Project Report

of

Pharmaceutical Chatbot using machine learning.

for

Neural Networks

A Project Report is submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

Neha [CO17341]

Under the Guidance of

Dr. Varun Gupta (Professor, CSE Dept.)

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY

(DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University, Chandigarh

Sector-26, Chandigarh. PIN-160019

INDEX:

1. Abstract **4**

2. Chapter 1 **5**

1.1Motivation

1.2Introduction

1.3 Method

3 Chapter 2 **7**

2.1 Numpy

2.2 Pandas

2.3 Keras

2.4 Nltk

2.5 Tkinter

4 Chapter 3 13

3.1 Dataset

5 **Chapter 4** 14

4.1 Code Snippets

4.2 Working

6 Chapter 5 16

5.1 Output

7 Chapter 6 18

6.1 Conclusion

8 Chapter 7 20

7.1References

Abstract:

Chatbots are becoming more and more popular in many industries, including the pharmaceutical and medical industry. Their ability to mimic human conversations through text and auditory means can prove beneficial to the pharmaceutical and healthcare industry. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. They can make life easier for patients, doctors and employees alike.

For example, patients can be supported before and after treatment. Doctors can find out quickly and at any time of the day about the interactions or other properties of drugs, and sales staff can prepare for customer meetings with the help of a bot.

This project is aims at creating a chatbot to mimic human conversation about medical advice and guidance for the one in distress.

Chapter 1

Motivation:

Health systems around the world are being challenged by increasing demand for care of people with COVID-19 while trying to maintain the delivery of routine health services. Provide Access to Medical Information and Treatment Reminders at home easily.

For example, once a patient is prescribed a treatment plan, the chatbot can help the patient adhere to the treatment schedule. It can also inform patients of potential side effects, what to do if they experience side effects, and provide daily reminders on when the treatment should be administered. Customers can use the bot to ask about different queries, user information, product demos, product catalogs, and whatever else they may need.

A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. Chatbots are used a lot in customer interaction, marketing on social network sites, and instant messaging the client.

Introduction:

Once the user's intent has been identified, the chatbot must provide the most appropriate response for the user's request.

As the coronavirus pandemic rapidly sweeps across the world, it is inducing a considerable degree of fear, worry and concern in the population at large and among certain groups in particular, such as older adults, care providers and people with underlying health conditions. As the fear of getting infected deepens so does the fear of getting out of one's house does

contributing to deteriorated medical attention to the ones in need as some of the people are too afraid to step into hospitals full of COVID 19 patients.

As a response to the situation prevailing all over the world it is absolutely necessary to find reliable means for the diagnosis of the needful. This project aims at providing reliable information regarding on topics like drug reaction, nearby pharmacies etc.

Method/ Usage:

The steps to create a chatbot:

1. Import and load the data file
2. Pre-process data
3. Create training and testing data
4. Build the model
5. Predict the response

Chapter 2

Libraries:

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

The advantages of using Numpy are given below as :

1. POWERFUL N-DIMENSIONAL ARRAYS

   Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

2. NUMERICAL COMPUTING TOOLS

   NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

3. INTEROPERABLE

   NumPy supports a wide range of hardware and computing platforms and plays well with distributed, GPU, and sparse array libraries.

4. PERFORMANT

The core of NumPy is a well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

5. EASY TO USE

   NumPy's high-level syntax makes it accessible and productive for programmers from any background or experience level.

6. OPEN SOURCE

   Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

numpy.ravel()

numpy.ravel (*a*, *order='C'*)

Return a contiguous flattened array.

A 1-D array, containing the elements of the input, is returned. A copy is made only if needed.

As of NumPy 1.10, the returned array will have the same type as the input array. (for example, a masked array will be returned for a masked array input)

Pandas:

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-

clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Library features.

- Tools for reading and writing data between in-memory data structures and different file formats.
- DataFrame object for data manipulation with integrated indexing.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and subsetting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation[4] and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- Provides data filtration.

The library is highly optimized for performance, with critical code paths written in Cython or C.

pandas.read_csv

Read a comma-separated values (csv) file into DataFrame.

Also supports optionally iterating or breaking of the file into chunks.

Additional help can be found in the online docs for IO Tools.

Tkinter

Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's *de-facto* standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python. The name *Tkinter* comes from the *Tk interface*. Tkinter was written by Fredrik Lundh. Tkinter is free software released under a Python license

The tkinter package ("Tk interface") is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.)

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Keras:

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is the key to doing good research.

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras empowers engineers and researchers to take full advantage of the scalability and cross-platform capabilities of TensorFlow 2: you can run Keras on TPU or on large clusters of GPUs, and you can export your Keras models to run in the browser or on a mobile device.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

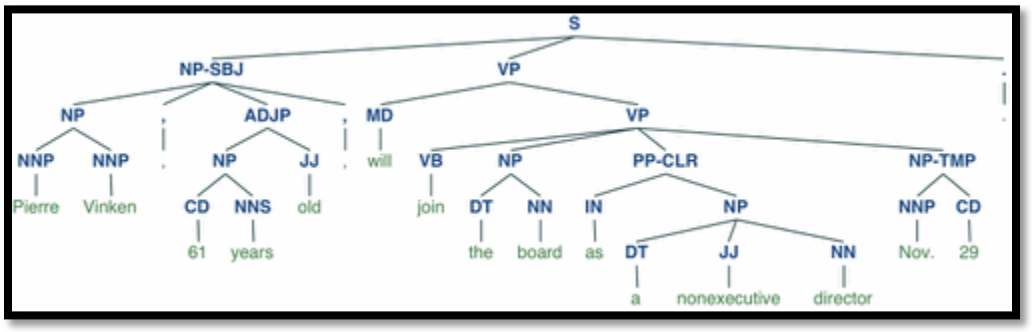
In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

Nltk:

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit,[5] plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.



Chapter 3

Dataset:

https://github.com/ali-ce/datasets/blob/master/Artificial-Intelligence/ChatterbotsDB.csv

What is Overfitting/ Underfitting a Model?

As mentioned, in statistics and machine learning we usually split our data into two subsets: training data and testing data (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data.

Overfitting

Overfitting means that the model we trained has trained "too well" and is now, well, fit too closely to the training dataset. This usually happens when the model is too complex (i.e. too many features/variables compared to the number of observations).

Underfitting

In contrast to overfitting, when a model is underfitted, it means that the model does not fit the training data and therefore misses the trends in the data.

Train/Test Split

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

Chapter 4

Code Snippets:

To read the data from the dataset.

```
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 keras.optimizers import SGD
import random

words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)
```

```
for intent in intents['intents']:
    for pattern in intent['patterns']:

        #tokenize each word
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        #add documents in the corpus
        documents.append((w, intent['tag']))

        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list.

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
```

Working:

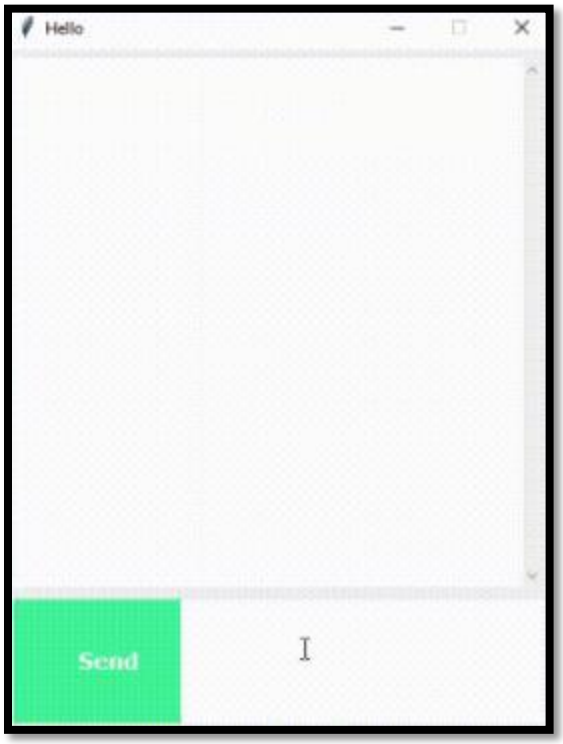
Here are the steps used to create a chatbot:

- Import and load the data file
- Pre-process data
- Create training and testing data
- Build the model
- Predict the response

Chapter 5

Output:

Initial Window:



Disease Predictor predicts diseases based on disease symptoms and shows output based on three classifiers Decision Trees, Random Forest and Naïve Bayes.

User Input:

The problem, situation, topic or problem that you are dealing with at the moment can be described in English language.

Output:

An appropriate response by the chatbot as per the input received from the user.

Secondary Window:


Hello
You: WHat can you do
Bot: I can guide you through Adverse drug r eaction list, Blood pressure tracking, Hospit als and Pharmacies
You: i am looking for some drugs
Bot: Navigating to Adverse drug reaction mo dule
You: thank you
Bot: Happy to help!
You: see ya got to go
Bot: Have a nice day
Send

Chapter 6

Conclusion:

This project is aims at creating a chatbot to mimic human conversation about medical advice and guidance for the one in distress.

Chatbots are becoming more and more popular in many industries, including the pharmaceutical and medical industry. Their ability to mimic human conversations through text and auditory means can prove beneficial to the pharmaceutical and healthcare industry. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. They can make life easier for patients, doctors and employees alike.

Chapter 7

References:

1. https://github.com/rajaskakodkar/CNN-based-Gun-detection/blob/master/BE_Project_Report.pdf
2. https://botfriends.de/en/10-chatbot-beispiele-fuer-die-pharma-und-medizinbranche/
3. http://www.datascienceassn.org/sites/default/files/Natural%20Language%20Processing%20with%20Python.pdf