**Mid term report on mini project**

On

**Interviewer Chat bot**

Based On

**Machine Learning**

Done By

**Akshat Goyal(171500028)**

**Rohitash Yadav(171500268)**

In Under Guidance Of

**Mr Piyush Vashistha**

**(**Asstt. Professor)

Submitted To

****

**Department Of Computer Science And Engineering**

**Abstract**

Chat bots, or conversational interfaces as they are also known, present a new way for individuals to interact with computer systems. Traditionally, to get a question answered by a software program involved using a search engine, or filling out a form. A chat bot allows a user to simply ask questions in the same manner that they would address a human. The most well known chat bots currently are voice chat bots: Alexa and Siri. However, chat bots are currently being adopted at a high rate on computer chat platforms.

The technology at the core of the rise of the chat bot is natural language processing (“NLP”). Recent advances in machine learning have greatly improved the accuracy and effectiveness of natural language processing, making chat bots a viable option for many organizations. This improvement in NLP is firing a great deal of additional research which should lead to continued improvement in the effectiveness of chat bots in the years to come.

A simple chatbot can be created by loading an FAQ (frequently asked questions) into chat bot software. The functionality of the chat bot can be improved by integrating it into the organization’s enterprise software, allowing more personal questions to be answered, like“What is my balance?”, or “What is the status of my order?”.

Most commercial chat bots are dependent on platforms created by the technology giants for their natural language processing. These include Amazon Lex, Microsoft Cognitive Services, Google Cloud Natural Language API, Facebook DeepText, and IBM Watson. Platforms where chat bots are deployed include Facebook Messenger, Skype, and Slack, among many others.

**Table Of Content**

1. Introduction 1-5

1.1General Introduction To Topic 1

1.2 Area of Computer Science 2-4

1.3 Hardware And Software Requirements 5

1. Problem Definition 6
2. Objective 7
3. Methodology 8
4. Implementation Details 9-12
5. Contribution summary 12
6. Progress In Present Time 12
7. work left 13

9. references 13

**Introduction**

General Introduction To Topic:-

Chat botperforming a similar role for online interviews, helping eliminate bias, without the need for unearthly heads that may put candidates off. Any business can build its own chat bot to conduct such interviews, or acquire one off-the-shelf, or have one custom-made depending on the depth of the interview tasks.

Chat bots solutions can operate using a plain script, use natural language processing (NLP) and natural language understanding (NLU) to recognize and react to keywords from the candidate while deeper AI bots can better understand complex conversations and access wider data sets to respond in an appropriate manner.

In the very near future, incredibly smart bots will be able to field all types of interview questions and analyse the responses from candidates. For now, as chat bots become more common in HR, they can help provide a baseline set of results for first interviews. They can also deliver knowledge, aptitude and other tests to all candidates.

If a business is hiring temporary or seasonal employees, the bot can be used to identify groups of suitable candidates based on objectivity and skill set, rather than relying on intuition, their demeanour and other elements that might influence recruiters during a face-to-face interview.

**Area of Computer Science:-**

Computer Architecture

Energy-Efficient Systems: With the explosive growth in mobile devices, there has been a push towards increasing energy efficiency of computation for longer battery life. Reducing power consumption is also important for desktop computing to alleviate challenges of heat removal and power delivery. A special focus in our department has been on the development of energy-efficient graphics hardware. Another area of future interest is energy-harvesting systems, which are ultra-low-power systems that operate on energy scavenged from the environment.

Computer Graphics

### Computer graphics research at UNC investigates the representation, creation, and manipulation of image data that serve as the visual interface between people and computers. One of our primary research focuses is interactive graphics where the main challenges are the rapid generation of photo realistic images and high-quality simulation in response to user inputs, as well as the development of both software and hardware mechanisms for human interaction with graphical systems.

High-Performance

Computing Parallel programming models and their embodiment in programming languages and run time systems. Programming models for nested parallelism, multi - core and many-core processors, and the parallel memory hierarchy. Design and analysis of parallel algorithms.

Application of HPC principles and techniques for real-time physically based simulations and for large-scale scientific computing problems. Examples include simulation of physical, visual, and acoustic properties of spaces and materials, and computational fluid dynamics problems to understand mechanisms of flying and swimming in organisms from tiny insects to giant whales.

Machine Learning

Machine Learning: The problems we study combine vast amounts and disparate types of measurements with equally complex prior knowledge, posing unique challenges for machine learning. Our interests include both modeling paradigms, such as Bayesian nonparametric methods, and inference methodologies, such as MCMC, variational methods and convex optimization. We also work on structured, interpretable, and generalizable deep learning models. Other topics of focus include multi-task learning, reinforcement learning, and transfer learning.

Natural Language Processing

Natural language processing (NLP) or computational linguistics is an area in machine learning and artificial intelligence that deals with understanding the meaning of human-style language and its interactions with machines. NLP research in the UNC Department of Computer Science (Prof. Bansal’s group) focuses on human-like language generation and question-answering/dialogue, multimodal, grounded, and embodied semantics (i.e., language with vision and speech, for robotics), and interpretable and structured deep and structured learning models. It also has strong connections with computer vision and machine learning research (Prof. Alex Berg’s and Tamara Berg’s groups) for natural language generation based on image content and text-to-image reference resolution.

Networking

Subareas: Distributed Systems, Internet Measurements, Multimedia Systems, Multimedia Transport, Network Protocols

Networking research at UNC is centered on how to provide better or more efficient communication services by making sure that the components of distributed computers and networks all work together seamlessly. Specifically, the research focus of Drs. Jeffay, Smith, and Aikat is on network management software, software for routers and experimental methods for networking research. Dr. Kaur concentrates on the communications software found in PCs as well as large systems and Dr. Mayer-Patel’s research is directed at different ways to make the media used in applications richer. Drs. Reiter and Monrose work on ways to make networks more secure. Dr. Nirjon works on low-power networks and radio frequency (RF) sensing techniques.

Real-Time Systems

The Real-Time Systems Group studies application systems in which timing constraints exist. Usually, such systems are structured as a set of tasks (i.e., programs) and timing constraints are defined in terms of per-task deadline requirements. To ensure that timing constraints are met, offline validation algorithms are required that check whether deadlines will be met at runtime. The design of such validation algorithms hinges on how tasks are scheduled and how various complexities such as synchronization are addressed. Accordingly, research involving the design of algorithms for validating, scheduling, and synchronizing real-time task systems has been a continual focus within the group. Additionally, the development and evaluation of operating system infrastructure, in the form of scheduling and synchronization methods, has been an active topic of investigation. Many of the group’s current efforts focus on resource allocation problems that arise when implementing real-time applications on multicore machines.

Robotics

Subareas: Assistive Robotics, Manipulation, Medical Robotics, Motion Planning & Control, Robot Learning, Robot Perception (see: Computer Vision)

Robotics is increasingly affecting our daily lives and impacting healthcare, transportation, defense, manufacturing, and entertainment. New robots and algorithms are enabling physicians to perform more precise surgical procedures, assisting individuals with tasks of daily living, enabling autonomous agents to maneuver through cluttered environments, and manipulating materials at large and small scales.

At UNC, we are creating new algorithms to address fundamental computational challenges in robotics, including increasing the autonomy of robots, motion planning in complex environments, and providing new interfaces for natural human-robot interaction. We bring a broad range of expertise to these problems, including geometric methods, probabilistic methods, physically-based simulation, many-core CPU and GPU parallelization, cloud computing, machine learning, natural language processing, and computer vision. We apply the new algorithms and methods we develop to a variety of applications involving both physical robots as well as virtual agents. Current applications include robot-assisted medical procedures, surgery training, robot design optimization, autonomous vehicles, and personal assistant robots.

Security:-

Subareas: Cloud Computing Security, Hardware Security, Mobile Device Security, Network Security

Security research focuses on developing techniques to defend computer systems and networks against misuse and interference. Presently our department is engaged in several research directions in this general area.

Network security: Today’s Internet infrastructure is a common target of attack and the vehicle for numerous unwanted activities in network applications (e.g., spam, phishing).  We are conducting research to evaluate the extent of these vulnerabilities and to develop defenses against them.  This includes research on both protecting the Internet infrastructure from attack and designing defenses within the context of network applications.

**Hardware And Software Requirements:-**

**Hardware:-**

* **2-GB Ram(recommended)**
* **32-Bit Operating System**
* **Processor - Intel Dual and More**

**Software:-**

* **Jupyter Notebook**
* **Spyder**
* **Pychram**
* **Python 3**

**Problem Definition**

**Scripted bots ask questions and record responses from a set list**

Scripted bots walk users through a decision tree and are effective ways to collect data for precisely defined situations. College degree? Check. Driver’s license? Check. Late-model car? Check. Willing to relocate? Check. Seventy-five percent travel? Check. Stand on head for hours? No check. Sorry, you don’t qualify. Having a machine work through the drudgery of qualifying 500 applicants on these knockout questions is a godsend.

**Unscripted bots take user questions and offer answers from a data set**

Unscripted bots attempt to get answers for users from an archive of company information. They are like a search engine designed to provide answers to employee questions. When applied to HR, they tend to be tools that guide benefits choices, answer questions about policies, help employees use their benefits, provide intake for complaints or navigate administrative processes.

The knowledge-acquisition section of the bot consumes company policies, procedures, FAQs and other documentation to create a pool of indexed, searchable data. The translation function interprets user queries (in text or sometimes voice). The matching operation pairs the two to produce an answer.

Unscripted bots can be trained to deliver the right result approximately 80 percent of the time. If the company data are complete (and that’s often a big, expensive assumption), an unscripted bot can understand the question and deliver the correct answer four times out of five.

**When bots work well and when they don’t**

If the problem is simple or just a little complicated (such as determining who oversees sexual-harassment reports), answers are easy. So if what you want is a rigorous and unforgiving application of policy to specific transactions, chatbots are a logical solution. HR certainly has its share of procedural areas for which exacting precision is a necessity.

Chatbots (and all current intelligent tools) work best when the project involves a question that can be clarified with a structured set of subordinate questions. But things get more difficult when the topic is complicated and become impossible when it is complex or chaotic.

**Objective**

There are following objective of our project-

1. Candidates are comfortable interacting with a chat bot

* Candidates are very comfortable with it.
* This minimize the fear of candidates during the interview.
* Session becomes very interactive.
* It consumes very less time during interview.

1. A chat bot can help recruiters with administrative tasks

* collect information from candidates such as their resume and contact information
* ask screening questions about candidates’ experience, knowledge, and skills
* rank candidates on metrics such as qualifications, engagement, or recent activity
* answer FAQs about the job and the application process
* schedule an interview with a human recruiter

**Implementation details**

We follow the following steps for this project

* For this project programming language that is used is python
* For compilation of code we use jupyter notebook.
* In this code of project there are total 5 methods in present. That are following:-

>first is Start interview-for starting the conversation.

> second is end interview-for ending the conversation.

> third is respond-for response of question and answer asked by interviewer and interviewee.

> Fourth is question loop-for the question that are asked by interviewer during the interview.

> Fifth is common\_question-used for question like whatis your name and other common question

**Code-**

from email.mime.text import MIMEText  
from termcolor import colored  
import markovify  
import datetime  
import smtplib  
import random  
import time  
import json  
import sys  
import os  
  
# Get model for prompts  
with open('./model.json') as data\_file:  
    model = json.load(data\_file)  
  
# Get raw text as string.  
with open("./words.txt") as f:  
    text = f.read()  
  
index = 0  
transcript = {}  
interviewer = ''  
email = ''  
  
bot\_email = os.environ.get('INTERVIEW\_BOT\_EMAIL')  
bot\_email\_password = os.environ.get('INTERVIEW\_BOT\_EMAIL\_PASSWORD')  
interviewee = os.environ.get('INTERVIEWEE')  
interviewee\_email = os.environ.get('INTERVIEWEE\_EMAIL')  
  
def is\_common\_question(interview\_question):  
    q = interview\_question.lower()  
    answer = False  
  
    for j in model['common']:  
        hasQ = j in q  
        # question is within 10 characters of common question  
        closeEnough = len(q) <= (len(j) + 5) and len(q) >= (len(j) - 5)  
        if hasQ and closeEnough:  
            answer = str(model['common'][j])  
  
    return answer  
  
def respond(interview\_question):  
    global index  
    global transcript  
    answer = ''  
  
    # get question words  
    wordlist = interview\_question.split()  
  
    # check if actually a question  
    if not wordlist[-1].endswith('?') and wordlist[-1] != '?':  
        return 'Please, let\'s stick to questions only.'  
  
    # check if already asked  
    if (len(transcript) >= 1):  
        for i in transcript:  
            if (transcript[i]['question'].lower() == interview\_question.lower()):  
                return interviewer + ', please. You already asked me that.'  
  
    # check if common question  
    common = is\_common\_question(interview\_question)  
    if (type(common) is str):  
        return common  
  
    # Build the model.  
    text\_model = markovify.Text(text)  
  
    # generate response  
    for i in range(random.randint(1, 5)):  
        answer += text\_model.make\_sentence()  
  
    # save question + response  
    # transcript[str(index)]['answer'] = nswer  
    transcript.update({index: {  
            'question': interview\_question,  
            'response': answer  
        }})  
  
    # increment index + return response  
    index += 1  
    return answer  
  
def end\_interview():  
    print (colored('Ok, ', 'red') + colored(interviewer, 'magenta') + colored(', thats enough questions for today. Thank you.', 'red'))  
    print (colored('I\'ll send you an email of our conversation.', 'red'))  
    print ('')  
  
    message = ''  
  
    for q in transcript:  
        message += ('Question #' + str(q) + ': ')  
        message += '\n'  
        message += ('You asked: ' + transcript[q]['question'])  
        message += '\n'  
        message += ('I answered: ' + transcript[q]['response'])  
        message += '\n'  
        message += '\n'  
  
    content = MIMEText(str(message))  
    content['Subject'] = interviewer + ', here\'s the ' + interviewee + ' Interview from ' + str(datetime.date.today()) + ' | ' + time.strftime("%H:%M:%S")  
  
    s = smtplib.SMTP('[smtp.gmail.com](http://smtp.gmail.com/" \t "https://mail.google.com/mail/u/1/" \l "sent/_blank)', 587)  
    s.ehlo()  
    s.starttls()  
    s.login(bot\_email, bot\_email\_password)  
    s.sendmail(bot\_email, email, content.as\_string())  
    s.sendmail(interviewee\_email, email, content.as\_string())  
    s.close()  
  
    sys.exit()  
  
def question\_loop():  
    global index  
  
    if index > len(model['prompts']) - 1:  
        end\_interview()  
        return  
  
    question\_str = model['prompts'][index]  
    interview\_question = input(colored(question\_str, 'red'))  
    response = respond(interview\_question)  
    print( colored('-----------------------------', 'red'))  
    print (colored(response, 'blue'))  
    print ('')  
  
    time.sleep(1)  
    question\_loop()  
  
def start\_interview():  
    global interviewer  
    global email  
    interviewer = input(colored('Hey there! What\'s your name? => ', 'red'))  
    print( '')  
    email = input(colored(interviewer, 'magenta') + colored(', What\'s your email? I\'ll send a transcript of our interview => ', 'yellow'))  
    print ('')  
  
    question\_loop()  
  
start\_interview()

**Contributing summary**

There are total two members in the team;

**Akshat Goyal**:-He design the methods Question\_loop and Response and also design the data set.

**Rohitash yadav**:-He design methods like Start\_interview and Stop\_interview.

**Progress In present time**

* Back end code is completed

**Work Left**

* Front end designing is left and some data related to conversation during the interview also left.

**References**

* Git hub
* Geeks for geeks
* Kaggle

[https://www.python.org](https://www.upgrad.com/blog/6-interesting-machine-learning-project-ideas-for-beginners/)

[https://www.upgrad.com](https://www.upgrad.com/blog/6-interesting-machine-learning-project-ideas-for-beginners/)