academic companion which keeps the user's tasks and schedule in check.

The remainder of the paper is structured as follows: Section 2 contains Literature Survey, Section 3 contains System Design, Section 4 contains the Methodology, Section 5 contains Results, Section 6 contains Application, Section 7 contains Conclusions and Section 8 talks about the future scope. .

II. LITERATURE SURVEY

Research on conversational agents involves task oriented conversation systems [12] and open domain conversation systems [13]. Other forms of conversational agents include a conversational system with a knowledge base. Leuski et al. [14] implement a system to return the most suitable response to the user input query from the question-answer pairs. This is achieved by utilizing a statistical language model in cross-lingual information retrieval [14], but has a major restriction which takes the form of creation of the knowledge base.

In recent years, deep learning has made significant improvement in NLP [15]. Deep neural networks can extract underlying features of data automatically by exploring multiple layers of non-linear transformation [16]. Yan et al. proposed to establish an automatic conversation system between humans and computers. Given a user input, the proposed system returns the corresponding response based on a deep learning-to-respond schema [13].

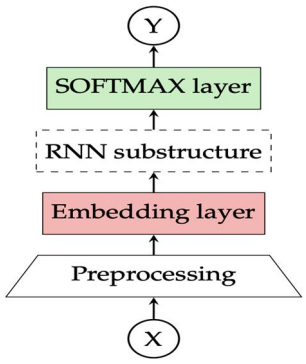


Figure 1 : Base Neural Network Structure [1]

For the neural network model to accurately identify intents and slots from user queries, [1] explores the variations of neural network models for accurate intent detection and proposes a base network structure as seen in Fig. 1. The RNN substructure is replaced by Long Short Term Memory (LSTM), Reversed input sequence LSTM, single GRU, bilayer GRU and Bidirectional GRU. We decided to adopt the single GRU network for intent detection and slot filling since fewer parameters are used to tune as compared to LSTM thereby shorter training time for GRU [1], keeping in mind the overall environmental impact of the methodology, discussed in Section 4 .

Liu et al. [2] propose a hybrid learning method involving supervised learning and reinforcement learning for a task-oriented conversation system. First, the agent interacts with the user using its own policy learned from supervised training. When the agent makes a mistake, user corrects the mistake by demonstrating the agent the right actions. This corrected dialogue sample is then added to the existing training data.

Wen et al. [7] propose a neural network based end-to-end task-oriented conversation system along with a method of collecting dialogue data based on Wizard of Oz (WOZ) framework. It is unknown how well the model performance generalizes to unseen dialogue state during user interactions [2]. On similar lines, Wu et al. [5] worked on MultiWOZ, a human-human dialogue dataset, going a step further by proposing a task oriented conversation system supporting transfer of knowledge across multiple domains.

Vijayakumar et al. [6] proposed a chatbot system as a web service for college related queries using Dialog Flow. Plug and Play Language Model, as mentioned in Dathathri et al. [3] employs a pre-trained language model (LM) which can generate text with controlled attributes such as topic and sentiment. This is utilised in constant text prediction that enables the user to communicate his/her intent quickly.

III. SYSTEM DESIGN

The application follows a three tier architecture [Fig.2]. The front end or the UI Layer consists of the Flutter mobile application written in Dart. The user interacts with the bot through the text interface. The user is provided with speech to text interface for ease of use and the conversational bot's responses can be played as an audio, to cater to all types of users. The service layer consists of the interpretation of the user's inputs. For the intent and slot detection we have used Dialog Flow and an RNN model (GRU network) to work in tandem. For constant text prediction while the user types, PPLM is used. The user profile, user schedule and other metadata is stored in a database. The database acts as a repository of the history of the user's interaction and the analytics gained from that data. The APIs to interact with Flutter are written in Flask and the algorithms and other scripts are implemented in Python.

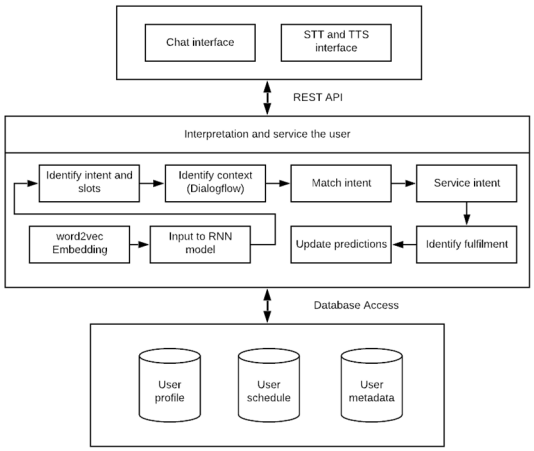


Figure 2 : System Architecture

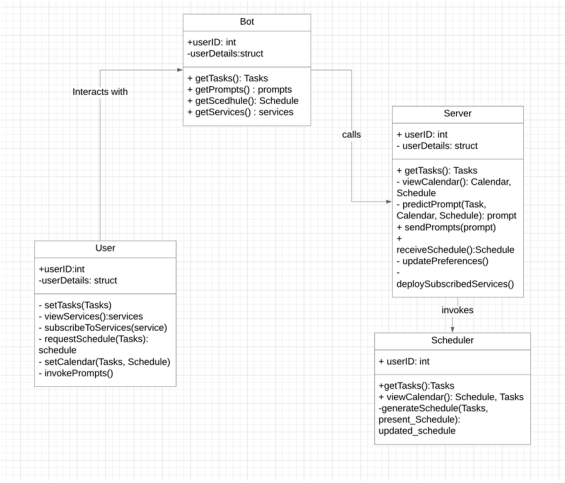


Figure 3 : Class diagram of the system

There are 4 major classes in the class diagram[Fig 3]: User, Conversational Bot, Server and the Scheduler. The user has attributes userID and all of the details input by the user are stored in the attribute userDetails which takes the form of a structure. The user, through the methods contained in the class, can input his tasks, view the services, subscribe to the same, request a schedule given input of tasks he has to do, give his current schedule as his input i.e any meetings, or lectures he has to attend in order to get prompts.

The bot acts as a middleware to interact and take inputs from the user as well connect with the server and pass on the cleaned data received from the user. The bot displays prompts, schedule and the subscribed services of the user.

The server receives all relevant inputs like Tasks, Schedule, User's Calendar and generates prompts, services and invokes them. To generate the schedule, the server calls the scheduler. The scheduler gets all the necessary parameters to generate a schedule for the user, and returns the same.

The interaction of the user with the system can be summarised in the sequence diagram [Fig 4].

The major challenge is accurate execution of the Natural Language Understanding (NLU) components, viz. Intent detection and slot filling. Intent detection or mapping an input text to an intent is considered as a multi-class classification problem [1]. The words contained in the sentence are fed as input to the neural network. The output is a vector of class probabilities. The intent class with the highest probability is considered as the desired intent.

Dialog Flow provides a certain abstraction over the whole process of intent and entity detection, giving the user the options to utilise system information by means of its python module. Since it essentially is a black box, we have designed a recurrent neural network model to consolidate the process.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue IV Apr 2020- Available at www.ijraset.com

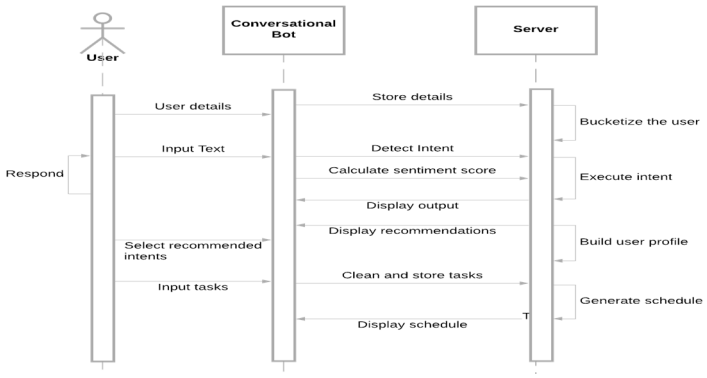


Figure 4 : Sequence diagram of the system

The basic components of Dialog Flow include:

1) *Intent:* Essentially meaning the purpose of the conversation, intents define how conversations work. In each intent, one defines example strings that can trigger the intent, the features to extract from the user string, and the response.
2) *Entity:* It is a feature to identify and extract useful data from user inputs. Entities pick out specific pieces of information that the user mentions. Essentially, entities are the slots.
3) *Context:* It represents the current state of a user's request and allows the agent to transfer information across intents. This enables the agent to provide a natural flow to the conversation.

Based on the sequence to sequence model proposed by Sutskever et al. [8], the model comprises two main components: an encoder RNN and a decoder RNN. Essentially, the encoder encapsulates the information of the input text into a fixed representation and the decoder takes that representation, and generates an optimal variable length text. Optimal signifies an output that best responds to the encoded representation. The model consists of two outputs: the intent confidence scores and the tagged slots. A GRU network to minimize the vanishing gradient problem, and hence the model keeps relevant information and transfers it to the next time step.

IV. METHODOLOGY

The user first signs into the application using his/her Google account. The user is verified and matched with the database. Since the user is recognised, his/her profile is loaded and accordingly recommendations are displayed. The user interacts with the bot by asking him to add a task, create a schedule, make a reminder, and other academic intents. The input text from the Flutter client is passed to the server through a Flask app using REST principles, wherein the input is executed in Dialog Flow using the dialog flow module in Python, simultaneously it is fed as input to the neural network model. In NLP models, a word cannot be fed as the input to the neural network. A typical approach is to map a discrete word to a dense, low-dimensional, real-valued vector, called an embedding [10]. Using word embedding as inputs to Recurrent Neural Networks portray no "curse of dimensionality" and they maintain syntactic and semantic relationship between associated words [17]. For the embedding layer, Word2Vec is used which is a word embedding model proposed by Mikolov et al. [10]. According to Naili et al. Word2Vec presents the best word vector representations with a small dimensional semantic space [4].

As and when an intent is recognised, corresponding changes are made to the database to update user profile as well as the user schedule. The input text along with the confidence scores of all intents for the given input are stored in one table and the input text along with the fulfilment score is stored in another table. The usage of these tables is explained in the recommender system algorithm under the next section.

The intent and slot detection from the user's input text is done using Dialog Flow and the RNN model. The intents are defined on Dialog Flow with output, input contexts and entity detection to ensure a smooth flow of conversation. The recognized intents are: intial_welcome, input_name, input_age, addTask, addDeadline, createSchedule, viewSchedule, deleteTask, addExam, removeExam. Entities are: name, age, exam, task.

The primary purpose of using an RNN model (GRU network) in tandem with Dialog Flow is to avail confidence scores of all intents given a user input string, since Dialogflow offers the confidence score of the detected intent only (the intent with maximum confidence score is termed as detected intent). The model is trained over a custom made dataset with over 90 sample messages. The output obtained from the model is the list of confidence scores of all intents, given an input string, along with the tagged slots in the input sentence.

For the constant text prediction when the user types into the text field, PPLM's freely available implement-

Layer (type)	Output Shape	Param #	Connected to
Input (InputLayer)	[(1, None, 300)]	0	
Encoder (GRU)	[(1, None, 100), (No	120300	Input[0][0]
Decoder (GRU)	(1, None, 100)	60300	Encoder[0][0] Encoder[0][1]
intent_classifier (Dense)	(None, 7)	707	Encoder[0][1]
slots_tagger (Dense)	(1, None, 5)	505	Decoder[0][0]

Total params: 181,812
Trainable params: 181,812
Non-trainable params: 0

Figure 5: Model Summary

Action [3] is used where the input is the string of words typed by the user and the output is the most probable word to be typed by the user. This is a constant process until the user finishes typing. To fine tune the output generated, the parameters "stepsize" and "sentiment" are varied.

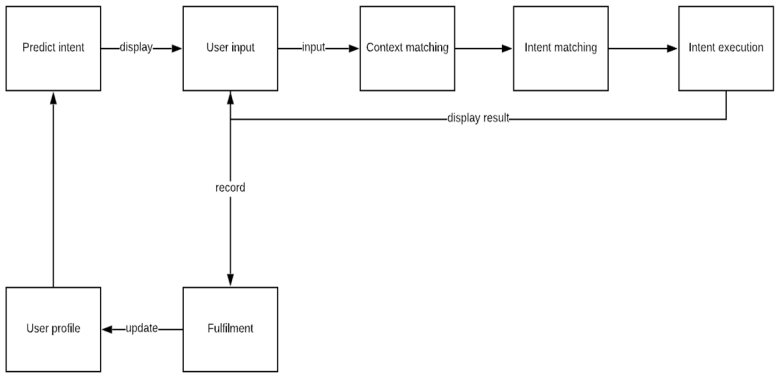


Figure 6 : System flow diagram

When a "create schedule" intent is recognised, the schedule is generated using the proposed Modified Longest Task first algorithm. The "predicted intent strip" contains the top three most likely intents to be fired by the user, which is generated using a content based recommender system. Both the algorithms are discussed below:

A. Algorithms

1) Scheduling Algorithm (Longest Task First)

Traditional Longest task first algorithm -

a) Task with the highest weight is implemented first.

b) Time is allocated to each task based on the total time available and weight proportional to each task.

Modified Longest task algorithm -

a) Basic functionality follows the traditional version.

b) An indicator will be present for indicating the difficulty of each task based on which the weight of that task can be altered to increase the accuracy of the schedule.

c) Time will be allocated for revision/modification of the tasks.

d) Previous schedule check is done to ensure that those time slots are completely free for implementing the given task.

Inputs: Number of tasks, weight of each task, difficulty indicator for each task, total available time duration to complete all tasks, previous schedule for that time period.

2) *Recommendation Algorithm:* To predict the user's next intent according to his/her interaction, we have implemented a content-based recommendation system. This technique attempts to figure out what a user's favourite aspects of an item is, and then recommends items that present those aspects. In our case, we're trying to figure out the user's intent from the input list and fulfilment given from past interactions.

There are 2 tables.

a) Input id | Input text | Intent confidence scores (list of all intents)

Input ID	Input Text	addExam	addTasks	viewSchedule
1.				
2.				

Figure 7: Table 1

b) User input | Fulfilment

Input Text	Fulfilment

Figure 8 : Table 2

i) Given a user input text, input is mapped to the database entry in table 1.
ii) Intent with maximum confidence score is returned to the user.
iii) The sentiment of the following input text is stored as fulfilment score alongside the database entry of the preceding user input text in table 2. In other words, Fulfilment = sentiment score of the query text after the intent is fulfilled.

B. Recommendation

1) From table 1, fetch input ID for the given user input.

Input = Input id | User input | Fulfilment

2) Fetch recorded intent detection confidence for the same input.

userIntentDetection = Input id | intent confidence scores (column for each intent)

3) Generate userProfile. It consists of weights of each intent.

userProfile = userIntentDetection.transpose().dot(input['fulfilment'])

4) To generate recommendation, fetch intentConfidence scores from table 1

5) recommendation =

*((intentConfidence*userProfile).sum()/(userProfile.sum())*

This generates the list of most likely inputs for the user.

Most of the technologies required to be incorporated in the product are open-source, hence do not require external expenditure. The datastore has to be constantly up and running. Some form of replication can be performed to ensure full-time availability.

Since the proposed product comes under Industry 4.0, integrating Industry 4.0 with the sustainable development goals (SDG) in an eco-innovation platform, is really important to ensure environmental performance. We have to consider the deployment, operation, SDG, and the long term scenario. Since the product will be capitalising on constant use of deep learning and NLP techniques, considerable computation power is required to ensure quick training and production of the required output. According to Strubell et al. [18], it was found that "the computational and environmental costs of training an NLP-based deep learning model grew proportionally to model size and then exploded when additional tuning steps were used to increase the model's final accuracy." Thus, the RNN model used in this product is not made unnecessarily complex.

V. RESULTS

The application which contains the functionalities and features as explained in the above sections was developed and tested. The intents were recognised accurately by Dialogflow given input patterns closer to the train dataset. The RNN model consolidates the use of Dialog Flow due to the use of GRUs which eliminate the vanishing gradient problem since the model keeps the relevant information and transfers it to the next time step without washing out the new input every time. The accuracy of intent prediction through Dialog Flow however cannot be noted due to the black box nature of the tool.

The images of the final application portray the conversation held by a user where she adds reminders, tasks and the schedule is displayed by the bot. The images also display the "intent prediction strip" which contains the most likely intents to be fired by the user, as output by the recommender system according to the user's profile.

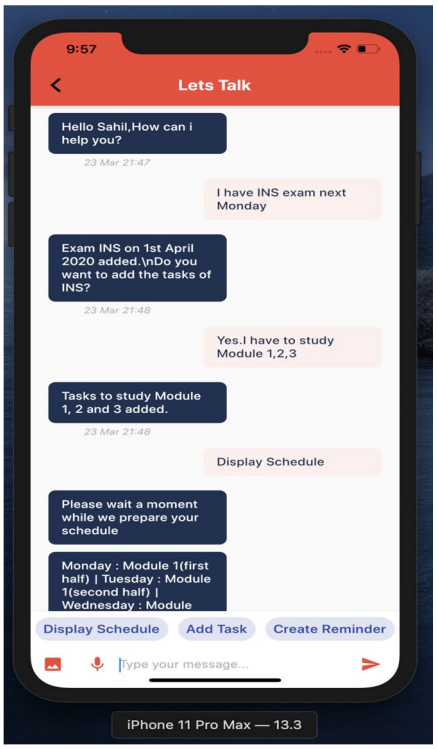


Figure 9: Application screenshot

VI. APPLICATION

The user can either enter the schedule or enter the tasks to be done so that the bot can generate the schedule, and the bot shall interact with the user throughout the day, providing with relevant assistance and thus gaining data insights into the user's daily routine. The bot learns about the user through intent fulfilment and intent confidence scores. The user can input his/her schedule along with the tasks required to be completed and the bot shall generate and recommend a suitable schedule that the user shall follow. For instance, the input shall be the examination schedule along with the topics that need to be studied and the bot shall generate a schedule assigning topics to days.

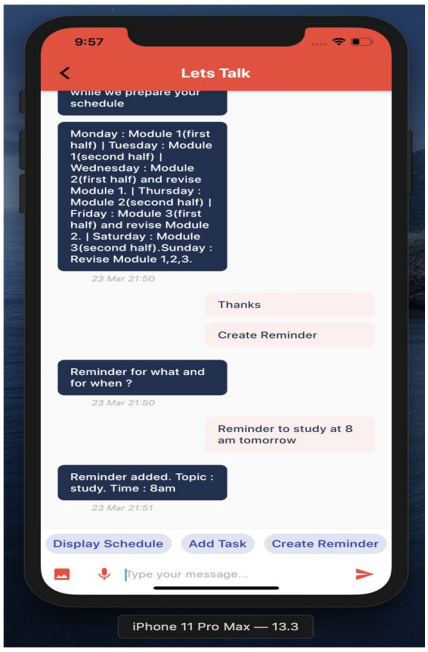


Figure 10: Application screenshot

VII. CONCLUSION

A conversational bot with intents, from which the user can plan out his studies for the exam is implemented using Dialog Flow and the RNN model. The user can add exams to his schedule, add particular tasks to be performed for the exam for which the bot will generate the schedule for the user to follow. The schedule is generated using a modified longest task first algorithm. The bot reminds the user about his tasks and progress at a set time daily. The most likely intents are displayed above the input field using a content based recommender system. The user is provided with speech to text for ease of use and the conversational bot's responses can be played as an audio, to cater to all types of users.

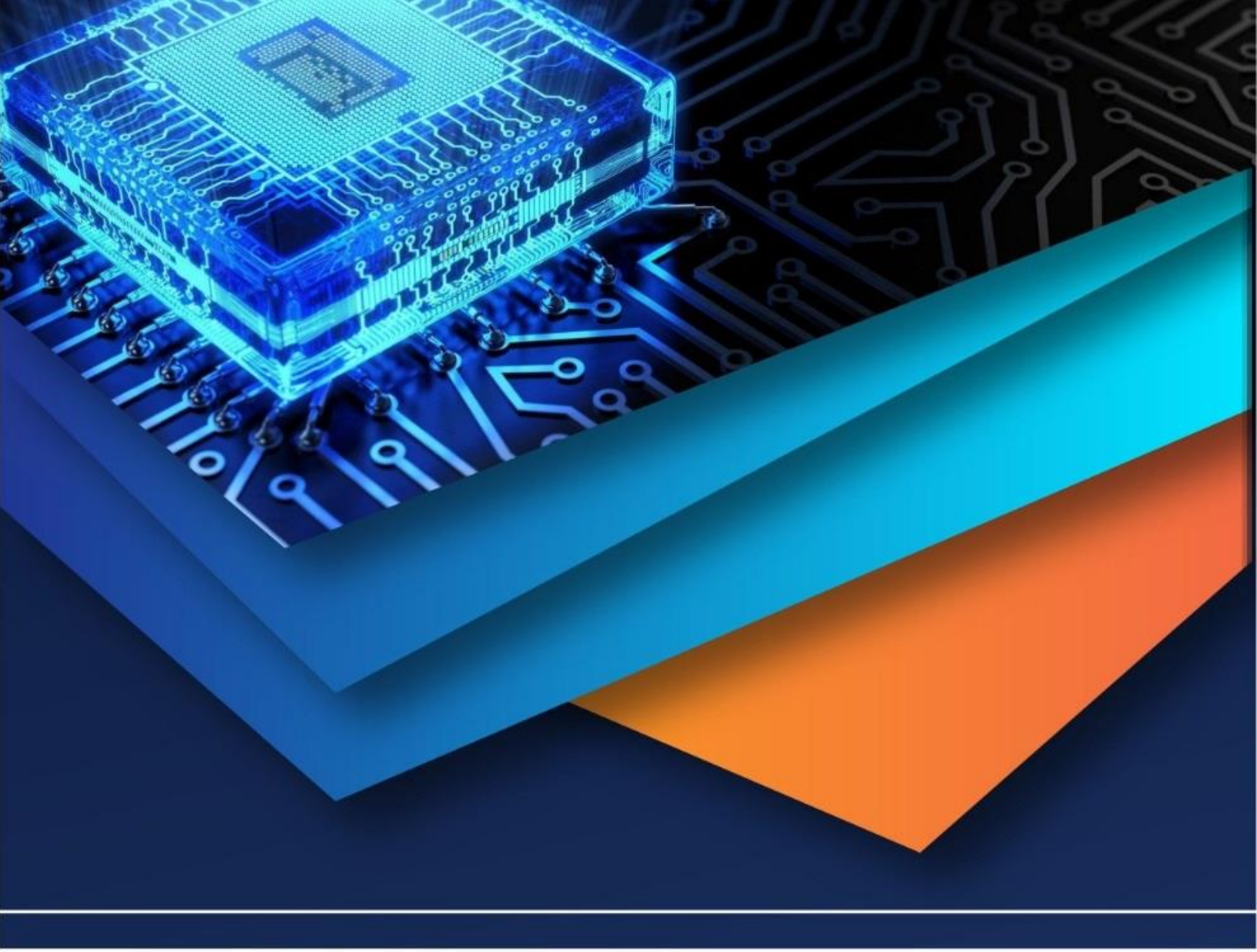
VIII. FUTURE ENHANCEMENT

The scope of the application can be widened in order to incorporate more features, not just limiting itself to exams, timetables and reminders. Various services can be implemented, which the user can subscribe to, for instance, a GRE preparatory word game, in order to boost the user's skills. Furthermore, through Natural Language Understanding, features and services that involve images and videos can be incorporated. For improved accuracy of the intent detection and slot filling, we can use a transformer architecture with multi layer self attention [9].

REFERENCES

[1] Santhosh Kumar Nagarajan. Server-Less Rule-Based Chatbot Using Deep Neural Network. Department of Information Technology, Uppsala University, June 2019. https://uu.diva-portal.org/smash/get/diva2:1365664/FULLTEXT01.pdf
[2] Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. Dialogue learning with human teaching and feedback in end-to-end train- able task-oriented dialogue systems. In NAACL. arXiv preprint arXiv:1804.06512, 2018
[3] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, Rosanne Liu. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. arXiv arXiv:1912.02164, 2019.
[4] Naili, Marwa & Habacha, Anja & Ben Ghezala, Henda. (2017). Comparative study of word embedding methods in topic segmentation. Procedia Computer Science. 112. 340-349. 10.1016/j.procs.2017.08.009.
[5] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, Pascale Fung. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. arXiv:1905.08743, 2019.
[6] Vijayakumar R, Bhuvaneshwari B, Adith S, Deepika M, "AI Based Student Bot for Academic Information System using Machine Learning", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 5 Issue 2, pp. 590-596, March-April 2019. Available at doi : https://doi.org/10.32628/CSEIT1952171 Journal URL : http://ijsrcseit.com/CSEIT1952171
[7] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, Steve Young. A Network-based End-to-End Trainable Task-oriented Dialogue System. arXiv:1604.04562, 2017
[8] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In NIPS, pages 3104–3112, 2014. arXiv:1409.3215, 2014.
[9] Ashish Vaswani, Llion Jones, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. Attention Is All You Need, in the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.
[10] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin. Advances in Pre-Training Distributed Word Representations
[11] Business Insider Intelligence, THE MESSAGING APPS REPORT: Messaging apps are now bigger than social networks (As visited on 29th March, 2020)
[12] K. Zhai and D. J. Williams. Discovering latent structure in task-oriented dialogues. In ACL, pages 36–46, 2014.
[13] Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to Respond with Deep Neural Networks for Retrieval-Based Human-Computer Conversation System. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16). Association for Computing Machinery, New York, NY, USA, 55–64. DOI:https://doi.org/10.1145/2911451.2911542
[14] A. Leuski, R. Patel, D. Traum, and B. Kennedy. Building effective question answering characters. In SIGDIAL, pages 18–27, 2009.
[15] C.-J. Lee, Q. Ai, W. B. Croft, and D. Sheldon. An optimization framework for merging multiple result lists. In CIKM '15, pages 303–312, 2015.
[16] Y. Bengio. Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2(1):1–127, 2009.
[17] Y. Bengio, R. Ducharme, and P. Vincent, "A Neural Probabilistic Language Model", Journal of Machine Learning Research, vol. 3, pp. 932–938, January 2000, https://dl.acm.org/citation.cfm?id=944966.
[18] Emma Strubell, Ananya Ganesh, Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP. arXiv:1906.02243, 2019.

★★★



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY