RETRIEVAL-BASED CHATBOT USING NLP AND PYTHON

A mini project report submitted by

VENNA ROHITH SAI (20181CSE0785)
VINAY K P (20181CSE0790)
VINAY LAKSHMAN M (20181CSE0791)

as part of lab-based course Programming in Python, CSE 317
of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

under the supervision of

Ms. Napa Lakshmi, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

PRESIDENCY UNIVERSITY
Itgalpur Rajanakunte, Yelahanka, Bengaluru, Karnataka-560064

November 2020

PRESIDENCY UNIVERSITY

(Established under the Presidency University Act, 2013 of the Karnataka Act 41 of 2013)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project report entitled, **"Retrieval-based chatbot using NLP and Python"** is a bonafide record of Mini Project work done as part of CSE 317 Programming in Python during the academic year 2020-2021 by

VENNA ROHITH SAI (20181CSE0785)
VINAY K P (20181CSE0790)
VINAY LAKSHMAN M (20181CSE0791)

Submitted for the Viva Voice held on _____

Examiner

ABSTRACT

In this mini-project, we are going to build a simple retrieval based chatbot using python and the nltk library. In retrieval-based models, a chatbot uses some heuristic to select a response from a library of predefined responses. The chatbot uses the message and context of the conversation for selecting the best response from a predefined list of bot messages. The context can include a current position in the dialogue tree, all previous messages in the conversation, previously saved variables (e.g. username). Heuristics for selecting a response can be engineered in many different ways, from rule-based if-else conditional logic to machine learning classifiers. Though it is a very simple bot with hardly any cognitive skills, its a good way to get into NLP and get to know about chatbots. Though 'CT (Cloud Tech)' responds to user input, for a production system we will need one of the existing bot platforms or frameworks, but this example should help us think through the design and challenge of creating a chatbot

CONTENTS

Abstract **i**

1. Introduction **5**

1.1 Objectives

1.2 Motivation

1.3 Overview of the Project

2. Design and Architecture **6-7**

2.1 Architecture/Working

2.2 Anatomy of a Chatbot

3. Implementation. **8-15**

3.1. Pre-requisite software/packages

3.2. Module description/ Algorithms and techniques used

3.3. Project code

4. Conclusion and future scope. **16**

4.1. Conclusion

4.2. Limitations

4.3 Future Scope

5. Output and screenshots **17-18**

References **19**

CHAPTER 1
INTRODUCTION

1.1 Objectives:

- Building a simple chatbot from scratch in python using NLTK library.
- Build a retrieval-based bot.
- respond to questions based on cloud computing.

1.2 Motivation:

Nowadays there are a variety of chatbots being used all over the internet for all sorts of things. They range from simple rule-based bots to complex generative bots that can have actual conversations with people.

The bot built in this project is a great way to learn more about how chatbots function and the different methods they use to generate responses, thereby it serves as a great introduction to analysis of chatbots and NLP in python.

1.3 Overview of the Project:

In this mini-project, we are going to build a simple retrieval based chatbot using python and the NLTK library. We pre-process the text from the corpus using stemming and lemmatization functions in the NLTK library. Then we use the TF-IDF method to generate responses to the queries entered by the user.

CHAPTER 2
DESIGN AND ARCHITECTURE

2.1 Architecture/Working:

A chatbot is an artificial intelligence-powered piece of software in a device (Siri, Alexa, Google Assistant etc…), application, website or other networks that try to gauge consumer's needs and then assist them to perform a particular task like a commercial transaction, hotel booking, form submission etc… Today almost every company has a chatbot deployed to engage with the users.

There are broadly two variants of chatbots: Rule-Based and Self-learning:

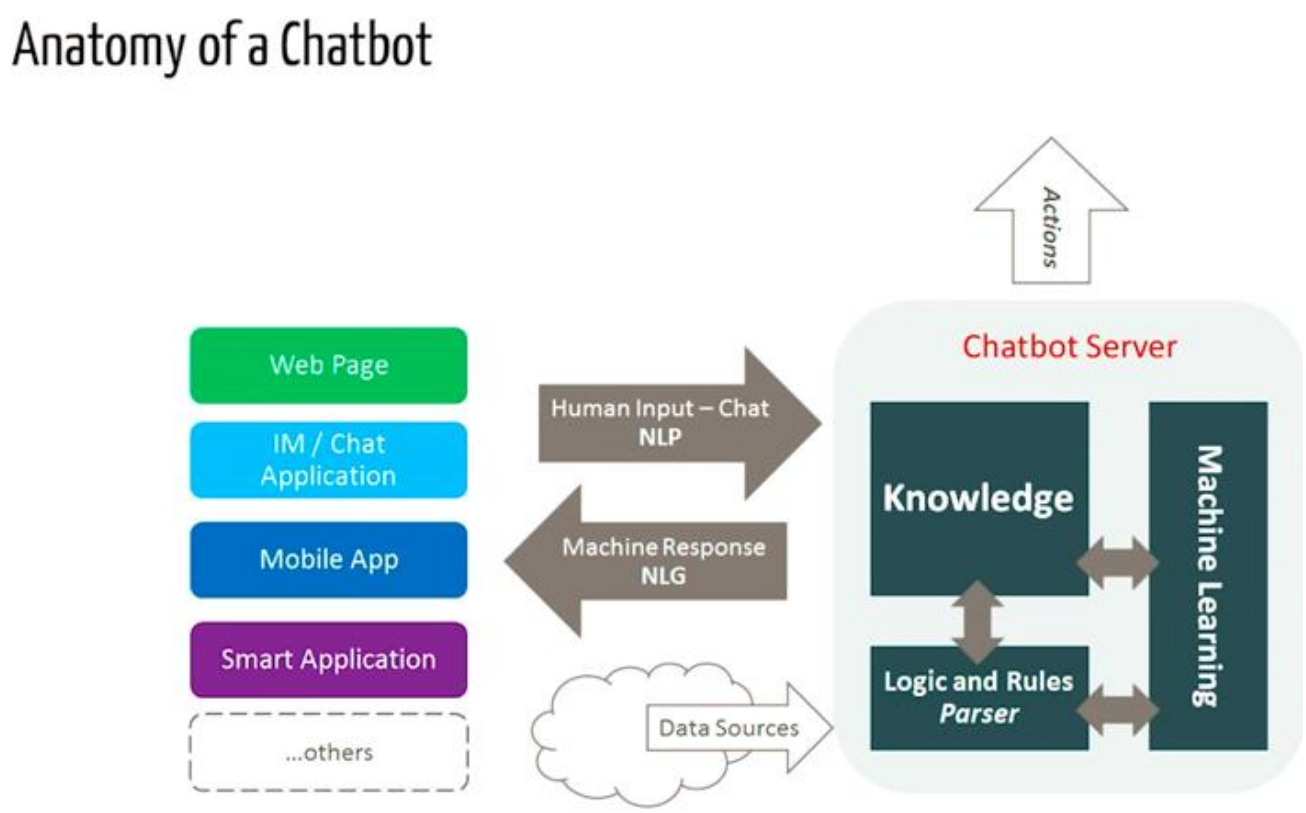
1. In a Rule-based approach, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. The bots can handle simple queries but fail to manage complex ones.

2. Self-learning bots are the ones that use some Machine Learning-based approaches and are definitely more efficient than rule-based bots. These bots can be of further two types: Retrieval Based or Generative.

    - In retrieval-based models, a chatbot uses some heuristic to select a response from a library of predefined responses. The chatbot uses the message and context of the conversation for selecting the best response from a predefined list of bot messages. The context can include a current position in the dialogue tree, all previous messages in the conversation, previously saved variables (e.g. username). Heuristics for selecting a response can be engineered in many different ways, from rule-based if-else conditional logic to machine learning classifiers.

    - Generative bots can generate the answers and not always replies with one of the answers from a set of answers. This makes them more intelligent as they take word by word from the query and generates the answers.

2.2 Anatomy of a Chatbot:



The above diagram shows the different steps involved in how the chatbot works to generate a response to a given query

Here, first the input (query) is taken from the user through various means listed above like Web pages, mobile apps etc. This is then passed on to the Chatbot server that stores the knowledge gained by the chatbot and the logic and rules used, in combination with machine learning to give back a meaningful response to the user.

The more input the bot receives from the user, the more it learns and trains on the given input to perform better and give meaningful responses. Another way of feeding knowledge to the bot is various data sources like, text corpuses specifically designed for bots. These are directly stored in the Knowledge database in the Chatbot server and used from there.

CHAPTER 3
IMPLEMENTATION

3.1 Pre-requisite software/packages:

- Notebooks preferred. (jupyter notebook, google colab, etc)
- Python 3.5 or above.
- NLTK and sklearn packages installed.

3.2 Module description/ Algorithms and techniques used:

NLP (Natural Language Processing):

NLP is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation.

Downloading and installing NLTK:

NLTK (Natural Language Toolkit) is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

Reading in the corpus:

For our example, we will be using an article on cloud computing from the internet as our corpus. Copy the contents from the page and place it in a text file named 'cc.txt'. However, you can use any corpus of your choice.

The main issue with text data is that it is all in text format (strings). However, the Machine learning algorithms need some sort of numerical feature vector in order to perform the task. So before we start with any NLP project we need to pre-process it to make it ideal for working. Basic text pre-processing includes:

- Converting the entire text into uppercase or lowercase, so that the algorithm does not treat the same words in different cases as different
- **Tokenization**: Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e. words that we actually want. Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.
- Removing Noise, i.e. everything that isn't in a standard number or letter.
- Removing the Stop words. Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words

- **Stemming**: Stemming is the process of reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form. Example if we were to stem the following words: “Stems”, “Stemming”, “Stemmed”, and “Stemitization”, the result would be a single word “stem”.
- **Lemmatization**: A slight variant of stemming is lemmatization. The major difference between these is that, stemming can often create non-existent words, whereas lemmas are actual words. So, your root stem, meaning the word you end up with, is not something you can just look up in a dictionary, but you can look up a lemma. Examples of Lemmatization are that “run” is a base form for words like “running” or “ran” or that the word “better” and “good” are in the same lemma so they are considered the same.

Keyword matching:

Next, we shall define a function for a greeting by the bot, i.e. if a user’s input is a greeting, the bot shall return a greeting response. CT (Cloud Tech) uses a simple keyword-matching for greetings. We will utilize the same concept here.

Generating Response:

Bag of Words:

After the initial preprocessing phase, we need to transform text into a meaningful vector (or array) of numbers. The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

- A vocabulary of known words.
- A measure of the presence of known words.

Why is it is called a “bag” of words? That is because any information about the order or structure of words in the document is discarded and the model is only concerned with whether the known words occur in the document, not where they occur in the document.

The intuition behind the Bag of Words is that documents are similar if they have similar content. Also, we can learn something about the meaning of the document from its content alone.

For example, if our dictionary contains the words {Learning, is, the, not, great}, and we want to vectorize the text “Learning is great”, we would have the following vector: (1, 1, 0, 0, 1).

TF-IDF Approach:

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much “informational content”. Also, it will give more weight to longer documents than shorter documents.

One approach is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like “the” that are also frequent across all documents are penalized. This approach to scoring is called Term Frequency-Inverse Document Frequency, or TF-IDF for short, where:

Term Frequency: is a scoring of the frequency of the word in the current document.

TF = (Number of times term t appears in a document)/(Number of terms in the document)

Inverse Document Frequency: is a scoring of how rare the word is across documents.

IDF = 1+log(N/n),
where, N is the number of documents and n is the number of documents a term t has appeared in.

Cosine Similarity:

TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus.

Cosine Similarity (d1, d2) = Dot product (d1, d2) / ||d1|| * ||d2||
where d1, d2 are two non-zero vectors.

To generate a response from our bot for input questions, the concept of document similarity will be used. We define a function response which searches the user's utterance for one or more known keywords and returns one of several possible responses. If it doesn't find the input matching any of the keywords, it returns a response:" I am sorry! I don't understand you"

Finally, we will feed the lines that we want our bot to say while starting and ending a conversation depending upon the user's input.

3.3 Project code:

Import necessary libraries:

Code:

```
import io
import random
import string # to process standard python strings
import warnings
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import warnings
warnings.filterwarnings('ignore')
```

Downloading and installing NLTK:

Code:

```
pip install nltk
```

Installing NLTK Packages:

Code:

```
import nltk
from nltk.stem import WordNetLemmatizer
nltk.download('popular', quiet=True) # for downloading packages
#nltk.download('punkt') # first-time use only
#nltk.download('wordnet') # first-time use only
```

Reading in the corpus:
Code:

```
f=open('cc.txt','r',errors = 'ignore')
raw=f.read()
raw = raw.lower()# converts to lowercase
```

Tokenisation:
Code:

```
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
print(sent_tokens[:2])
word_tokens = nltk.word_tokenize(raw)# converts to list of words
print(word_tokens[:8])
```

Preprocessing:
Code:

```
lemmer = nltk.stem.WordNetLemmatizer()
#WordNet is a semantically-oriented dictionary of English included in NLTK.
def LemTokens(tokens):
    return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)


def LemNormalize(text):
    return LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

Keyword matching:
Code:

```
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)
GREETING_RESPONSES = ["hi","It's a pleasure to have you here today!", "hey", "hi there",
"hello", "I am glad! You are talking to me"]
def greeting(sentence):

    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

Generating Response

Code:

```
def response(user_response):
    robo_response=''
    sent_tokens.append(user_response)
    TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
    tfidf = TfidfVec.fit_transform(sent_tokens)
    vals = cosine_similarity(tfidf[-1], tfidf)
    idx=vals.argsort()[0][-2]
    flat = vals.flatten()
    flat.sort()
    req_tfidf = flat[-2]
    if(req_tfidf==0):
        robo_response=robo_response+"I am sorry! I don't understand you, try rephrasing your question"
        return robo_response
    else:
        robo_response = robo_response+sent_tokens[idx]
        return robo_response

flag=True
print("CT: My name is Cloud Tech, you can call me CT. I am here to increase your knowledge on cloud computing ^.^\n If you want to exit, type Bye!\n")
while(flag==True):
    user_response = input()
    user_response=user_response.lower()
    if(user_response!='bye'):
        if(user_response=='thanks' or user_response=='thank you' ):
            flag=False
            print("CT: You are welcome..")
        else:
            if(greeting(user_response)!=None):
                print("CT: "+greeting(user_response))
            else:
                print("CT: ",end="")
                print(response(user_response))
                sent_tokens.remove(user_response)
    else:
        flag=False
        print("CT: Bye! take care..")
```

cc.txt (text corpus):

Cloud computing
Cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the Internet, usually in a completely seamless way, Exactly where the hardware and software is located and how it all works doesn't matter to you, the user—it's just somewhere up in the nebulous "cloud" that the Internet represents.

Simple example:
Soundcloud—one of my favorite examples of a website (and mobile app) that uses It to good effect, Musicians and DJs upload their music, which "followers" can listen to (or preview) for free through real-time streaming, You can build up a personal collection of tracks you like and access them from any device, anytime, anywhere, The music you listen to stays up in the cloud: in theory, there is only ever one copy of every music file that's uploaded, Where is the music stored? No-one but Soundcloud needs to know—or care.

Types of cloud computing:
Infrastructure as a Service
Software as a Service
Platform as a Service.

IaaS means you're buying access to raw computing hardware over the Net, such as servers or storage, Since you buy what you need and pay-as-you-go, this is often referred to as utility computing, Ordinary web hosting is a simple example of IaaS: you pay a monthly subscription or a per-megabyte/gigabyte fee to have a hosting company serve up files for your website from their servers.

SaaS means you use a complete application running on someone else's system, Web-based email and Google Documents are perhaps the best-known examples, Zoho is another well-known SaaS provider offering a variety of office applications online.

PaaS means you develop applications using Web-based tools so they run on systems software and hardware provided by another company, So, for example, you might develop your own ecommerce website but have the whole thing, including the shopping cart, checkout, and payment mechanism running on a merchant's server, App Cloud (from salesforce.com) and the Google App Engine are examples of PaaS.

Advantages and disadvantages

Advantages include, allowing you to buy in only the services you want, when you want them, cutting the upfront capital costs of computers and peripherals, You avoid equipment going out of date and other familiar IT problems like ensuring system security and reliability, You can add extra services (or take them away) at a moment's notice as your business needs change, It's really quick and easy to add new applications or services to your business without waiting weeks or months for the new computer (and its software) to arrive.

Drawbacks include, Instant convenience comes at a price, Instead of purchasing computers and software, It means you buy services, so one-off, upfront capital costs become ongoing operating costs instead, That might work out much more expensive in the long-term.

Growth is, The figures speak for themselves: in every IT survey, news report, and pundit's op-ed, It seems the only show in town, Back in 2008, over a decade ago, the Pew Internet project reported that 69 percent of all Internet users had "either stored data online or used a web-based software application" (in other words, by their definition, used some form of It), In 2009, Gartner priced the value of It at $58.6 billion, in 2010 at $68.3 billion, in 2012 at over $102 billion, and in 2017 at $260 billion; its current forecast is for the sector to reach $302.5 billion by 2021, In 2013, management consultants McKinsey and Company forecast It (and related trends like big data, growing mobilization, and the Internet of Things) could have a "collective economic impact" of between $10–20 trillion by 2025, In 2016, Amazon revealed that its AWS offshoot, the world's biggest provider of It, had become a $10 billion-a-year business; by 2019, that figure had leaped to an astonishing $25,7 billion—more than the entire revenue for the giant global McDonald's empire in 2018, The Microsoft Cloud isn't far behind.

Business benefits include, Businesses have shrewder and more interesting reasons for liking the cloud, Instead of depending on Microsoft Office, to give one very concrete example, they can use free, cloud-based open-source alternatives such as Google Docs, So there are obvious cost and practical advantages: you don't have to worry about expensive software licenses or security updates, and your staff can simply and securely share documents across business locations (and work on them just as easily from home), Using It to run applications has a similarly compelling business case: you can buy in as much (or little) computing resource as you need at any given moment, so there's no problem of having to fund expensive infrastructure upfront. If you run something like an ecommerce website on cloud hosting, you can scale it up or down for the holiday season or the sales, just as you need to. Best of all, you don't need a geeky IT department because—beyond commodity computers running open-source web browsers—you don't need IT.

Mobilization means, One of the biggest single drivers of It is the huge shift away from desktop computers to mobile devices, which (historically, at least) had much less processing power onboard, Web-connected smartphones, tablets, Kindles, and other mobile devices are examples of what used to be called "thin clients" or "network computers" because they rely on the servers they're connected to, via the network (usually the Internet), to do most of the work. A related trend referred to as bring your own device (BYOD) reflects the way that many companies now allow their employees to logon to corporate networks or websites using their own laptops, tablets, and smartphones.

From the smartphone in your pocket to the mythical fridge that orders your milk, the number and range of devices connected to the Internet is increasing all the time. A new trend called the Internet of Things anticipates a massive increase in connected devices as everyday objects and things with built-in sensors (home heating controllers, home security webcams, and even parcels in transit) get their own IP addresses and become capable of sending and receiving data to anything or anyone else that's online. That will fuel the demand for It even more.

It makes it possible for cellphones to be smartphones and for tablets to do the sorts of things that we used to do on desktops, but it also encourages us to do more things with those devices—and so on, in a virtuous circle. For example, if you buy a smartphone, you don't simply do things on your phone that you used to do on your PC: you spend more time online overall, using apps and services that you previously wouldn't have used at all. It made mobile devices feasible, so people bought them in large numbers, driving the development of more mobile apps and better mobile devices, and so on.

CHAPTER 4
CONCLUSION AND FUTURE SCOPE

4.1 Conclusion:

Though it is a very simple bot with hardly any cognitive skills, its a good way to get into NLP and get to know about chatbots. Though 'CT (Cloud Tech)' responds to user input, for a production system we will need one of the existing bot platforms or frameworks, but this example should help us think through the design and challenge of creating a chatbot. The Internet is flooded with resources and this mini-project serves as a great introduction to NLP and the analysis of a chatbot.

4.2 Limitations:

Since this bot is retrieval based, the response it gives is greatly dependent on the corpus. It will not perform as good as generative bots, but it still covers all the basics involved in creating a chatbot in python.

4.3 Future scope:

This bot serves as an introduction to making a chatbot from scratch using python and the NLTK library.

There are many existing bot frameworks that can be integrated with the existing code to make the bot respond to complex queries. We can also implement a deep learning algorithm to enable the bot to learn more and more as the user inputs data, thereby enabling the bot the have an actual conversation with people (generative bots).

CHAPTER 5
OUTPUT SCRENSHOTS

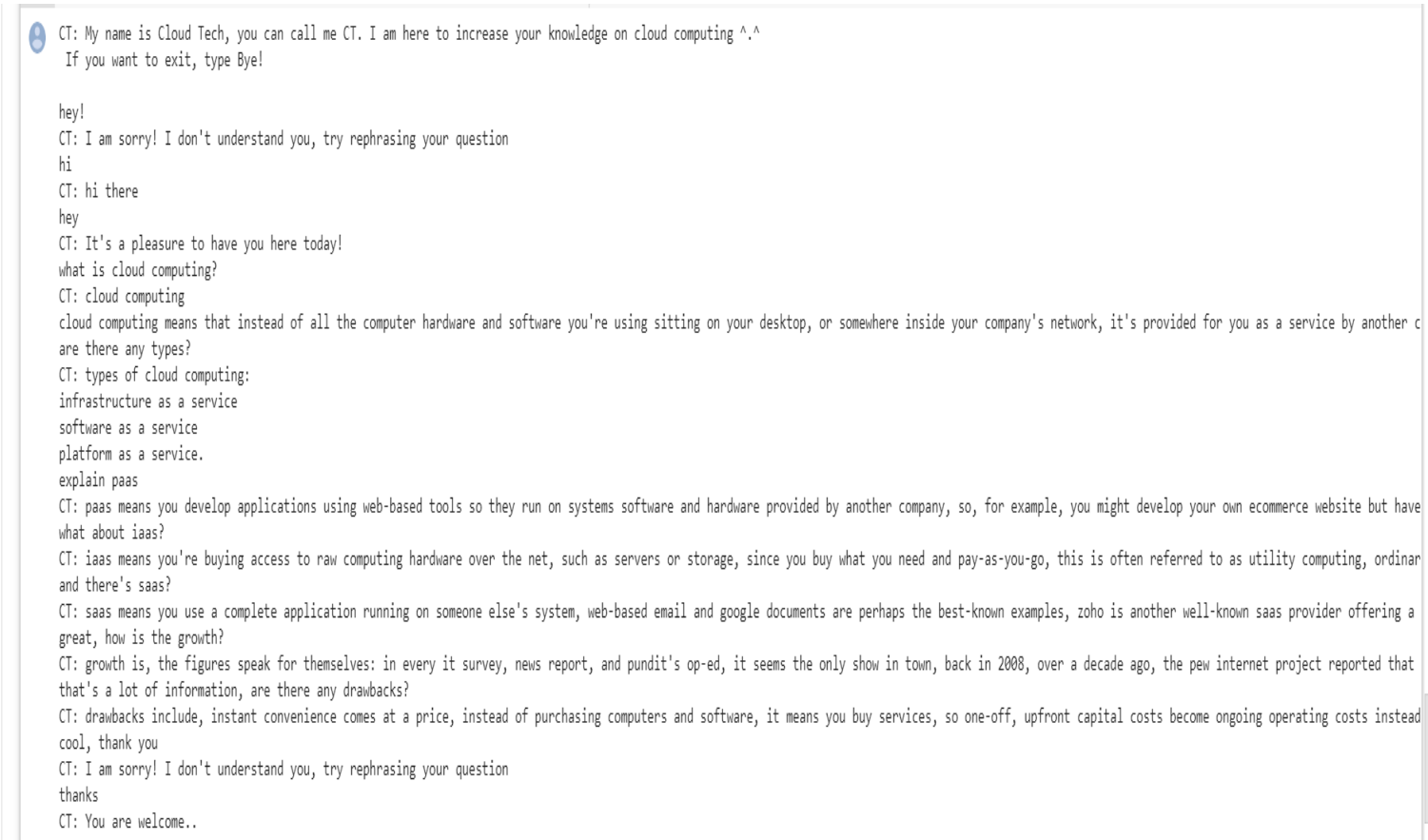
Output conversation:

CT: My name is Cloud Tech, you can call me CT. I am here to increase your knowledge on cloud computing ^.^
If you want to exit, type Bye!

hey!
CT: I am sorry! I don't understand you, try rephrasing your question
hi
CT: hi there
hey
CT: It's a pleasure to have you here today!
what is cloud computing?
CT: cloud computing
cloud computing means that instead of all the computer hardware and software you're using sitting on your desktop, or somewhere inside your company's network, it's provided for you as a service by another company and accessed over the internet, usually in a completely seamless way, exactly where the hardware and software is located and how it all works doesn't matter to you, the user ”it's just somewhere up in the nebulous "cloud" that the internet represents.
are there any types?
CT: types of cloud computing:
infrastructure as a service
software as a service
platform as a service.
explain paas
CT: paas means you develop applications using web-based tools so they run on systems software and hardware provided by another company, so, for example, you might develop your own ecommerce website but have the whole thing, including the shopping cart, checkout, and payment mechanism running on a merchant's server, app cloud (from salesforce.com) and the google app engine are examples of paas.
what about iaas?
CT: iaas means you're buying access to raw computing hardware over the net, such as servers or storage, since you buy what you need and pay-as-you-go, this is often referred to as utility computing, ordinary web hosting is a simple example of iaas: you pay a monthly subscription or a per-megabyte/gigabyte fee to have a hosting company serve up files for your website from their servers.
and there's saas?
CT: saas means you use a complete application running on someone else's system, web-based email and google documents are perhaps the best-known examples, zoho is another well-known saas provider offering a variety of office applications online.
great, how is the growth?
CT: growth is, the figures speak for themselves: in every it survey, news report, and pundit's op-ed, it seems the only show in town, back in 2008, over a decade ago, the pew internet project reported that 69 percent of all internet users had "either stored data online or used a web-based

software application" (in other words, by their definition, used some form of it), in 2009, Gartner priced the value of it at $58.6 billion, in 2010 at $68.3 billion, in 2012 at over $102 billion, and in 2017 at $260 billion; its current forecast is for the sector to reach $302.5 billion by 2021, in 2013, management consultants Mckinzey and company forecast it (and related trends like big data, growing mobilization, and the internet of things) could have a "collective economic impact" of between $10â€“20 trillion by 2025, in 2016, amazon revealed that its aws offshoot, the world's biggest provider of it, had become a $10 billion-a-year business; by 2019, that figure had leaped to an astonishing $25,7 billionâ€”more than the entire revenue for the giant global mcdonald's empire in 2018, the microsoft cloud isn't far behind.
that's a lot of information, are there any drawbacks?
CT: drawbacks include, instant convenience comes at a price, instead of purchasing computers and software, it means you buy services, so one-off, upfront capital costs become ongoing operating costs instead, that might work out much more expensive in the long-term.
cool, thank you
CT: I am sorry! I don't understand you, try rephrasing your question
thanks
CT: You are welcome..

Screenshot:



REFERENCES

[1][online]

https://www.researchgate.net/publication/340678278_A_Smart_Chatbot_Architecture_based_NLP_and_Machine_Learning_for_Health_Care_Assistance

[2][online]

https://ieeexplore.ieee.org/document/8289883

[3][online]

https://www.pluralsight.com/guides/build-a-chatbot-with-python

[4][online]

https://www.zdnet.com/google-amp/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/

.