A
EST Practical Activity Report
Submitted for

ARTIFICIAL INTELLIGENCE (UCS411)

Submitted by:

(102003627) Sarthak Narang

(102003746) Khushant Rana

BE Second Year

Submitted to-

Dr. Seema Wazarkar

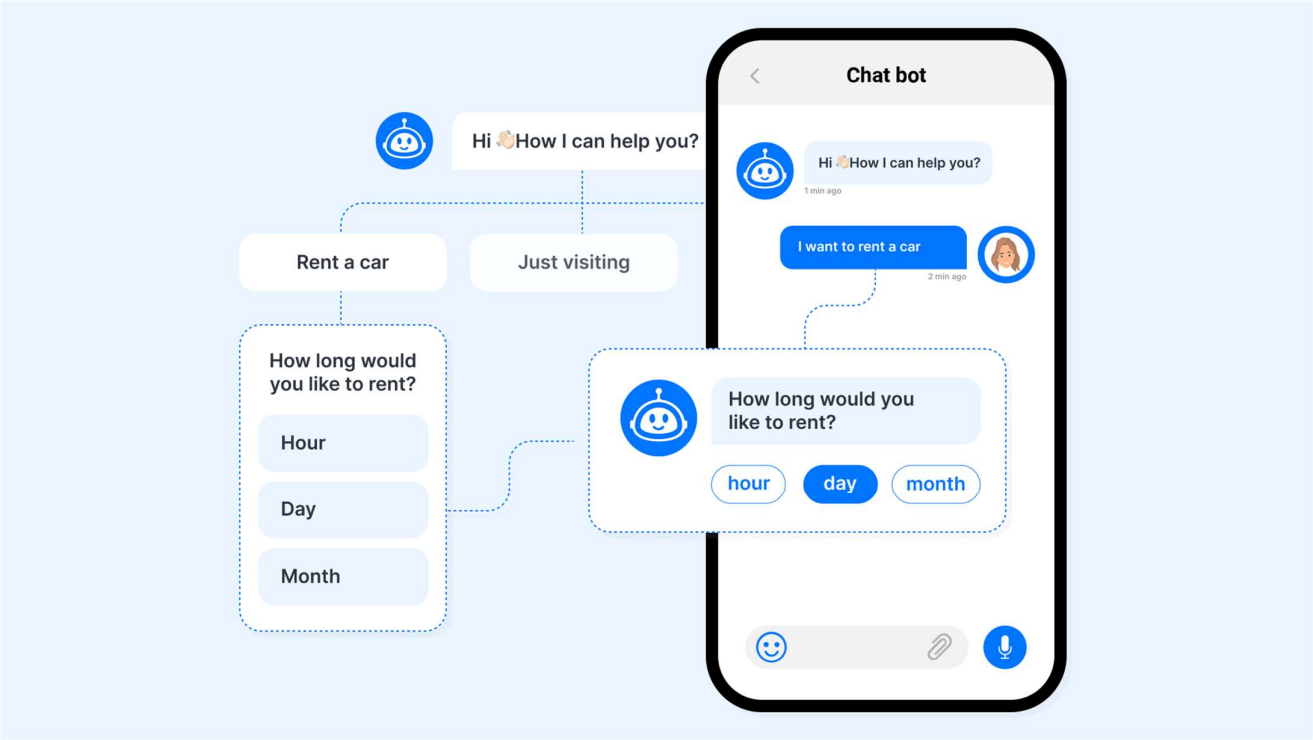
Computer Science and Engineering Department

TIET, Patiala

Jan-June 2022

TABLE OF CONTENTS

SELLER-CHATBOT PROJECT



INTRODUCTION

OBJECTIVE:

The objective of this project is building the live chatbot with help of PyTORCH. It involves the concept of NLP- Natural Language Processing and Deep Learning.

PRE-REQUISITES:

In this project, we require random, json, torch, numpy, nltk module of python. We have to import these modules / files for our project :

Pytorch

NLTK

NumPy

Random

json (for dataset)

tkinter (for GUI)

METHODOLOGY

NATURAL LANGUAGE PROCESSING:

Natural Language Processing (NLP) is one of the most challenging fields in Artificial Intelligence (AI). It comprehends a set of algorithms that allow computers to understand or generate the language used by humans. These algorithms can process and analyze vast amounts of natural language data from different sources (either sound or text) to build models that can understand, classify, or even generate natural language as humans would. Like other fields in AI, NLP has significantly progressed thanks to the advent of Deep Learning (DL), which has resulted in models that can obtain results on par with humans in some tasks.

These advanced NLP techniques are being applied in healthcare. During a typical patient-provider encounter, a conversation ensues where the doctor constructs, through questions and answers, a chronological description of the development of the patient's presenting illness or symptoms. A physician examines the patient and makes clinical decisions to establish a diagnosis and determine a treatment plan. This conversation, and data in the EHR, provide the required information for physicians to generate the clinical documentation, referred to as medical reports.

NATURAL LANGUAGE TOOLKIT:

NLTK 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, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analysing linguistic structure, and more.

NUMPY:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, statistical operations, random simulation and much more.

PYTORCH:

PyTorch is an open source machine learning (ML) framework based on the Python programming language and the Torch library. It is one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment.

PyTorch is similar to NumPy and computes using tensors that are accelerated by graphics processing units (GPU). Tensors are arrays, a type of multidimensional data structure, that can be operated on and manipulated with APIs. The PyTorch framework supports over 200 different mathematical operations.

Key Features of PyTorch:

Some of the key features of PyTorch include:

TorchScript- This is the production environment of PyTorch that enables users to seamlessly transition between modes. TorchScript optimizes functionality, speed, ease-of-use and flexibility.

Dynamic graph computation- This feature allows users to change network behavior on the fly, rather than waiting for the entire code to be executed.

Automatic differentiation- This technique numerically computes the derivative of a function by making backward passes in neural networks.

Python support- Because PyTorch is based on Python, it can be used with popular libraries and packages such as NumPy, SciPy, Numba and Cynthon.

TKINTER:

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

Although Tkinter is considered the de facto Python GUI framework, it's not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you're looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to quickly build something that's functional and cross-platform.

IMPLEMENTATION

1.) First, we have made intents file (**dataset**) in which we have different tags under which we have made different patterns and responses according to the particular tag.

Like for example for tag "goodbye" we have made patterns like

"patterns": ["Bye", "See you later", "Goodbye"],

"responses":
[
"See you later, thanks for visiting",
"Have a nice day",
"Bye! Come back again soon."
]

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi",
        "Hey",
        "How are you",
        "Is anyone there?",
        "Hello",
        "Good day"
      ],
      "responses": [
        "Hey :-)",
        "Hello, thanks for visiting",
        "Hi there, what can I do for you?",
        "Hi there, how can I help?"
      ]
    },
```

2.) Then we made the file(nltkutil) which return the response of words according to the given input.

In this firstly we use the "tokenize" function which split the sentence into array of words using nltk library.

Tokenization: splitting a string into meaningful units
(e.g. words, punctuation characters, numbers)

"what would you do with 1000000$?"

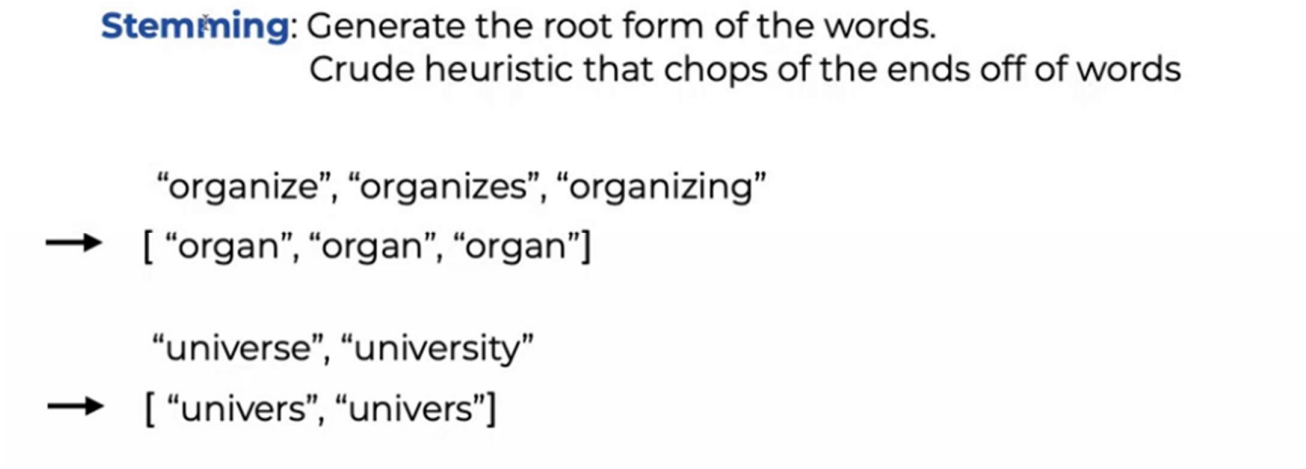
→ ["what", "would", "you", "do", "with", "1000000", "$", "?"]

"aren't you happy with so much money?"

→ ["are", "n't", "you", "happy", "with", "so", "much", "money", "?"]

```
def tokenize(sentence):
    return nltk.word_tokenize(sentence)
```

Then we use the “stem” function which uses the prefix of similar characters in word.




```
def stem(word):
    return stemmer.stem(word.lower())
```

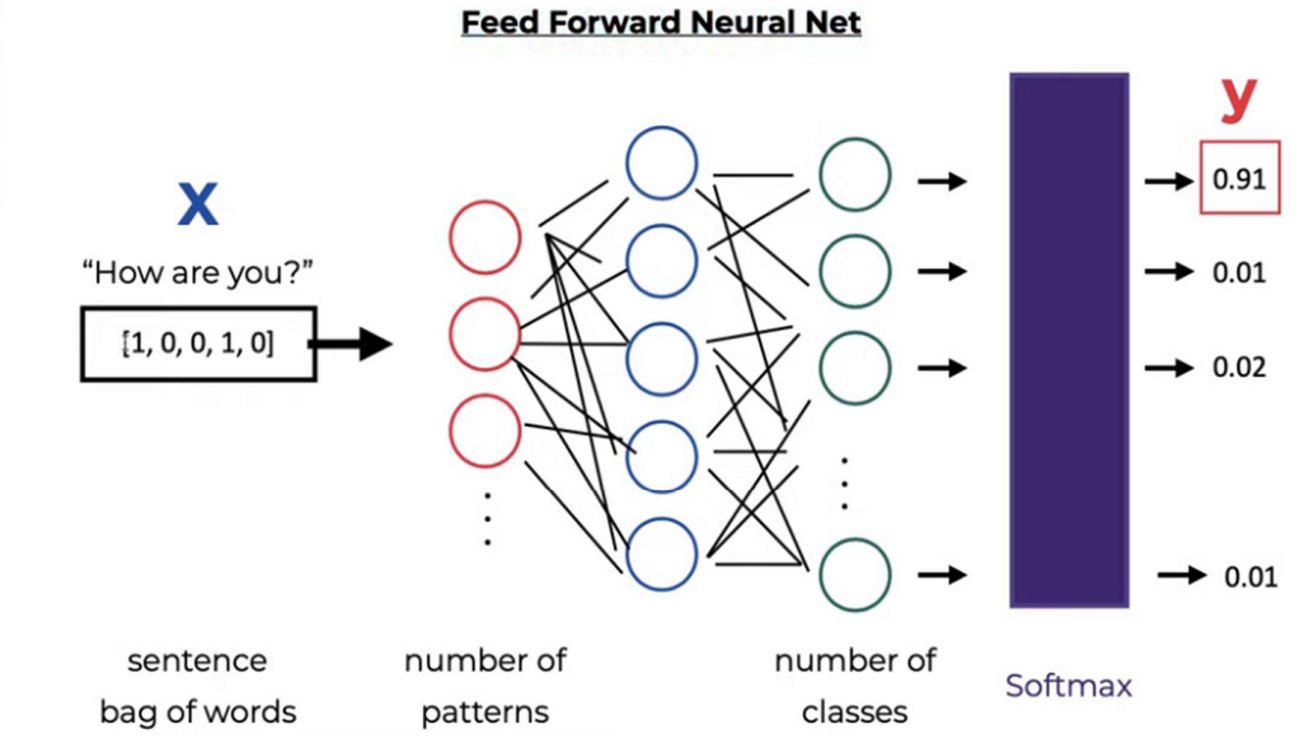
At last, we use the function of bag_of_words which returns the bag of words array 1 for each known word that exist otherwise 0.

```
def bag_of_words(tokenized_sentence, words):
    sentence_words = [stem(word) for word in
tokenized_sentence]
    # initialize bag with 0 for each word
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1

    return bag
```

3.) Then we made another file (model.py) which check the probability of words which is to be used according to the past experience implemented by neural network. In this we use the bag_of_words function which was returned in the previous file. And it will return the output according to given input.

```
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()
```



4.) Then at last we made the trainer file. In this we are trying to improve the response of bot according to experience and probability of words. In this we have objective to minimize the loss and maximize the probability (epoch) of the output from the neural network.

```python
# create training data
X_train = []
y_train = []
for (pattern_sentence, tag) in xy:
    # X: bag of words for each pattern_sentence
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)
    # y: PyTorch CrossEntropyLoss needs only class labels, not one-hot
    label = tags.index(tag)
    y_train.append(label)

X_train = np.array(X_train)
y_train = np.array(y_train)

# Hyper-parameters
num_epochs = 1000
batch_size = 8
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 8
output_size = len(tags)
print(input_size, output_size)
```

"Hi" →	[1,	0,	0,	0,	0,	0 ,	0]	
"How are you?" →	[0,	1,	1,	1,	0,	0 ,	0]	0 (greeting)
"Bye" →	[0,	0,	0,	0,	1,	0 ,	0]	
"See you later" →	[0,	0,	0,	1,	0,	1 ,	1]	1 (goodbye)
		X						y

Our NLP Preprocessing Pipeline

"Is anyone there?"

↓ tokenize

["Is", "anyone", "there", "?"]

↓ lower + stem

["is", "anyon", "there", "?"]

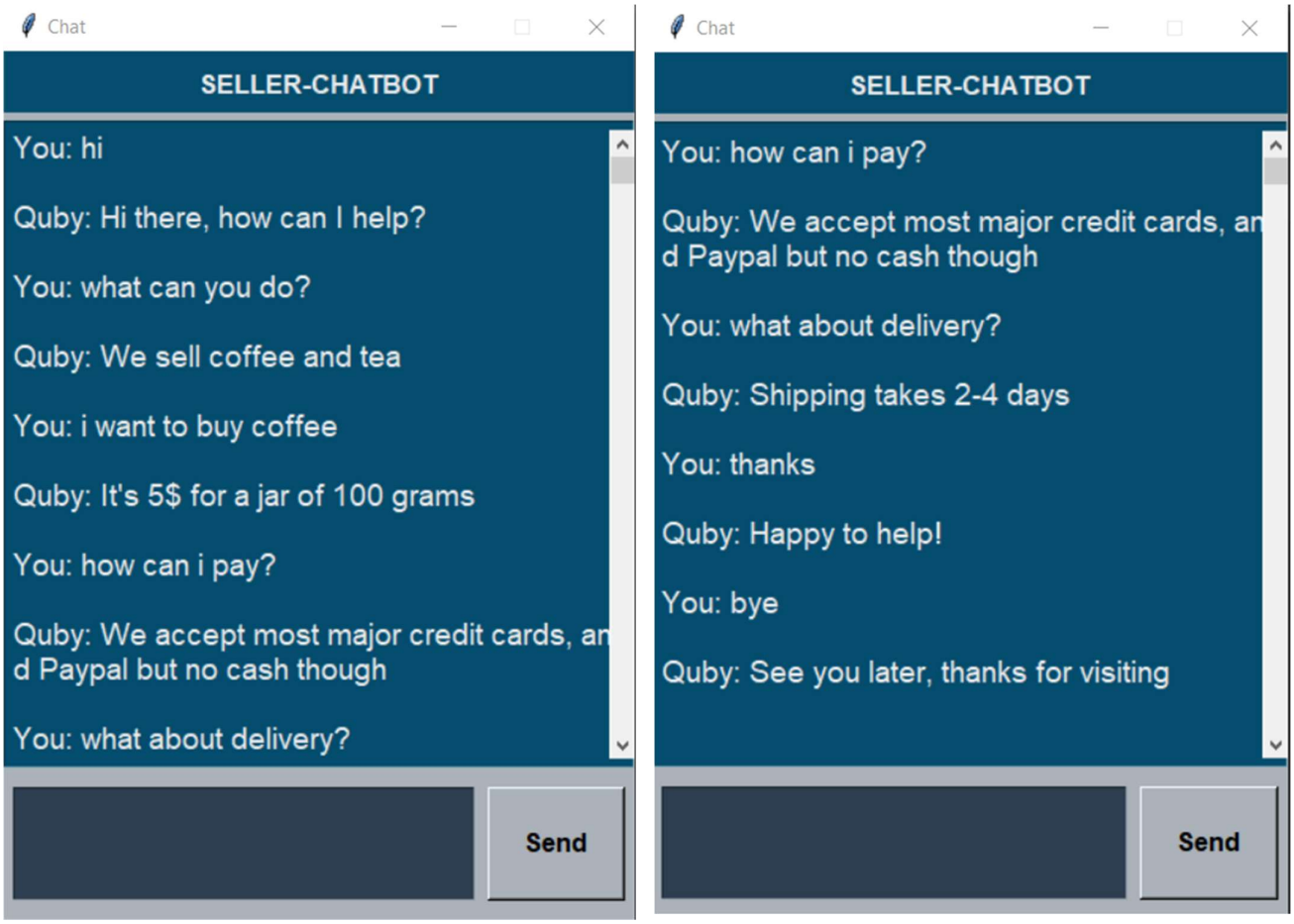
↓ exclude punctuation characters

["is", "anyon", "there"]

↓ bag of words

X [0, 0, 0, 1, 0, 1, 0, 1]

5.) GUI:



In simple terms, UI is the means by which a human and a computer interact. For GUI we uses the Tkinter library of python. Tkinter is the most commonly used method in Python to create GUI (Graphical User Interface) applications. It is a standard Python interface to the Tk GUI toolkit shipped directly with Python. Python with tkinter is the fastest and easiest way to create GUI applications.

CONCLUSION

With a chatbot, your organization can easily offer high-quality support and conflict resolution any time of day, and for a large quantity of customers simultaneously.

According to Microsoft, **90%** of consumers expect an online portal for customer service. As a significant aspect of business evolution, the need for AI-powered chatbots will only continue to rise. Now is the time to deploy a chatbot solution so that your company doesn't get left behind.

RESULT

It has been interesting to investigate how the participants interacted with the chatbot and how they reported on it afterwards. Our findings have some indicators leading towards that a chatbot could be a good alternative for acting as a helpful friend for freshman at industry.