AI Based Student Guidance Chatbot

# Dr. Ch. Vijaya Kumar1, Thrylokya Jakkam2, Chandulal Kunsothu3, Vamshi Kommana4

*1Associate. Professor, CSE Dept, ACE Engineering College, Hyderabad, India* *vijay.chandarapu@gmail.com*

*2Student, CSE Dept, ACE Engineering College, Hyderabad, India*

*thrylokyasahu@gmail.com*

*3 Student, CSE Dept, ACE Engineering College, Hyderabad, India*

*kunsothchandulal@gmail.com*

*4 Student, CSE Dept, ACE Engineering College, Hyderabad, India*

*kommanavamshi@gmail.com*

**ABSTRACT**

The "AI-BASED STUDENT GUIDANCE CHATBOT" project taken from the Smart India Hackathon website under the domain Smart Education aims to build a Chatbot that democratize access to career guidance for secondary-level students by creating an interactive AI model. This model will empower students to make informed career choices by assessing their aptitudes and interests i.e. they can access various aptitude tests, offering personalized career recommendations, and providing detailed career paths. The project's core objective is to bridge the gap between students' aspirations and the available opportunities, ensuring that every child has access to tailored counselling regardless of their background or locationThis initiative seeks to equip students with the knowledge and confidence needed to embark on fulfilling career journeys, ultimately contributing to their personal growth and the socio-economic development of their committee.

# INTRODUCTION

The aim is to contribute to developing a more secure digital environment by offering an advanced approach to phishing site detection. By In today's fast-paced educational landscape, students encounter a multitude of challenges that demand personalized guidance. Enter the AI-based Student Guidance Chatbot, a revolutionary tool designed to seamlessly integrate technology with mentorship. This cutting-edge solution harnesses the power of artificial intelligence to provide students with tailored assistance in their academic journey.

 With the ability to understand and respond to individual needs, this Chatbot serves as a virtual companion, offering insights on course selection, study strategies, and career planning. It leverages machine learning algorithms to adapt and evolve, continuously improving its capacity to offer relevant advice. The Chatbot isn't just a repository of information.

 This innovative tool also excels in streamlining administrative tasks, aiding students in navigating enrollment procedures, scheduling, and accessing essential resources. The AI-based Chatbot is accessible 24/7, ensuring that students can seek guidance whenever they need it, promoting a more efficient and responsive educational experience.

 In summary, the AI-based Student Guidance Chatbot stands at the forefront of educational support systems, ushering in a new era of personalized, accessible, and intelligent assistance for students navigating the complexities of their academic journey.

# LITERATURE SURVEY

**Georgescu, "Chatbots for Education-Trends Benefits and Challenges, 2018".**

This paper explores the latest trends in using chatbots within educational settings. It discusses the benefits, such as improved student engagement and personalized learning experiences, as well as challenges like technical limitations and resistance to adoption.

**Fleming et al., "Streamlining student course requests using chatbots, 2018".**

This study focuses on the use of chatbots to facilitate the process of student course requests. It highlights how chatbots can simplify administrative tasks, reduce the workload on staff, and provide timely information to students.

**Fernoaga et al., "Intelligent Education Assistant Powered by Chatbots, 2018”.**

This paper presents the development and implementation of an intelligent educational assistant powered by chatbot technology. It emphasizes the assistant's capabilities in providing academic support, answering frequently asked questions, and enhancing the overall educational experience.

**Shevat, Designing bots,Creating conversational experiences, O'Reilly Media, Inc., 2017.**

This book by Amir Shevat provides a comprehensive guide on designing and creating conversational bots. It covers the principles of bot design, user interaction, and best practices for developing engaging and effective chatbot experiences.

**Molnár and Szüts, "The Role of Chatbots in Formal Education", 2018.**

This article investigates the role of chatbots in formal education systems. It examines their impact on teaching and learning processes, the potential for improving educational outcomes, and the integration of chatbots into existing educational frameworks.

**Siswadi and Tarigan, "UGLEO: A Web Based Intelligence Chatbot for Student Admission Portal using MEGAHAL Style", J. Ilm. Inform. Komput., Jan. 2018.**

This research introduces UGLEO, a web-based intelligent chatbot designed for student admission portals using the MEGAHAL style. It describes the chatbot's architecture, functionalities, and effectiveness in assisting prospective students with admission-related inquiries.

1. **COMPARISION ANALYSIS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.****No** | **Paper Title** | **Work done on paper** | **Future work** | **Drawbacks** |
| **1** | Georgescu, "Chatbots for Education-Trends Benefits and Challenges, 2018" | Explores trends, benefits, and challenges of using chatbots in education. Analyzes current applications and their impact on student engagement and personalized learning. | Further research on overcoming technical limitations and increasing adoption rates. Investigation into long-term impacts on learning outcomes. | Technical limitations and resistance from educators and institutions to adopt new technologies. |
| **2** | Fleming et al., "Streamlining student course requests using chatbots, 2018" | Examines the use of chatbots to streamline student course requests. Demonstrates how chatbots can reduce administrative workload and provide timely assistance to students.2019, pp. 1175-1186. | Expansion of chatbot functionalities to cover more administrative tasks. Integration with other educational systems for a more comprehensivesolution. | Limited scope focused mainly on course requests. Potential reliability issues and user acceptance concerns. |
| **3** | Fernoaga et al., "Intelligent Education Assistant Powered by Chatbots", 2018 | Developed an intelligent educational assistant chatbot. Showcased its ability to provide academic support and enhance the educational experience. | Enhancing AI capabilities for more complex queries.Expanding the assistant's reach to cover more subjects and educational levels. | Initial development stages may have limited functionalities. Dependence on accurate data input for optimal performance. |
| **4** | Shevat, Designing bots: Creating conversational experiences, O'Reilly Media, Inc., 2017 | Provides a guide on designing and creating conversational bots. Covers principles of bot design, user interaction, and best practices. | Continued development of design methodologies as technology evolves.Addressing emerging ethical considerationsin bot design. | Practical application examples may become outdated as technology progresses. General guidance may needcustomization for specific educational contexts. |
| **5** | Molnár and Szüts, "The Role of Chatbots in Formal Education", 2018 | Investigates the role of chatbots in formal education. Examines their impact on teaching, learning processes, and educational outcomes. | Longitudinal studies to assess the long-term impact on learning outcomes. | Potential over-reliance on technology by educators and students. Challenges in maintaining human interaction elements ineducation. |
| **6** | Siswadi and Tarigan, "UGLEO: A Web Based Intelligence Chatbot for Student Admission Portal using MEGAHAL Style",Jan. 218 | Introduces UGLEO, an intelligent chatbot for student admissions. Details its architecture and effectiveness in handling admission queries. | Enhancing the chatbot's linguistic capabilities.Expanding its use to other areas of student services. | Initial language processing capabilities may be limited. Dependence on predefined datasets for accurate responses. |

## IMPLEMENTATION

**training.py**

import nltk

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import json

import pickle

import numpy as np

from keras.models import Sequential

from keras.layers import Dense, Activation, Dropout

from tensorflow.keras.optimizers import SGD

import random

words=[]

classes = []

documents = []

ignore\_words = ['?', '!']

data\_file = open('data.json').read()

intents = json.loads(data\_file)

for intent in intents['intents']:

 for pattern in intent['patterns']:

 w = nltk.word\_tokenize(pattern)

 words.extend(w)

 documents.append((w, intent['tag']))

 if intent['tag'] not in classes:

 classes.append(intent['tag'])

words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore\_words]

words = sorted(list(set(words)))

classes = sorted(list(set(classes)))

print (len(documents), "documents")

print (len(classes), "classes", classes)

 print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('texts.pkl','wb'))

pickle.dump(classes,open('labels.pkl','wb'))

training = []

output\_empty = [0] \* len(classes)

for doc in documents:

 bag = []

 pattern\_words = doc[0]

 pattern\_words = [lemmatizer.lemmatize(word.lower())

for w in words:

 bag.append(1) if w in pattern\_words else bag.append(0)

 output\_row = list(output\_empty)

 output\_row[classes.index(doc[1])] = 1

 training.append([bag, output\_row])

random.shuffle(training)

Data\_type = object

training = np.array(training,dtype=Data\_type)

train\_x = list(training[:,0])

train\_y = list(training[:,1]

print("Training data created")

 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(len(train\_y[0]), activation='softmax'))

sgd=SGD(learning\_rate=0.01,momentum=0.9,decay=(0.01/25),nesterov=False)

model.compile(loss='categorical\_crossentropy', optimizer=sgd, metrics=['accuracy'])

hist = model.fit(np.array(train\_x), np.array(train\_y), epochs=200, batch\_size=5, verbose=1)

model.save('model.h5', hist)

print("model created")

 **app.py**

import nltk

nltk.download('popular')

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

import pickle

import numpy as np

from keras.models import load\_model

model = load\_model('model.h5')

import json

import random

intents = json.loads(open('data.json').read())

words = pickle.load(open('texts.pkl','rb'))

classes = pickle.load(open('labels.pkl','rb'))

def clean\_up\_sentence(sentence):

 sentence\_words = nltk.word\_tokenize(sentence)

 sentence\_words = [lemmatizer.lemmatize(word.lower()) for word in sentence\_words]

 return sentence\_words

def bow(sentence, words, show\_details=True):

 sentence\_words = clean\_up\_sentence(sentence)

 bag = [0]\*len(words)

 for s in sentence\_words:

 for i,w in enumerate(words):

 if w == s:

 bag[i] = 1

 if show\_details:

 print ("found in bag: %s" % w)

 return(np.array(bag))

def predict\_class(sentence, model):

 p = bow(sentence, words,show\_details=False)

 res = model.predict(np.array([p]))[0]

 ERROR\_THRESHOLD = 0.25

 results = [[i,r] for i,r in enumerate(res) if r>ERROR\_THRESHOLD]

 results.sort(key=lambda x: x[1], reverse=True)

 return\_list = []

 for r in results:

 return\_list.append({"intent": classes[r[0]], "probability": str(r[1])})

 return return\_list

def getResponse(ints, intents\_json):

 tag = ints[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

def chatbot\_response(msg):

 ints = predict\_class(msg, model)

 res = getResponse(ints, intents)

 return res

from flask import Flask, render\_template, request

app = Flask(\_\_name\_\_)

app.static\_folder = 'static'

@app.route("/")

def home():

 return render\_template("index.html")

@app.route("/get")

def get\_bot\_response():

 userText = request.args.get('msg')

 return chatbot\_response(userText)

if \_\_name\_\_ == "\_\_main\_\_":

 app.run()

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

 <meta charset="UTF-8">

 <title>Chatbot</title>

 <meta charset="UTF-8">

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

 <meta http-equiv="X-UA-Compatible" content="ie=edge">

 <link rel="stylesheet" href="{{ url\_for('static', filename='styles/style.css') }}">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

</head>

<body>

 <!-- partial:index.partial.html -->

 <section class="msger">

 <header class="msger-header">

 <div class="msger-header-title">

 <i class="fa-regular fa-building-columns"></i> Student Guidance Chatbot <i class="fa-regular fa-building-columns"></i>

 </div>

 </header>

 <main class="msger-chat">

 <div class="msg left-msg">

 <div class="msg-img" style="background-image: url('bot3.svg')"></div>

 <div class="msg-bubble">

 <div class="msg-info">

 <div class="msg-info-name">Chatbot</div>

 <div class="msg-info-time">12:45</div>

 </div>

 <div class="msg-text">

 Hi, Welcome to Student Guidance Automated Chatbot! Go ahead and send me a message.

 </div>

 </div>

 </div>

</main>

 <form class="msger-inputarea">

 <input type="text" class="msger-input" id="textInput" placeholder="Enter your message...">

 <button type="submit" class="msger-send-btn">Send</button>

 </form>

 </section>

 <!-- partial -->

 <script src='https://use.fontawesome.com/releases/v5.0.13/js/all.js'></script>

 <script>

 const msgerForm = get(".msger-inputarea");

 const msgerInput = get(".msger-input");

 const msgerChat = get(".msger-chat");

 const BOT\_IMG = "bot3.svg";

 const PERSON\_IMG = "https://image.flaticon.com/icons/svg/145/145867.svg";

 const BOT\_NAME = "ChatBot";

 const PERSON\_NAME = "You";

 msgerForm.addEventListener("submit", event => {

 event.preventDefault();

 const msgText = msgerInput.value;

 if (!msgText) return;

 appendMessage(PERSON\_NAME, PERSON\_IMG, "right", msgText);

 msgerInput.value = "";

 botResponse(msgText);

 });

 function appendMessage(name, img, side, text) {

 // Simple solution for small apps

 const msgHTML = `

<div class="msg ${side}-msg">

 <div class="msg-img" style="background-image: url('bot3.svg)"></div>

 <div class="msg-bubble">

 <div class="msg-info">

 <div class="msg-info-name">${name}</div>

 <div class="msg-info-time">${formatDate(new Date())}</div>

 </div>

 <div class="msg-text">${text}</div>

 </div>

</div>

 msgerChat.insertAdjacentHTML("beforeend", msgHTML);

 msgerChat.scrollTop += 500;

 }

 function botResponse(rawText) {

 // Bot Response

 $.get("/get", { msg: rawText }).done(function (data)

 {console.log(rawText);

 console.log(data);

 const msgText = data;

 appendMessage(BOT\_NAME, BOT\_IMG, "left", msgText);

 });

 }

 function get(selector, root = document) {

 return root.querySelector(selector);

 }

 function formatDate(date)

{

 const h = "0" + date.getHours();

 const m = "0" + date.getMinutes();

 return `${h.slice(-2)}:${m.slice(-2)}`;

 }

 </script>

</body>

</html>

**style.css**

:root

{

 --body-bg: linear-gradient(135deg, #f5f7fa 0%, #c3cfe2 100%);

 --msger-bg: #fff;

 --border: 2px solid #ddd;

 --left-msg-bg: #ececec;

 --right-msg-bg: #579ffb;

}

html

{

 box-sizing: border-box;

}

\*,

\*:before,

\*:after

 {

 margin: 0;

 padding: 0;

 box-sizing: inherit;

}

body

{

 display: flex;

 justify-content: center;

 align-items: center;

 height: 100vh;

 background-image: var(--body-bg);

 font-family: Helvetica, sans-serif;

}

.msger {

 display: flex;

 flex-flow: column wrap;

 justify-content: space-between;

 width: 100%;

 max-width: 867px;

 margin: 25px 10px;

 height: calc(100% - 50px);

 border: var(--border);

 border-radius: 5px;

 background: var(--msger-bg);

 box-shadow: 0 15px 15px -5px rgba(0, 0, 0, 0.2);

}

.msger-header

 {

 /\* display: flex; \*/

 font-size: medium;

 justify-content: space-between;

 padding: 10px;

 text-align: center;

 border-bottom: var(--border);

 background: #eee;

 color: #666;

}

.msger-chat

{

 flex: 1;

 overflow-y: auto;

 padding: 10px;}

.msger-chat::-webkit-scrollbar {

 width: 6px;}

.msger-chat::-webkit-scrollbar-track

{

 background: #ddd;}

.msger-chat::-webkit-scrollbar-thumb

{

 background: #bdbdbd;}

.msg{

 display: flex;

 align-items: flex-end;

 margin-bottom: 10px;}

.msg-img

{

 width: 50px;

 height: 50px;

 margin-right: 10px;

 background: #ddd;

 background-repeat: no-repeat;

 background-position: center;

 background-size: cover;

 border-radius: 50%;}

.msg-bubble

 {

 max-width: 450px;

 padding: 15px;

 border-radius: 15px;

 background: var(--left-msg-bg);}

.msg-info{

 display: flex;

 justify-content: space-between;

 align-items: center;

 margin-bottom: 10px;}

.msg-info-name{

margin-right: 10px;

 font-weight: bold;

}

.msg-info-time {

font-size: 0.85em;}

.left-msg .msg-bubble{

border-bottom-left-radius: 0;}

.right-msg {

 flex-direction: row-reverse;

}

.right-msg .msg-bubble {

background: var(--right-msg-bg);

 color: #fff;

 border-bottom-right-radius: 0;

}

.right-msg .msg-img{

 margin: 0 0 0 10px;

}

.msger-inputarea {

 display: flex;

 padding: 10px;

 border-top: var(--border);

 background: #eee;

}

.msger-inputarea \* {

 padding: 10px;

 border: none;

 border-radius: 3px;

 font-size: 1em;}

.msger-input {

 flex: 1;

 background: #ddd;}

.msger-send-btn {

 margin-left: 10px;

 background: rgb(0, 196, 65);

 color: #fff;

 font-weight: bold;

 cursor: pointer;

 transition: background 0.23s;}

.msger-send-btn:hover{

 background: rgb(0, 180, 50);

}

.msger-chat {

 background-color: #fcfcfe;

 background-image: url(“bot3.svg width='260' height='260'”)}

1. **OUTPUT SCREENS**



##

##

## FUTURE SCOPE

Further enhancements for an AI-based Student Guidance Chatbot could include the following:

**Enhanced Natural Language Understanding (NLU)**

Improve the chatbot's NLU capabilities to better comprehend complex and nuanced student queries, including slang, regional language variations, and context-based questions.Implement sentiment analysis to gauge the emotional tone of students and adjust responses accordingly for more empathetic interactions.

**Expanded Knowledge Base**

Continuously update and expand the chatbot's knowledge base with the latest academic programs, courses, internships, scholarships, and career opportunities.

Integrate with reputable educational databases, industry reports, and job portals to provide comprehensive and up-to-date information to students.

**Integration with Academic Institutions**

Forge partnerships with educational institutions to integrate the chatbot with learning management systems (LMS) and student information systems (SIS).

Enable seamless access to academic records, course registrations, and academic support services through the chatbot interface.

**Continuous Feedback and Improvement**

Implement robust feedback mechanisms to gather insights from students, educators, and counselors about their experiences with the chatbot.

**Personalization and User Profiling:**

Develop advanced user profiling capabilities to personalize recommendations based on individual preferences, academic history, career aspirations, and extracurricular interests. Utilize machine learning algorithms to dynamically adjust the chatbot's responses to align with each student's unique profile.

**Multi-modal Interface:**

Extend the chatbot's capabilities beyond text-based interactions to support voice input/output, image recognition, and multimedia content. This enhancement can cater to diverse learning styles and accessibility needs, providing a more engaging and inclusive user experience.

# CONCLUSION

The conclusion drawn from the review of various journals emphasizes the user-friendly nature of chatbots, which can be utilized by individuals familiar with basic mobile and desktop operations. Customized chatbot implementation relies heavily on sophisticated AI algorithms and comprehensive training data. Ultimately, the deployment of personalized chatbots proves instrumental in saving time and streamlining interactions among individuals.

The efficiency of chatbots can be significantly enhanced by integrating additional machine learning (ML) algorithms. This approach aims to optimize the chatbot's performance and responsiveness, ensuring a seamless user experience. The increasing demand for such systems in our country's IT industry underscores the necessity of developing innovative solutions to accommodate the growing population's needs.

 A chatbot system's fundamental objective is to replicate human-like conversation. Its architecture involves the integration of a language model and computational algorithms designed to simulate natural language interactions between humans and computers. Leveraging AI-based Natural Language Toolkit (NLTK) is a strategic choice due to its effectiveness in chatbot development.

 In summary, the conclusion from the journal review underscores the transformative potential of chatbots in facilitating seamless and efficient human-computer interactions. The integration of AI algorithms, particularly NLTK, enhances the chatbot's performance, paving the way for user-friendly and adaptable systems that cater to the evolving needs of our IT-driven society.

# REFERENCES

1. Georgescu, "Chatbots for Education-Trends Benefits and Challenges,2018
2. M. Fleming et al., "Streamlining student course requests using chatbots, 2018
3. V. Fernoaga et al., "Intelligent Education Assistant Powered by Chatbots", 2018
4. Shevat, Designing bots: Creating conversational experiences, O'Reilly Media, Inc., 2017
5. G. Molnár and Z. Szüts, "The Role of Chatbots in Formal Education", 2018
6. Siswadi and A. Tarigan, "UGLEO: A Web Based Intelligence Chatbot for Student Admission Portal using MEGAHAL Style", J. Ilm. Inform. Komput., Jan. 2018.