HEALTHCARE CHATBOT

TEAM NO: 12

TEAM MEMBERS:

ABINUS MERCY A – 2019103501

KAVISHREE S – 2019103537

NIRUTHIYASHRI P – 2019103549

Guided by:

Dr.A.R.ARUNARANI,

Teaching Fellow,

Department of Computer Science and Engineering,

Anna University.

CONTENTS:

INTRODUCTION:

- The famous proverb “Health is Wealth” has not long forgotten. As health is considered the most valuable and precious for every living Today’s people are busy with their works at home, office and more addicted to Internet. They avoid going hospitals for small problems.

- Those small problems may become major problem.To avoid this, we have come up with a chatbot which will help the users to predict their possible disease in an easy way.

- This system application uses question and answer protocol within the style of chatbot to answer user queries.The response to the question is replied supported the user question.The significant keywords are fetched from the sentence through which we can identify symptoms of explicit illness.

- This system is developed to scale back the tending price and time of the users because it is not potential for the users to go to the doctors once in a real time required.

OBJECTIVE:

- ✓ In this project, we create a healthcare chatbot system using Python and NLT that can diagnosis the disease and provide basic information about the disease before consulting a doctor.

- ✓ This real time chat system is simple and interactive. We crafted this system in such a way to detect disease according to the symptoms.

- ✓ This chatbot's true well-being is to promote the general public by offering sufficient advice in terms of a great and balanced life.

RELATED WORKS:

SNO	TITLE,AUTHOR,YEAR	METHODOLOGY	METRICS	MERITS AND DEMERITS	BRIEF NOTE
1	Florence- A Health Care Chatbot: a medical chatbot with self-diagnosis using Artificial Intelligence by Jahnvi Gupta , Vinay Singh , Ish kumar-2021	With RASA framework and natural language processing a chatbot can accurately diagnose patients with the analysis of basic symptoms and conversational approach	Tested among people and result in correct predication of disease on the basis of database created	It will make people aware about how serious the disease they are having and if they need to take action against it.	RASA NLU and RASA CORE is to extract structured information from user messages. This usually includes the user's intent and any entities their message contains.
2	Chatbot for Disease Prediction and Treatment Recommendation using Machine Learning by Rohit Binu Mathew,	A medical chatbot is built to be a conversational agent that motivates users to discuss about their health issues and based	Tested among people and result in correct predication of disease on the basis of database created	Since it is completely personal it helps users to be more open with their health matters and paves way	Text Processing is done using NLP.NLP makes human to communicate with machine easily. NLTK which provides the functionalities required for NLP.

	Sandra Varghese , Sera Elsa Joy,Swanthana Susan Alex-2019	symptoms provided by them ,chatbot returns the diagnosis		for chatbot to efficiently identify the disease.	
3	Influence of word normalization and chi-squared feature selection on Support Vector Machine(SVM) Text classification by Ardy Wibowo Haryanto, Edy Kholid Mawardi, Mulijono-2018	Comparing the performance of SVM using the same dataset with different pre-processing data accompanied by the addition of Chi-squared feature selection in classifying the news.	SVM text classification using stemming enhanced by Chi-square received a precision value of 95% and accuracy of 95.05%. SVM text classification using lemmatization enhanced by Chi-square received a precision value of 93% and accuracy of 93.24%.	Accuracy of text classification.	Lancaster Stemmer for Stemming and WordNet Lemmatizer for lemmatization.
4	Predicting Frequently Asked Questions on the COVID-19 chatbot using Dual Intent and Entity Transformer(DIET)	RASA framework to predict intent and entity of each word related to COVID-19 using the DIET classifier model to provide information	The confidence value of the Intent which is the entity of the question produced an answer with an adequately high value of around 0.979 and for intent value that	It helps people to clarify any doubt about COVID-19.	Dual Intent and Entity Transformer (DIET) with open source RASA 1.8.0 can use trained embeddings from BERT language model in NLU RASA pipeline.

	Classifier by Wistiani Astuti, Desy Pratiwi Ika Putri, Aji Prasetya Wibawa, Yulita Salim, Purnawansyah, Anusua Ghosh-2021	related to COVID-19.	is not entity of the question a moderate or small value of around 0.0002. The results of the taste.core.test for FI-score, Precision, Accuracy have a value of 1.0 for correct answer in accordance with the intent and entity has been used.		
5	The software challenges of building smart chatbots by Gwendal Daniel and Jordi Cabot in 2021.	All bots will be built using Xatkit, an open source chatbot development platform. Continuous integration and testing aspects will be Illustrated with Botium.		Helps to build our own chatbot in an efficient way.	Technical Briefing: 1)Predefined exercises (75%):A series of bots to build where each exercise focuses on a different dimension of the bot building process. 2)Community built bot (25%):An interactive session where attendees will specify a bot to build and start its implementation.

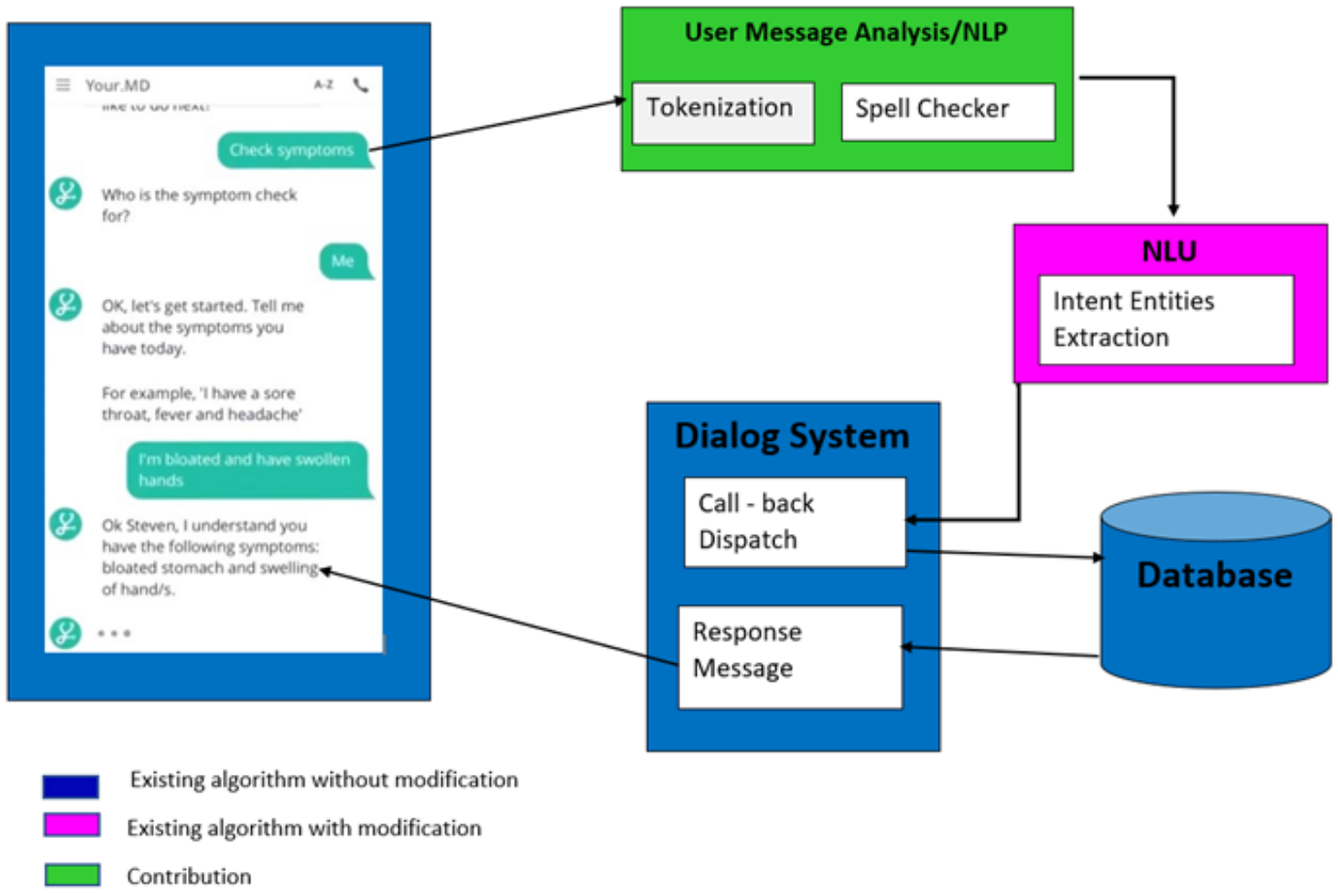
SUMMARY OF ISSUES:

- One of the key concerns in the use of AI chatbots in healthcare is user privacy. The users of such software products might be reluctant to share their personal information with bots.User privacy is a critical issue when it comes to any type of AI implementation, and sharing information about one's medical conditions with a chatbot seems less reliable than sharing the same information with a human doctor.

- AI-enabled chatbots are prone to errors, which eventually leads to patient injury or other significant problems.

- Chatbot using nltk might misunderstood some spellings leading to wrong consultation.

SYSTEM ARCHITECTURE:



STEP 1:

Getting user input.

STEP 2:

After getting user input text preprocessing takes place such as converting to lowercase, tokenization, lemmatization, spellchecker.

STEP 3:

From json file intents and entities are stored as list to train the data.

STEP 4:

Finally the given user input and intents are tested to give a predicted Output(Disease).

MODULES USED:

i. Creating separate lists of all patterns and tags
ii. Creating training data
iii. Creating deep learning model
iv. Cleaning text
v. Creating bag of words
vi. Predicting classes
vii. Getting response and Running chatbot

MODULE 1 - Creating separate lists of all patterns and tags

- We have given input as tags with their respective patterns and responses

  So,First we have to separate them.
- Create a vocabulary of all the words used in the patterns(recall the patterns are the queries posed by the user)
- Create a list of the classes-this is simply the tags of each intent.
- Create a list of all the patterns within the intents file.
- Create a list of all the associated tags to go with each pattern in the intents file.

CODE:

```
# initializing lemmatizer to get stem of words
lemmatizer = WordNetLemmatizer()
# Each list to create
words = []
classes = []
doc_X = []
doc_y = []
f=open('/content/drive/MyDrive/Colab Notebooks/datas.json')
data=json.load(f)
# Loop through all the intents
# tokenize each pattern and append tokens to words, the patterns and
# the associated tag to their associated list
for intent in data["intents"]:
    for pattern in intent["patterns"]:
        tokens = nltk.word_tokenize(pattern)
        words.extend(tokens)
        doc_X.append(pattern)
        doc_y.append(intent["tag"])

    # add the tag to the classes if it's not there already
    if intent["tag"] not in classes:
        classes.append(intent["tag"])
# lemmatize all the words in the vocab and convert them to lowercase
# if the words don't appear in punctuation
words = [lemmatizer.lemmatize(word.lower()) for word in words if word not in string.punctuation]
```

```
# sorting the vocab and classes in alphabetical order and taking the # set to ensure no duplicates occur
words = sorted(set(words))
classes = sorted(set(classes))

DF1=pd.DataFrame(words)
DF1.to_csv("/content/drive/MyDrive/Colab Notebooks/words.csv")
DF2=pd.DataFrame(classes)
DF2.to_csv("/content/drive/MyDrive/Colab Notebooks/classes.csv")
DF3=pd.DataFrame(doc_X)
DF3.to_csv("/content/drive/MyDrive/Colab Notebooks/patterns.csv")
DF4=pd.DataFrame(doc_y)
DF4.to_csv("/content/drive/MyDrive/Colab Notebooks/tags.csv")
```

RESULT:

TAGS

A	B
	0
0	greeting
1	greeting
2	greeting
3	greeting
4	greeting
5	fever
6	fever
7	fever
8	fever
9	fever
10	fever
11	common cold
12	common cold
13	common cold
14	common cold
15	common cold
16	common cold
17	common cold
18	common cold
19	common cold
20	common cold
21	common cold
22	common cold
23	common cold

CLASSES

A	B
	0
0	AIDS
1	Alcoholic hepatitis
2	Allergy
3	Bronchial Asthma
4	Cervical spondylosis
5	Chicken pox
6	Chronic cholestasis
7	Dengue
8	Diabetes
9	Drug reaction
10	Fungal infection
11	GERD
12	Gastroenteritis
13	Hypertension
14	Jaundice
15	Malaria
16	Migraine
17	Paralysis (brain hem
18	Peptic ulcer disease
19	Tuberculosis
20	Typhoid
21	acne
22	arthritis
23	common cold

PATTERNS

A	B
	0
0	Hello
1	How are you?
2	Hi there
3	Hi
4	Whats up
5	high temperaure
6	body pain with head
7	headache
8	cough and sneezing
9	running nose and co
10	fever cold
11	continuous sneezing
12	fatigue
13	cough
14	fever
15	running nose
16	chills
17	cough
18	fever
19	loss of smell
20	muscle pain
21	headache
22	malaise
23	swelled lymph nodes

WORDS

A	B
	0
0	abdomen
1	abdominal
2	abnormal
3	acidity
4	acute
5	adios
6	alcohol
7	altered
8	and
9	anxiety
10	appetite
11	are
12	around
13	back
14	balance
15	behind
16	belly
17	blackhead
18	bladder
19	bleeding
20	blister
21	blood
22	blurred
23	body

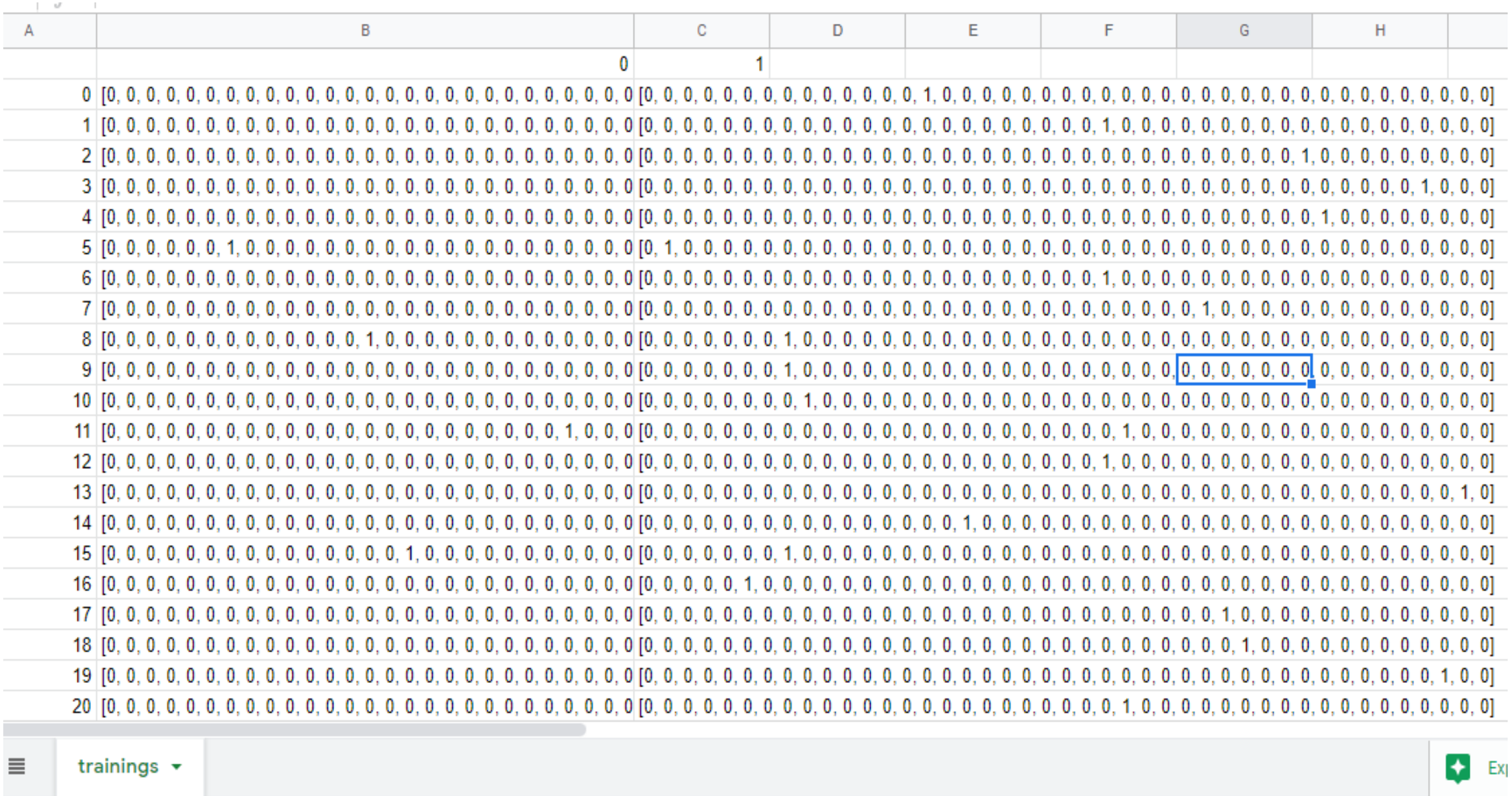
MODULE 2 - **Creating training data**

- Neural Networks expect numerical values, and not words, to be fed into them, therefore, we first have to process our data so that a neural network could read what we are doing.
- In order to convert our data to numerical values, we are going to leverage a technique called bag of words.

CODE:

```
# list for training data
training = []
out_empty = [0] * len(classes)
# creating the bag of words model
for idx, doc in enumerate(doc_X):
    bow = []
    text = lemmatizer.lemmatize(doc.lower())
    for word in words:
        bow.append(1) if word in text else bow.append(0)
    # mark the index of class that the current pattern is associated
    # to
    output_row = list(out_empty)
    output_row[classes.index(doc_y[idx])] = 1
    # add the one hot encoded BoW and associated classes to training
    training.append([bow, output_row])
# shuffle the data and convert it to an array
random.shuffle(training)
training = np.array(training, dtype=object)
# split the features and target labels
train_X = np.array(list(training[:, 0]))
train_y = np.array(list(training[:, 1]))
print(train_X)
print(train_y)
DF=pd.DataFrame(training)
DF.to_csv("/content/drive/MyDrive/Colab Notebooks/trainings.csv")
```

Training data that will be used for prediction



	A	B	C	D	E	F	G	H
		0	1					
0		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
1		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
2		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]					
3		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]					
4		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]					
5		[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
6		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
7		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
8		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
9		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
10		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
11		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]					
12		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]					
13		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]					
14		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
15		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
16		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					
17		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]					
18		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]					
19		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]					
20		[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]					

trainings

Module 3 – Creating deep learning model

- With our data converted into a numerical format, we can now build a Neural Network model that we are going to feed our training data into. The idea is that the model will look at the features and predict the tag associated with the features then will select an appropriate response from that tag.

CODE:

```
# defining some parameters
input_shape = (len(train_X[0]),)
output_shape = len(train_y[0])
epochs = 200
# the deep learning model
model = Sequential()
model.add(Dense(128, input_shape=input_shape, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(output_shape, activation = "softmax"))
adam = tf.keras.optimizers.Adam(learning_rate=0.01, decay=1e-6)
model.compile(loss='categorical_crossentropy',
              optimizer=adam,
              metrics=["accuracy"])
print(model.summary())
model.fit(x=train_X, y=train_y, epochs=200, verbose=1)
```

RESULT:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 128)               27264

 dropout_4 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 64)                8256

 dropout_5 (Dropout)         (None, 64)                0

 dense_8 (Dense)             (None, 43)                2795

=================================================================
Total params: 38,315
Trainable params: 38,315
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/200
10/10 [==============================] - 0s 3ms/step - loss: 3.7719 - accuracy: 0.0223
Epoch 2/200
10/10 [==============================] - 0s 2ms/step - loss: 3.6087 - accuracy: 0.0541
Epoch 3/200
10/10 [==============================] - 0s 2ms/step - loss: 3.4996 - accuracy: 0.0541
Epoch 4/200
10/10 [==============================] - 0s 2ms/step - loss: 3.3352 - accuracy: 0.0955
Epoch 5/200
10/10 [==============================] - 0s 2ms/step - loss: 3.1454 - accuracy: 0.1497
Epoch 6/200
10/10 [==============================] - 0s 2ms/step - loss: 2.9227 - accuracy: 0.1975
Epoch 7/200
10/10 [==============================] - 0s 3ms/step - loss: 2.6350 - accuracy: 0.2611
Epoch 8/200
10/10 [==============================] - 0s 3ms/step - loss: 2.4627 - accuracy: 0.2707
Epoch 9/200
10/10 [==============================] - 0s 2ms/step - loss: 2.4022 - accuracy: 0.2962
```

Module 4 – Cleaning text

- This is about preprocessing the input entered by the user.
- First we are tokenizing the words and then applying lemmatization technique to find the stem of words

CODE:

```
def clean_text(text):
  tokens = nltk.word_tokenize(text)
  tokens = [lemmatizer.lemmatize(word) for word in tokens]
  with open('/content/drive/MyDrive/Colab Notebooks/tokens.csv','a') as file:
        writer=csv.writer(file)
        writer.writerow(tokens)
        file.close()
  return tokens
```

TOKENS.CSV

	A	B	C	D	E	F	G	H	I	J
1	Hi	there								
2	I	have	acidity							
3	I	have	patch	in	throat	and	itching			
4	I	have	visual	disturbance	with	string	headache			
5	I	have	abdominal	paina	nd	vomiting	with	nausea		
6	I	have	red	spot	all	over	body	and	fatigue	
7	I	have	fatigue	cramp						
8	I	have	blister	and	skin	rash				
9	i	have	bladder	discomfort	along	with	joint	pain		
10	i	have	anxiety	and	hip	joint	pain			
11	I	have	pu	filled	pimple	and	skin	rash	with	blackhead
12	thank	you								

Module 5 – Bag of words

- A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms.
- The approach is very simple and flexible, and can be used in a myriad of ways for extracting features from documents.
- A 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.
- Two algorithms are used here:

A)N-gram algorithm

N-gram is a neighboring sequence of n-items from a given sample of text. N-items means we can have two items, three items and so on. So, it is a contiguous sequence of some items.it helped to predicting the next words in a sentence

B)TF_IDF algorithm

This approach to scoring is called Term Frequency – Inverse Document Frequency, or TF-IDF for short, where:

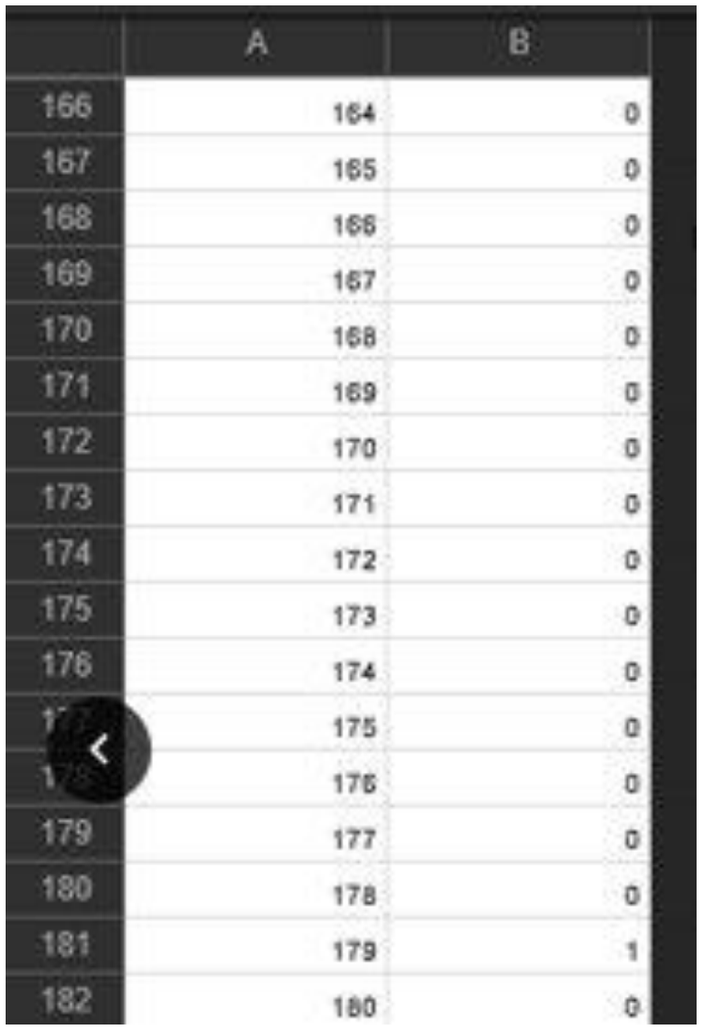
1) Term Frequency: is a scoring of the frequency of the word in the current document.

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

CODE:

```
def bag_of_words(text, vocab):
  tokens = clean_text(text)
  bow = [0] * len(vocab)
  for w in tokens:
    for idx, word in enumerate(vocab):
      if word == w:
        bow[idx] = 1
  DF5=pd.DataFrame(np.array(bow))
  DF5.to_csv("/content/drive/MyDrive/Colab Notebooks/np.csv")
  return np.array(bow)
```

BAG OF WORDS.CSV



	A	B
166	164	0
167	165	0
168	166	0
169	167	0
170	168	0
171	169	0
172	170	0
173	171	0
174	172	0
175	173	0
176	174	0
[illegible]	175	0
[illegible]	176	0
179	177	0
180	178	0
181	179	1
182	180	0

Module 6 – Predicting classes

- **Python predict() function** enables us to **predict the labels of the data values** on the basis of the trained model.

- The predict() function **accepts only a single argument** which is usually the data to be tested.

- It returns the labels of the data passed as argument based upon the learned or trained data obtained from the model.

- Thus, the predict() function works on top of the trained model and makes use of the learned label to map and predict the labels for the data to be tested.

CODE:

```
def pred_class(text, vocab, labels):
  bow = bag_of_words(text, vocab)
  result = model.predict(np.array([bow]))[0]
  thresh = 0.2
  y_pred = [[idx, res] for idx, res in enumerate(result) if res > thresh]

  y_pred.sort(key=lambda x: x[1], reverse=True)
  return_list = []
  for r in y_pred:
    return_list.append(labels[r[0]])
  with open('/content/drive/MyDrive/Colab Notebooks/returnlist.csv','a') as file:
        writer=csv.writer(file)
        writer.writerow(return_list)
        file.close()
  return return_list
```

PREDICTED CLASSES.CSV:

	A	B
1	greeting	
2	GERD	Migraine
3	AIDS	
4	fever	
5	hepatitis E	
6	Chicken pox	Dengue
7	varicose veins	
8	impetigo	
9	urinary tract infection	
10	osteoarthritis	
11	acne	
12	goodbye	

MODULE 7:GETTING RESPONSE AND RUNNING THE CHATBOT

- Create a while loop that allows a user to input some query which is then cleaned, meaning we take the tokens and lemmatize each word. After that, we convert our text to numeric values using our bag of words model and make a prediction of what tag in our intents the features best represent.

- From there, we would take a random response from our responses within that intents tag and use that to respond to the query.

- So when a chatbot is running , it gets the user input and predicts the class where it belongs and then compare them and writes the result as reply of chatbot.

CODE:

```
def get_response(intents_list, intents_json):
  tag = intents_list[0]
  list_of_intents = intents_json["intents"]
  for i in list_of_intents:
    if i["tag"] == tag:
      result = random.choice(i["responses"])
      break
  return result


# running the chatbot
while True:
    message = input("")
    if(message!='exit'):
      intents = pred_class(message, words, classes)
      result = get_response(intents, data)
      with open('/content/drive/MyDrive/Colab Notebooks/result.csv','a') as file:
        writer=csv.writer(file)
        writer.writerow(result)
        file.close()
      print(result)

    else:
      print("Chatbot ended")
      break
```

RESULT.CSV

	A	B	C	D	E	F	G	H	I	J	K	L	
1	H	o	w	d	y		P	a	r	t	n	e	r
2	T	h	e	s	e		a	r	e		t	h	e
3	T	h	e	s	e		a	r	e		t	h	e
4	T	h	e	s	e		a	r	e		t	h	e
5	Y	o	u		m	u	s	t		h	a	v	e
6	y	o	u		m	a	y		h	a	v	e	
7	I	m	p	e	t	i	g	o		i	s		a
8	U	r	i	n	a	r	y		t	r	a	c	t
9	o	s	t	e	o	a	r	t	h	r	i	t	i
10	A	c	n	e		t	r	e	a	t	m	e	n
11	I	t		w	a	s		n	i	c	e		s

The result to respective input of the user is saved into csv file letter by letter in each column including the white space.

This is because when the result is processed it is read and printed letter by letter in the chatbot which results in this.

RESULT AND DISCUSSION:

On giving the following symptoms, these results were obtained.

```
Hi there
Howdy Partner!
I have acidity
These are the signs of GERD.Stronger medication may be required
I have patches in throat and itching
These are the symptoms of AIDS.There is no cure.But you can protect yourself and others from                    infection.
I have visual disturbances with string headache
It might be cold due to climatic changes
I have abdominal paina nd vomiting with nausea
These are the signs of hepatitis E.Treatment focuses on rehydration and rest.
I have red spots all over body and fatigue
You must have Chicken pox.Antiviral medication is required.
I have fatigue cramps
you may have varicose veins.Treatment involves compressions stockings and exercise
I have blister and skin rash
Impetigo is a highly contagious skin infection that causes red sores on the face.Antibiotics shorten the infection and can help prevent spread to others.
i have bladder discomfort along with joint pain
Urinary tract infection!! Drink plenty of water
i have anxiety and hip joint pain
osteoarthritis can be treated by physical exercise ,weight loss,ice packs.
I have pus filled pimples and skin rash with blackheads
Acne treatments include over-the-counter creams and cleanser, as well as prescription antibiotics.
thank you
It was nice speaking to you.Type exit to end this!
exit
Chatbot ended
```

CONCLUSION AND SUMMARY:

The task is pretty straightforward and it can be addressed using various techniques. But as it is the case with most of the machine learning projects, the biggest issues was the lack of training data. We tried a whole variety of things from basic techniques like a bag of words. The intent classifier needs to be as accurate as possible because the response of the bot largely depends on the output of the intent classifier. Here we used NLU pipeline with the DIET classifier.

The DIET classifier model in RASA framework predicts answers on the chatbots related to the symptoms given by the user which uses around 200 epochs. The DIET classifier model can predict each word or sentence from the chatbot for all the intents used.

To conclude, the machine learning model we designed is trained in such a way to response for the user input by DIET classifier of intent and entity extraction.

REFERENCES:

[1]. S. Divya, Indumathi, S. Ishwarya, M. Priyasankari, S.Kalpanadevi | A Self-Diagnosis Medical Chatbot Using Artificial Intelligence | Institute of Electrical and Electronics Engineers June 2019

[2]. Rohit Binu Mathew, Sandra Varghese, Sera Elsa Joy, Swanthana Susan Alex | Published 2019 | Computer Science - 3rd International Conference on Trends in Electronics and Informatics (ICOEI)

[3]. D. Madhu, C. Jain, Elmy Sebastain, Shinoy Shaji, A. Ajayakumar| Published 2017- Medicine International Conference on Inventive Communication and Computational Technologies (ICICCT)

[4]. H. Anandakumar and K. Umamaheswari, " A bio-inspired swarm intelligence technique for social aware cognitive radio handovers," Computers & Electrical Engineering, vol. 71, pp. 925-937, Oct. 2018. doi:10.1016/j.compeleceng.2017.09.016

[5]. S. Anil Kumar, C. Vamsi Krishna, P. Nikhila Reddy, B. Rohith Kumar Reddy, I. Jeena Jacob. (2020) | Self-Diagnosing Health Care Chatbot using Machine Learning | International Journal of Advanced Science and Technology, 29(05), 9323-9330

[6]. Shifa Ghare, Sabreen Shaikh, Tasmia Bano Shaikh and HabibFakih Awab | Self-Diagnosis Medical Chat-Bot Using Artificial Intelligence || EasyChair Preprint no. 2736

[7]. Shifa Ghare, Sabreen Shaikh, Tasmia Bano Shaikh and HabibFakih Awab | Self-Diagnosis Medical Chat-Bot Using Artificial Intelligence || EasyChair Preprint no. 2736.

[8].Paper: https://ieeexplore.ieee.org/document/9442006

[9].DIET classifier reference: https://medium.com/the-research-nest/using-the-diet-classifier-for-intent-classification-in-dialogue-489c76e62804