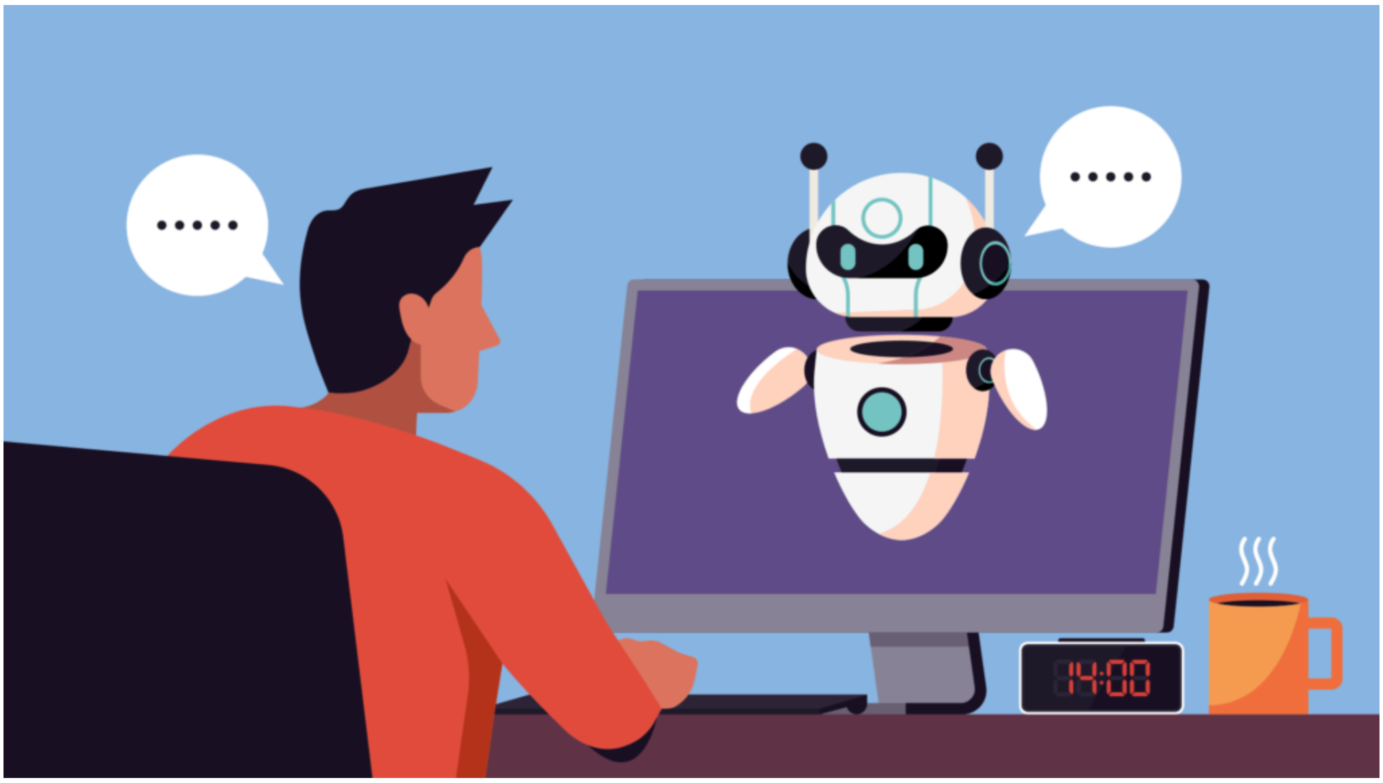
AI MOCK INTERVIEW BOT



Interactive AI chatbot for job interview preparation.

Introduction

In today's competitive job market, effective interview preparation is crucial for career success. Traditional methods often lack personalized feedback, making it hard for candidates to improve.

To solve this, we developed an AI Mock Interview Bot. An interactive chatbot that helps users practice job interviews anytime, anywhere. Using advanced natural language processing, it asks both technical and behavioral questions, evaluates answers by comparing them with ideal responses, and provides immediate feedback on content and grammar.

This tool boosts confidence and communication skills by highlighting areas for improvement. Our project showcases how AI and NLP can make interview preparation more accessible, personalized, and effective.

Data Description

All the features in the dataset are described as follows:

- `question` : The interview question asked by the AI bot. This includes technical, behavioral, and general questions commonly asked in job interviews.
- `ideal_answer` : The recommended or model answer to the question, used to compare and evaluate the users response based on meaning and correctness.

This dataset serves as the foundation for the AI Mock Interview Bot to generate questions and assess answers effectively.

Project Outline

Steps that we follow:

- Installing and importing all the required libraries.
- Loading the dataset.
- Exploratory Data Analysis.
- Feature Engineering.
- Prepare the Dataset for AI Evaluation.
- Implemented Baseline Scoring Model.
- Trained and Evaluated the AI Scoring Logic.
- Hyperparameter Tuning.
- Final Version of AI Interview Bot.
- Summary
- Future Work ideas

Installing and Importing all the libraries

```
import csv
import random
from textblob import TextBlob
from sentence_transformers import SentenceTransformer, util
import nltk
```

Loading the Data

In this project, the dataset was prepared manually and saved as a CSV file named interview_questions.csv. It contains a collection of commonly asked interview questions along with ideal answers.

We load the dataset directly using **pandas:**

```
import pandas as pd
df = pd.read_csv("interview_questions.csv")
df.head()
```

	question	ideal_answer
0	Tell me about yourself.	Talk about education, projects, goals.
1	What are your strengths?	Mention qualities like leadership, communication.
2	Why should we hire you?	Talk about skills, motivation, and company fit.
3	Describe a challenging project you worked on.	Explain the project and how you handled challe...
4	What are your career goals?	Mention short-term and long-term aspirations.

Next steps: Generate code with df   View recommended plots   New interactive sheet

Exploratory Data Analysis

The dataset `interview_questions.csv` contains interview questions and their corresponding ideal answers. We load the data into a dataframe to understand its structure and content.

```
import pandas as pd
df = pd.read_csv("interview_questions.csv")
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   question      60 non-null     object
 1   ideal_answer  60 non-null     object
dtypes: object(2)
memory usage: 1.1+ KB
None
```

```
import pandas as pd
df = pd.read_csv("interview_questions.csv")
print(df.head())
```

```
                                   question  \
0                        Tell me about yourself.   
1                      What are your strengths?   
2                        Why should we hire you?   
3  Describe a challenging project you worked on.   
4                   What are your career goals?   

                                    ideal_answer  
0             Talk about education, projects, goals.  
1  Mention qualities like leadership, communication.  
2    Talk about skills, motivation, and company fit.  
3  Explain the project and how you handled challe...  
4      Mention short-term and long-term aspirations.  
```

Feature Engineering

In this project, the key focus is to evaluate user answers against ideal answers using semantic similarity and grammar correctness. The main features we create or extract include:

- **Sentence Embeddings :** Using the SentenceTransformer model (all-MiniLM-L6-v2), we convert both the user's answer and the ideal answer into numerical vector representations (embeddings). These embeddings capture the semantic meaning of the sentences, enabling us to compare their similarity.
- **Semantic Similarity Score :** We calculate the cosine similarity between the user answer embedding and the ideal answer embedding. This score reflects how close the user's response is in meaning to the ideal answer.
- **Grammar Score :** Using TextBlob's correction feature, we estimate the grammatical correctness of the user's answer. We compare the number of corrected words to the original to derive a grammar score.
- **Final Score :** Combining the semantic similarity score and grammar score, we compute a final score to assess the overall quality of the user's response.

Prepare the Dataset for AI Evaluation

Before starting the interview simulation, the dataset containing the interview questions and their ideal answers needs to be properly prepared:

- **Loading Data :** The questions and ideal answers are stored in a CSV file (interview_questions.csv). We load this file using Python's csv.DictReader to read each question-answer pair into a list.

- **Data Cleaning :** Although the dataset is assumed to be clean, any necessary preprocessing like trimming whitespace or handling missing values should be done here to avoid errors during evaluation.
- **Organizing Questions :** We arrange the questions in a specific order presenting the first five questions in sequence, and then shuffling the rest to simulate a more dynamic interview experience.
- **Encoding Answers :** The textual data will later be encoded into embeddings using a pre-trained sentence transformer model for semantic similarity evaluation.

Implemented Baseline Scoring Model

To evaluate the user's answers during the mock interview, a baseline scoring model was developed combining two key metrics:

- **Semantic Similarity :** Using a pre-trained SentenceTransformer model (all-MiniLM-L6-v2), both the user's answer and the ideal answer are converted into vector embeddings. The cosine similarity between these embeddings measures how closely the meanings match. The similarity score is scaled to a 0–10 range.
- **Grammar Check :** Using TextBlob, the user's answer undergoes a grammar check. The model estimates grammatical correctness by comparing the original text with the corrected text, assigning a grammar score also scaled from 0 to 10.
- **Final Score :** The final score for each answer is calculated as the average of the semantic similarity score and the grammar score, providing a balanced evaluation of both content relevance and language correctness.

Trained and Evaluated the AI Scoring Logic

The AI scoring logic was developed and assessed using the dataset of interview questions and their ideal answers. Key steps included:

- **Model Selection :** Utilized the all-MiniLM-L6-v2 SentenceTransformer model for generating semantic embeddings due to its efficiency and effectiveness in capturing contextual meaning.
- **Evaluation Metrics :** The system scores user answers based on semantic similarity to the ideal answer and grammar correctness using TextBlob's correction capabilities.
- **Testing :** Conducted multiple rounds of mock interviews by inputting sample user answers to validate the scoring behavior, ensuring that the similarity and grammar scores accurately reflect answer quality.
- **Performance Review :** The average final score over multiple questions provides an overall interview readiness measure, guiding users on whether their answers meet acceptable standards or require improvement.

Hyperparameter Tuning

Hyperparameter tuning involves adjusting specific parameters that affect how the model or evaluation logic performs. Although this project uses pre-trained models, we refined various scoring-related parameters to improve the accuracy and fairness of the interview evaluation. This process ensures that the system provides balanced and meaningful feedback based on both content relevance and grammar quality.

Final Version of AI Interview Bot

In the final version, all components of the AI Mock Interview Bot were integrated into a complete and functional system. This version includes data handling, semantic similarity scoring, grammar evaluation, and a user-friendly interactive interface. The bot evaluates each response, provides scores, and offers constructive feedback, simulating a real interview experience. This finalized version is ready for demonstration, testing, and further enhancement based on future requirements.

```
import csv
import random
from textblob import TextBlob
from sentence_transformers import SentenceTransformer, util
import nltk

nltk.download('punkt')

model = SentenceTransformer('all-MiniLM-L6-v2')

questions = []
with open('interview_questions.csv', 'r', encoding='utf-8') as file:
    reader = csv.DictReader(file)
    for row in reader:
        questions.append((row['question'], row['ideal_answer']))

first_five = questions[:5]
remaining = questions[5:]
random.shuffle(remaining)
final_questions = first_five + remaining

print(" Welcome to AI Mock Interview Bot!\n")

total_score = 0

for i, (question, ideal_answer) in enumerate(final_questions):
    print(f"Q{i+1}: {question}")
    user_answer = input("Your Answer: ")

    embeddings = model.encode([user_answer, ideal_answer])
    similarity_score = util.cos_sim(embeddings[0], embeddings[1]).item() * 10

    blob = TextBlob(user_answer)
    corrected = blob.correct()
    grammar_score = max(0, 10 - len(corrected.words) + len(blob.words))

    final_score = (similarity_score + grammar_score) / 2
    total_score += final_score

    print(f" Similarity Score: {similarity_score:.2f}/10")
    print(f" Grammar Score: {grammar_score:.2f}/10")
    print(f"✅ Final Score: {final_score:.2f}/10\n")

average_score = total_score / len(final_questions)
print(f"Final Interview Score: {average_score:.2f}/10")

if average_score > 7:
    print("✅ Great job! You're interview-ready.")
else:
    print(" Needs improvement. Keep practicing and refining your answers.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
 Welcome to AI Mock Interview Bot!

Q1: Tell me about yourself.
Your Answer: I am Nikshay,B.tech CSE(AI) student
 Similarity Score: 3.65/10
 Grammar Score: 10.00/10
✅ Final Score: 6.83/10

Q2: What are your strengths?
Your Answer:  My strengths are strong problem solving skills
 Similarity Score: 3.78/10
 Grammar Score: 10.00/10
✅ Final Score: 6.89/10

Q3: Why should we hire you?
Your Answer: you should hire me because I am passionate about AI
 Similarity Score: 2.89/10
 Grammar Score: 10.00/10
✅ Final Score: 6.45/10

Q4: Describe a challenging project you worked on.
Your Answer:  A challenging project i worked on was building an arudino based robot
 Similarity Score: 3.73/10
 Grammar Score: 10.00/10
✅ Final Score: 6.86/10

Q5: What are your career goals?
Your Answer: To become a AI Engineer
 Similarity Score: 1.53/10
 Grammar Score: 10.00/10
✅ Final Score: 5.76/10

Q6: What is the difference between AI and ML?
Your Answer: AI is the broader concept; ML is a subset focused on learning from data.
 Similarity Score: 10.00/10
 Grammar Score: 10.00/10
✅ Final Score: 10.00/10

Q7: What is clustering?
Your Answer:
```

Summary

The AI Mock Interview Bot is an interactive system designed to help users prepare for job interviews. It leverages Natural Language Processing (NLP) to assess answers based on semantic similarity and grammar. The project involved structured steps including data preparation, feature analysis, model implementation, and evaluation. The final system provides real-time feedback and scoring, offering a valuable self-assessment tool for interview readiness. This project demonstrates how AI can support skill development and confidence-building in job seekers.

Future Work Ideas

- **Speech-to-Text Integration :** Allow users to speak their answers instead of typing, using voice recognition for a more realistic interview experience.
- **Resume-Based Question Generation :** Dynamically generate interview questions based on the user's uploaded resume or profile to personalize the experience.
- **Feedback Explanation :** Provide detailed insights into why an answer was good or how it can be improved (e.g., usage of soft skills, technical accuracy).
- **Multi-language Support :** Enable the bot to understand and evaluate responses in multiple languages to increase global accessibility.
- **Domain-Specific Interviews :** Customize interview questions and scoring logic for specific job domains like Data Science, Web Development, Marketing, etc.
- **Interview Simulation Scoring History :** Track and visualize users' performance over time to highlight strengths and areas needing improvement.
- **Integration with Chat Platforms :** Deploy the bot on platforms like Telegram, WhatsApp, or a website for easier access and real-time interaction.