VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

MEDICO YOUR MEDICAL COMPANION

By: -

20BCE2923: Bisuraj Sharma

20BCE2913: Sarthak Giri

20BKT0056: Shubhlaxh porwal

20BCE2423: Naman Mohan Paul

20BCE2132: Ashmit Patel

Submitted to: -

Professor DILIPKUMAR S.

Index

Acknowledgement

We would like to thank our professor DILIPKUMAR S. for giving us this opportunity to perform this project and guiding us about how to build and develop the project and ideology. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

Also, we are very grateful to Vellore Institute of Technology for this wonderful opportunity. It was a great leaning experience and we plan to take our idea further and develop it as real- world project.

Abstract

Healthcare is very important to lead a good life. However, it is very difficult to obtain the consultation with the doctor for every health problem. The idea is to create a medical chatbot using Artificial Intelligence that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot. The chatbots are computer programs that use natural language to interact with users. The chatbot stores the data in the database to identify the sentence keywords and to make a query decision and answer the question. Ranking and sentence similarity calculation is performed using n-gram, TFIDF and cosine similarity. The score will be obtained for each sentence from the given input sentence and more similar sentences will be obtained for the query given. The third party, the expert program, handles the question presented to the bot that is not understood or is not present in the database.

Introduction

- A Chatbot is a system that can interact with human users with natural language. The vast amount of information that is available on the internet allows Chatbots to provide accurate and efficient information based on the user's requirements.

  These systems can learn themselves and restore their knowledge using human assistance or using web resources. This application is incredibly fundamental since knowledge is stored in advance. Rather than a question and answer approach, our model uses the intelligent state of the art ML model and NLP model to classify the patterns and intents and to give the desired response.

Chatbot has been widely used in many ways, including in providing entertainment, education, tourism, and so on. Chatbot can be used as a tool for learning new languages, tools for accessing information systems, tools for visualizing corpus content, and tools for answering questions on certain domains and can be trained in different languages. In the banking sector, several studies were conducted to develop chatbots in providing bank information. One chatbot was developed in banking using natural language processing The dataset used is the FAQ data obtained from a banking website in India. The application uses rule-based and pattern-based techniques where NLP is used to process user queries. Chatbot is also used in other fields, such as providing tutors for counseling services , modular knowledge services ,providers of humor , and so on. In the implementation of chatbot, so that the system can respond to user queries more dynamically, the use of natural language processing plays an important role, namely in understanding user queries in natural languages. Therefore, it is necessary to use an algorithm to find query proximity with patterns in the database, such as the cosine similarity algorithm. Many cosine similarity algorithms are used to find the value of proximity between documents . It's just that this algorithm does not consider the position of tokens in the sentence so that patterns with different sentence structures can have the same cosine value.

Objective

The idea is to create a medical chatbot that can diagnose the disease and provide basic details about the disease before consulting a doctor. This will help to reduce healthcare costs and improve accessibility to medical knowledge through medical chatbot.

Rather than using rule-based bots, we are training the bot to intelligently understand the context using NLP, ML and DL and convert the natural language query to database query and then retrieve the result.

1. Predict the disease based on the symptoms inputted by user.

2. Gives the proper medication based on the symptoms

3. User can give the details of injuries and bot can predict the status using CNN.

4. Suggest the physician whom the patient can consult in case of major symptoms.

Related work and Literature Review

Medical Chatbots – Use Cases, Examples and Case Studies of Conversational AI in Medicine and Health

This paper gives insight into the present medical chatbots and their implementations. The author has focused on 6 medical chatbots that were considered to be innovative and which showed successful implementation though only to an extent.

These 6 chatbots included Your MD, NINA (askNestle), Ada,Mage,

The author also gave view of what the future might hold for this field. He was clear that even though the industry already was filled with various healthcare chatbots, none of them had experimentation with more evolved use cases during development. This limitation rose due to the long way Conversational AI still has left to cover in order to achieve success in this area.

He also highlighted that the accuracy of these chatbots will be sure to rise in the near future as the technologies advance with various experimentations.

The Top 5 Health Chatbots: -

The Medical Futurist

This paper also brings forth the various chatbots and their implementation practices that each one has used. This allows us to learn from the various ways chatbots have been implemented and choose the best option. It also touches on the topic of what place does these chatbots have in healthcare. They have highlighted the chatbots that have ceased to exist and how helpful chatbots can be in medical. The current chatbots that have been discussed in this paper are OneRemission, Youper, Babylon Health, Florence, Healthily, Ada Health, Sensely, Buoy Health, Infermedica and GYANT. All of these chatsbots and their functions helped us decided which one to focus on and develop.

Automated Medical Chatbot :

SSRN Electronic Journal – January 2017

It starts with giving an insight into the history of medical chatbot implementation techniques and the difference proposed in their model. Being the first technical paper that explains how the internal process of their chatbot works, this is one of the most important papers that formed the base of our project. The working of their project though is easy to understand yet very crucial for the development process.

Their main goal/basis while building a Chatbot was the need for it to be natural at responding to the user messages, thus it needed to have sustainable back-end logic to process user inputs and parameters to generate results.

This allowed the chatbot to capture every messages provided by the user as the chatbot enginer gets activated when a user starts interacting with it. Their Chatbot intended to use the AI/ML approach to reply to the user messages and to get the input that it can feed to the engine. The engine accepts initial symptoms and extracts keywords from the data. Using the keywords extracted from the symptom, the Chatbot engine shortlists some of the most likely illnesses that the user may be suffering through by matching the keywords with the disease tags. Once the engine has shortlisted diseases that the user may have, it starts to narrows down the selection to only one disease. For this, it sorts the list of selected diseases according to the most number of matches with the keywords and tags. The engine checks for top 3 symptoms from each of the shortlisted (sorted) selection until all the 3 symptoms matches with the user and it can know that the user has that particular disease. If any two or more disease qualifies for the same, the user is connected directly to the doctor. The engine identifies the disease and Chatbot asks questions to the user related to the symptoms that commonly happens in that disease. The engine can measure the seriousness of the problem by assigning a predetermined threshold value against every disease. Every symptom also has a seriousness score against it. The engine maintains an Integer variable where it sums up the scores of the symptoms if it matches with the user input. If the score hits a value greater than or equal to the threshold level, the Chatbot would connect the user with a doctor and provide temporary tips and medication until the doctor is available to chat. The engine also maintains two String Arrays during the chat session named 'Medication' and 'Remedies'. For every symptom that matches with the user input, the corresponding solution gets stored in the respective arrays. When the Chatbot has finished checking for all the symptoms it would then provide the user with all the Medication and Remedies it has found during the session.

The build the engine using xml and used tokenization, sorting, scores and various other techniques to finally implement their chatbot. They also provided a detailed comparison that allowed us to figure out the requirements a chatbot needs to satisfy.

HealthCare Chatbot : -

International Journal of Creative Research Thoughts(IJCRT)

This paper also touched on the architectural design and algorithms that can be used for classifying and predicting the diseases using a decision tree classifier.

The methodology of this paper was also something that helped us develop our project.

The Decision Tree algorithm belonging to the family of supervised learning algorithms was used to solve both regression and classification problems. The Decision trees used the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree. If the value is less than threshold, then go to the left child node. If the value is more than threshold, then go to the right child node. Then the disease was selected, after which appending the details of the disease and necessary precautions that were needed to be taken was done. If user is having that particular symptom for more than 13 days then, chatbot will respond as "You should take the consultation from doctor". If it is less than 13 days then chat bot will respond as, "It might not be that bad but you should take precautions". After predicting the disease, the bot will give necessary precautions that one must take. The bot will also give a basic description about the disease, as the user will get an idea of what disease that user might be facing. Bot will respond to everything in voice, for which they have used pyttsx3. (pyttsx3 is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline and is compatible with both Python 2 and 3).

Chatbot for Healthcare System Using Artificial Intelligence: -

International Conference on Reliability, Infocom Technologies and Optimization

With similar application of NLP concepts this paper also highlights how the chatbot can be developed and implemented. This paper was able to give us an insight into how the selection of the disease can be done after the keyword is taken out of the conversation with another human being. The algorithm and formula used form the basis for our selection process of the type of disease which is the most crucial part of working of the chatbot. The main motive of the paper was to help the users regarding minor health information. Initially when the user's visits the website first registers themselves and later can ask the bot their queries. The system uses an expert system to answer the queries if the answer is not present in the database. Here the domain experts also should register themselves by giving various details. The data of the chatbot stored in the database in the form of pattern-template. Here SQL is used for handling the database. The proposed architecture came with the use of n-gram model which expanded n-gram models with

variable length arrangements. The n-gram was used for text compression or reduce the data space in the document, to extract the relevant keywords from the database.

HealthCare Chatbot System using Artificial Intelligence: -

International Journal of Creative Research Thoughts(IJCRT)

The authors have given a detailed description about their methodology as well as the limitations and advantages a medical chatbot system brings. Our main focus has been to improve on the given limitation that this paper talks about. The methodology used was: -The health-Care Chat Bot System was written in Python and run Google conversation platform Google Dialogue flow, GUI hyperlinks and an easy, reachable community API. The machine ought to offer a potential parallel operation and machine layout have to now no longer introduce scalability problems in regard to the quantity of floor computers, drugs or presentations linked at anybody time. The stop machine has to additionally permit for seamless recuperation, without facts loss, from person tool failure. There ought to be a sturdy audit chain with all machines moves logged. While interfaces are really well worth noting that this machine is probable to comply to what is to be had. With that during mind, the maximum adaptable and transportable technology have to be used for the implementation. The machine has criticality into this point as it is far a stay machine. If the machine is down, then clients ought to now no longer note, or note that the machine recovers quickly (seconds). The machine ought to be dependable sufficient to run, crash and glitch loose extra or much less indefinitely, or facilitate blunders recuperation sturdy sufficient such that system faults are by no means discovered to its stop-customers.

Proposed Methodology

In our proposed system the user can chat with the bot regarding the query through text. The system uses an expert system to answer the queries. This system can be used by the multiple users to get the counselling sessions online. The data of the chatbot stored in the database in the form of pattern-template. Bot will provide analgesics and food suggestions that means which food you have to take based on the disease. The bot is connected to the cloud and the database, hence the user query can be stored and doctor can later perceive the symptoms when he is free. For our work we used 3 methods to create the bot, Neural Networks, Cosine distance similarity and the Support Vector Machine. The accuracy was higher in cosine distance similarity, hence we deployed the model over the web.

Cosine Similarity Algorithm

Cosine similarity finds a similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. This technique is also used to measure cohesion among clusters within the field of information (data) mining. Cosine similarity=AB/|A||B|. Cosine distance is nothing however getting distance between two vectors in n dimension area. Distance represent however words associated with one another.

Neural Networks

Inspired by the human brain, neural networks consist of layers of interconnected artificial neurons which communicate with one another. These neurons learn features from the data and work together to produce a meaningful output. Neural networks are data intensive and require huge volumes of data to learn patterns and trends in the data. To test this algorithm, we should determine whether the chatbot generates valid response to inputs, keeps the conversation flowing, addresses the need of the user and whether the chatbot is able to mimic the linguistic characteristics of a human to a reasonable extent. This may mean an adaptation of the Turing test may be an appropriate testing method.

Knowledge Base Representation

The data needs to be processed before it can be used to respond to user queries. It is also intended to increase system performance because the system does not need to pre-process the data against patterns in the database.

Preprocessing with

NLP conducted includes:

1. Tokenisation

User queries will be identified and broken down into tokens. At this stage, punctuation

such as dots, commas, question marks will be omitted.

2. Slang Word Checking

This will be Implementation where any slang word in English can be checked and converted in to proper English.

3. Morphology Checking

Tokens are analyzed morphologically to get the basic words and types of words.

Morphological examination is carried out by removing the prefix, suffix, prefixes, and

repetitions as shown in Figure 1 to get the basic word from each token.

4.User Query Processing The process through which the query is passed until the response is seen is shown in Figure .

The query is carried out by preprocessing first, then a pattern search is performed which is most similar to the query using the cosine similarity and parse tree algorithms.

Flowchart

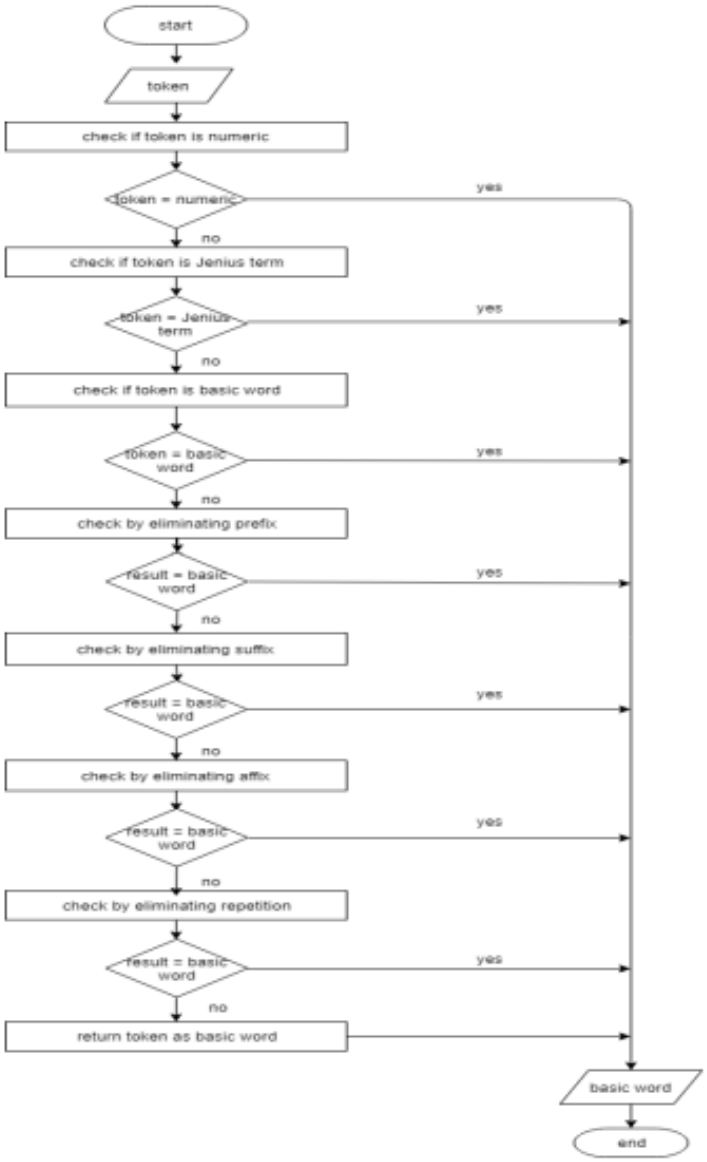


Fig 1. Morphology Checking

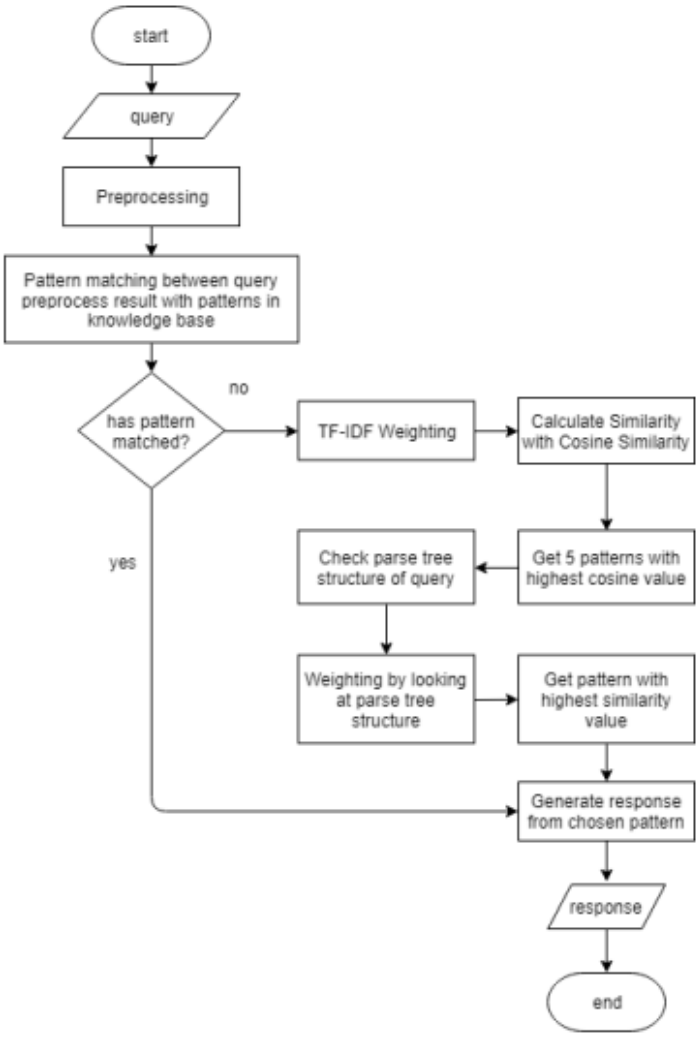


Fig 2. User Query processing Flow

How it Works ?

1. Preprocessing Stage

The preprocessing stages for user query is the same as in the knowledge base preparation stage, the user query will be performed: tokenisation, slang word checking and morphological checking. For example, the following query: user: what medication we should take for this kind of fever ?

It will follow the tokenization process to: fever ,medicine , take. These tokens will be checked in the dictionary keywords and omitted or replaced with a more standard word if it is a slang word. After that the basic word and type of words are searched through checking morphology. From this preprocess, preprocess results are obtained: fever (type of disease) , Medicine (Recommend a Medicine) take (Dosage of Medicine)

2.Pattern Matching.

The results of the query preprocessing are then matched with the patterns stored in the knowledge base. If the exact pattern found with the results of the pre-process query is found, the system will generate a response according to that pattern. But if there is no similar pattern, then the process is followed by a search pattern that is most similar to the cosine similarity algorithm.

3.Similarity Pattern Seacrh with Cosine Similarity

In this project, matching the pattern similarity using TF-IDF weighting and Cosine Similarity. In this process, patterns containing tokens that are present in the preprocess results will be included in the TF-IDF weighting process. All tokens that are in the preprocess results and patterns in the knowledge base will be calculated TF and IDF for each token by following equations (1) and (2) as follows

$$idf_i = \log(\frac{N}{df_i}) \quad (1)$$

Notes:
idf_i = Inverse Document Frequency for term-*i*
N = the number of documents to compare
df_i = the number of documents containings term-*i*

$$w_{i,j} = tf_{i,j} \times idf_i \quad (2)$$

Notes:
$w_{i,j}$ = documents widghting term-*i* in document-*j*
$tf_{i,j}$ = the number of frequency term-*i* in document-*j*
idf_i = Inverse Document Frequency for term-*i*

Taking Example

Query = “My body Temperature is 102 degree”

It will be compared to knowledge base first

K1= Body temperature greater than 98 might be fever

K2= Higher body temperature require quick medication

K3= Higher body temperature might be harmful

The result of weighting TF-IDF like Table 1 below:

Token	Query	K1	K2	K3	df	Log(n/df)	Query	K1	K2
Fever	1	1	1	1		0.0969	0.387	0.22815	

The process is carried out on other tokens / terms in the same way. After obtaining the weight of each term in the pattern, then the value of proximity between the patterns is calculated using the Cosine Similarity algorithm according to equation (3) below:

$$\cos(x, y) = \frac{x.y}{\|x\| \, \|y\|} \qquad (3)$$

Where . indicates the vector dot produc, $x.y = \sum_{k=1}^{n} x_k \, y_k$, and $\|x\|$ is the length of vector $x, x = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x.x}$

So for the results of weighting TF-IDF as in table 1, the Cosine Similarity value can be calculated between document Q with K1, K2, K3, K4 by calculating the length of the vector of each document as follows:

Table 2 Example of Vector Length Calculation Results

Vector length				
Q	K1	K2	K3	K4
0.8455	0.9979	1.0177	1.0700	0.2608

DATASET

Creating Dataset

In [4]:
```
dataset = pd.DataFrame(columns=['title', 'patterns', 'responses'])
for i in data:
    intents = i['tag']
    for t, r in zip(i['patterns'], i['responses']):
        row = {'title': intents, 'patterns': t, 'responses':r}
        dataset = dataset.append(row, ignore_index=True)
dataset
```

Out[4]:

	title	patterns	responses
0	greetings	hello	hello
1	greetings	hey	hey!
2	greetings	hi	what can i do for you?
3	goodbye	cya	have a nice day
4	goodbye	see you later	goodbye
...	...	...	...
293	fever prevention	What medicines can I buy to help me with my fe...	medicines you can consume : acetaminophen ,ibu...
294	diabetes prevention	What medicines can I buy to help me with my di...	medicines you can consume : Insulin ,Amylinomi...
295	depression prevention	What medicines can I buy to help me with my de...	medicines you can consume : brexpiprazole, qu...
296	asthma prevention	What medicines can I buy to help me with my as...	medicines you can consume : epinephrine,antich...
297	Consultation	who should i contact for consultation?	You can contact various doctors here for any k...

298 rows × 3 columns

Cosine Distance for Similarity of Texts

Data frame conversion

```
"intents": [{
        "tag": "greetings",
        "patterns": ["hello", "hey", "hi", "good day", "greetings", "what's up?", "how is it going"],
        "responses": ["hello", "hey!", "what can i do for you?"]
    },
    {
        "tag": "goodbye",
        "patterns": ["cya", "see you later", "goodbye", "have a good day", "bye", "cao", "see ya"],
        "responses": ["have a nice day", "goodbye"]
    },
    {
        "tag": "age",
        "patterns": ["how old", "how old are you?", "what is your age", "how old are you", "age?"],
        "responses": ["I get reborn after every compilation", "hey!", "my owners are averagely 20 years!"]
    },
    {
        "tag": "name",
        "patterns": ["what is your name", "what should i call you", "what's your name?", "who are you?", "can you tell m
        name"],
        "responses": ["you can call me Medbot!", "i am Medbot!", "i am Medbot your medical assistant"]
    },
    {
        "tag": "common cold symptoms",
        "patterns": ["Runny or stuffy nose",
            "Sore throat",
            "Cough",
            "Congestion",
            "Slight body aches or a mild headache",
            "Sneezing",
```

Snippet of the dataset. It is of the form tag, patterns and responses.

The Cosine distance similarity compares the cosine value of the user inputted natural query with all the patterns available in the dataset, the maximum cosine value is given as the result and the corresponding response from that field is given output.

Cosine Distance for Similarity of Texts

In [5]:
```
def cosine_distance_countvectorizer_method(s1, s2):

    # sentences to list
    allsentences = [s1 , s2]

    # packages
    from sklearn.feature_extraction.text import CountVectorizer
    from scipy.spatial import distance


    vectorizer = CountVectorizer()
    all_sentences_to_vector = vectorizer.fit_transform(allsentences)
    text_to_vector_v1 = all_sentences_to_vector.toarray()[0].tolist()
    text_to_vector_v2 = all_sentences_to_vector.toarray()[1].tolist()

#     print(all_sentences_to_vector)
#     print(text_to_vector_v1)
#     print(text_to_vector_v2)


    cosine = distance.cosine(text_to_vector_v1, text_to_vector_v2)
    return round((1-cosine),2)
```

Cosine distance computation

```
In [ ]: while True:
            text = str(input("BOT: "))
            if text.lower() == "exit":
                print("Response: Exiting.....")
                break
            print("Response:",respond(text))
```

```
BOT: hi
Response: what can i do for you?
BOT: who can i visit for consultation
Response: You can contact various doctors here for any kind of consultation: 1.
https://www.1mg.com/online-doctor-consultation, 2. https://www.tatahealth.com/o
nline-doctor-consultation/general-physician, 3. https://www.doconline.com/, or
you can pay a visit to your local area doctor or family doctor.
BOT: i am having chills and coughing
Response: It seem that you are suffering from Asthma
```

Code for NLP

```
import numpy as np

import pandas as pd

import os

import torch

import torch.nn as nn


import nltk

import numpy as np

import pandas as pd

nltk.download('punkt')

from nltk.stem.porter import PorterStemmer


import json,urllib

from torch.utils.data import Dataset, DataLoader


import random
```

```
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()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)

        return out
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()

    def forward(self, x):
        out = self.l1(x)
```

```
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax
        return out
stemmer = PorterStemmer()

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

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

def bag_of_words(tokenized_sentence, all_words):
    """
    sentence = ["hello, "how", "are", "you"]
    words = ["hi", "hello", "I", "you", "bye", "thank", "cool"]
    bag =   [0,     1,       0,   1,    0,      0,       0 ]
    """
    tokenized_sentence = [stem(w) for w in tokenized_sentence]

    bag = np.zeros(len(all_words), dtype = np.float32)
    for idx, w in enumerate(all_words):
        if w in tokenized_sentence:
            bag[idx] = 1.0
```

```
    return bag
import json
with open(url, 'r') as f:
    intents = json.load(f)



intents
with open(url, 'r') as f:
    intents = json.load(f)



all_words = []
tags = []
xy = []
for intent in intents['intents']:
    tag = intent['tag']
    tags.append(tag)
    for pattern in intent['patterns']:
        w = tokenize(pattern)
        all_words.extend(w)
        xy.append((w, tag))

ignore_words = ['?','!','.',',']
all_words = [stem(w) for w in all_words if w not in ignore_words]
all_words = sorted(set(all_words))
tags  = sorted(set(tags))
```

```
X_train  = []
y_train = []
for (pattern_sentence, tag) in xy:
    bag = bag_of_words(pattern_sentence, all_words)
    X_train.append(bag)

    label = tags.index(tag)
    y_train.append(label) # CrossEntropyLoss

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

class ChatDataset(Dataset):
    def __init__(self):
        self.n_samples = len(X_train)
        self.x_data = X_train
        self.y_data = y_train

    # dataset[idx]
    def __getitem__(self, index):
        return self.x_data[index], self.y_data[index]

    def __len__(self):
        return self.n_samples
```

```
# Hyper paramters
batch_size = 15
hidden_size = 15
output_size = len(tags)
input_size = len(X_train[0])
learning_rate = 0.001
num_epochs = 1000



dataset = ChatDataset()
train_loader = DataLoader(dataset = dataset, batch_size = batch_size,
                          shuffle =True, num_workers=0)


device  = torch.device('cpu')


model = NeuralNet(input_size, hidden_size, output_size)


# loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(), lr = learning_rate)


for epoch in range(num_epochs):
    for (words, labels) in train_loader:
        words = words.to(device)
        labels = labels.to(device)
        # Forward
```

```
        outputs = model(words)

        loss = criterion(outputs, labels)

        # backward and optimizer step
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch +1) % 100 == 0:
        # f'epoch {epoch+1}/{num_epochs}, loss={loss.item():.4f}'
        print("epoch {}/{}, loss={:.4f}.".format(epoch+1,num_epochs,loss.item()))

# print(f'Final loss, loss={loss.item():.4f}')
print("Final Loss, loss{:.4f}".format(loss.item()))

data = {
    "model_state":model.state_dict(),
    "input_size":input_size,
    "output_size":output_size,
    "hiddent_size": hidden_size,
    "all_words":all_words,
    "tags": tags
}
FILE = "data.pth"
torch.save(data, FILE)
```

```
print("Traning complete. file saved to",FILE)
device  = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)

with open(url,'r') as f:
  intents = json.load(f)



FILE = "data.pth"
data = torch.load(FILE)

input_size = data['input_size']
hidden_size = data['hiddent_size']
output_size = data['output_size']
all_words = data['all_words']
tags = data['tags']
model_state = data['model_state']

model = NeuralNet(input_size, hidden_size, output_size).to(device)
model.load_state_dict(model_state)
model.eval()
device  = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(device)

with open(url,'r') as f:
```

```
  intents = json.load(f)



FILE = "data.pth"
data = torch.load(FILE)

input_size = data['input_size']
hidden_size = data['hiddent_size']
output_size = data['output_size']
all_words = data['all_words']
tags = data['tags']
model_state = data['model_state']

model = NeuralNet(input_size, hidden_size, output_size).to(device)
model.load_state_dict(model_state)
model.eval()
bot_name = "Bisu'sbut"

print("Namste !!! and welcome to bisu's bot")
while True:
  sentence = input('You: ')

  for word in sentence.split():
    if word.isdigit():
      word = int(word)
      pf = list(df['Office Name'][df['Pincode'] == word ])
```

```
    if len(pf) == 0:
      print("Sorry to inform you, We don't deliver here :(")
      break
    sentence = callme()
    # sentence = input('You: ')

  # else:
    # print("Please Enter Pincode !!!")

if sentence =='quit':
  break

sentence = tokenize(sentence)
X = bag_of_words(sentence, all_words)
X = X.reshape(1, X.shape[0])
X = torch.from_numpy(X)

output = model(X)
_, predicted = torch.max(output, dim=1)
tag = tags[predicted.item()]

probs = torch.softmax(output, dim=1)
prob = probs[0][predicted.item()]



if prob.item() >0.75:
```

```
    for intent in intents["intents"]:
      if tag == intent["tag"]:
        print(bot_name, ":",random.choice(intent['responses']))
        # print("{}:{}".format(bot_name,random.choice(intent['responses'])))

   else:
    print("{} I do not understand...contact on WhatsApp 1234
```

Code For Cosine Distancing

```
# ChatBot using the Given Data



import numpy as np
import pandas as pd



import warnings
warnings.filterwarnings('ignore')
import json
with open('intents.json') as json_file:
    data = json.load(json_file)
data = data['intents']
dataset = pd.DataFrame(columns=['title', 'patterns', 'responses'])
for i in data:
    intents = i['tag']
    for t, r in zip(i['patterns'], i['responses']):
        row = {'title': intents, 'patterns': t, 'responses':r}
```

```
        dataset = dataset.append(row, ignore_index=True)
    dataset
    def cosine_distance_countvectorizer_method(s1, s2):

        # sentences to list
        allsentences = [s1 , s2]

        # packages
        from sklearn.feature_extraction.text import CountVectorizer
        from scipy.spatial import distance


        vectorizer = CountVectorizer()
        all_sentences_to_vector = vectorizer.fit_transform(allsentences)
        text_to_vector_v1 = all_sentences_to_vector.toarray()[0].tolist()
        text_to_vector_v2 = all_sentences_to_vector.toarray()[1].tolist()

    #     print(all_sentences_to_vector)
    #     print(text_to_vector_v1)
    #     print(text_to_vector_v2)


        cosine = distance.cosine(text_to_vector_v1, text_to_vector_v2)
        return round((1-cosine),2)
    def respond(text):
        maximum = float('-inf')
```

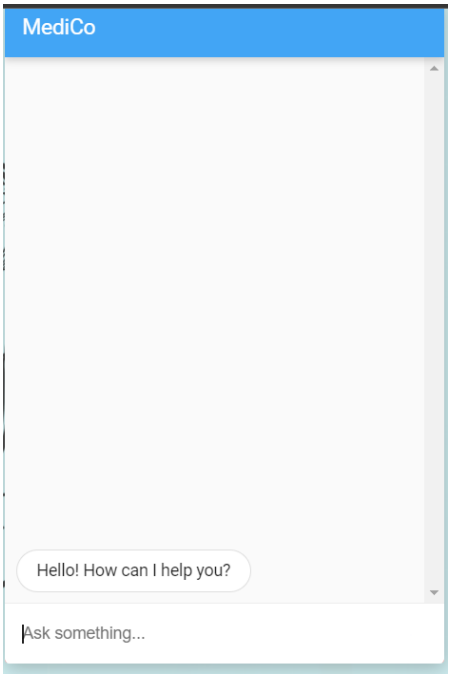
```
    response = ""
    closest = ""
    for i in dataset.iterrows():

        sim = cosine_distance_countvectorizer_method(text, i[1]['patterns'])



        if sim > maximum:
            maximum = sim
            response = i[1]['responses']
            closest = i[1]['patterns']
    return response
while True:
    text = str(input("BOT: "))
    if text.lower() == "exit":
        print("Response: Exiting.....")
        break
    print("Response:",respond(text))
```

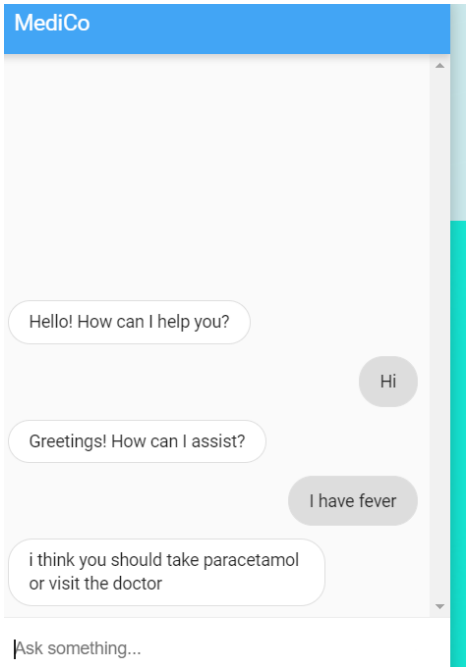
```
BOT: hi
Response: what can i do for you?
BOT: who can i visit for consultation
Response: You can contact various doctors here for any kind of consultation:
1. https://www.1mg.com/online-doctor-consultation, 2. https://www.tatahealth.
com/online-doctor-consultation/general-physician, 3. https://www.doconline.co
m/, or you can pay a visit to your local area doctor or family doctor.
BOT: i am having chills and coughing
Response: It seem that you are suffering from Asthma
```

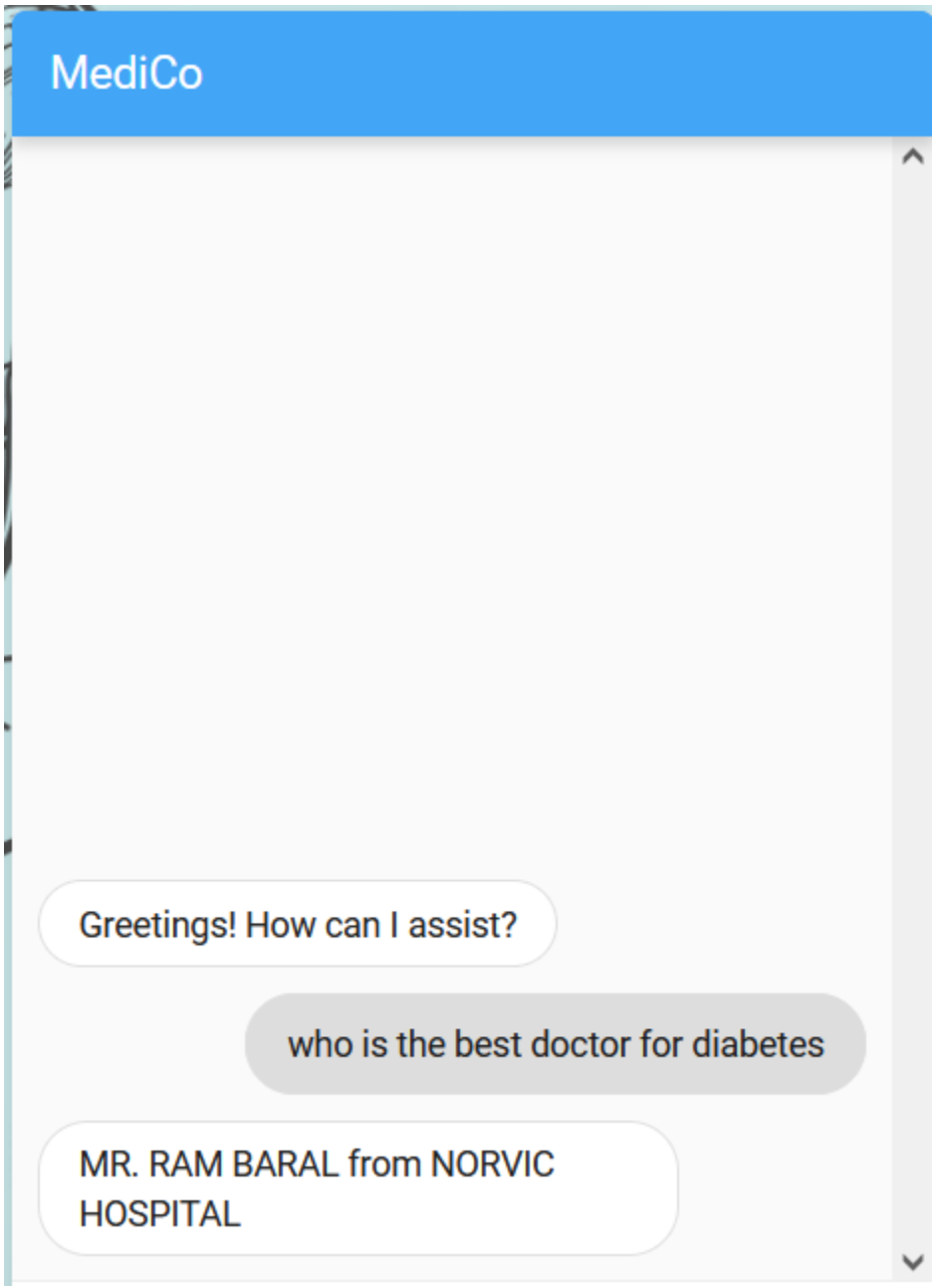
Deployed model over the web

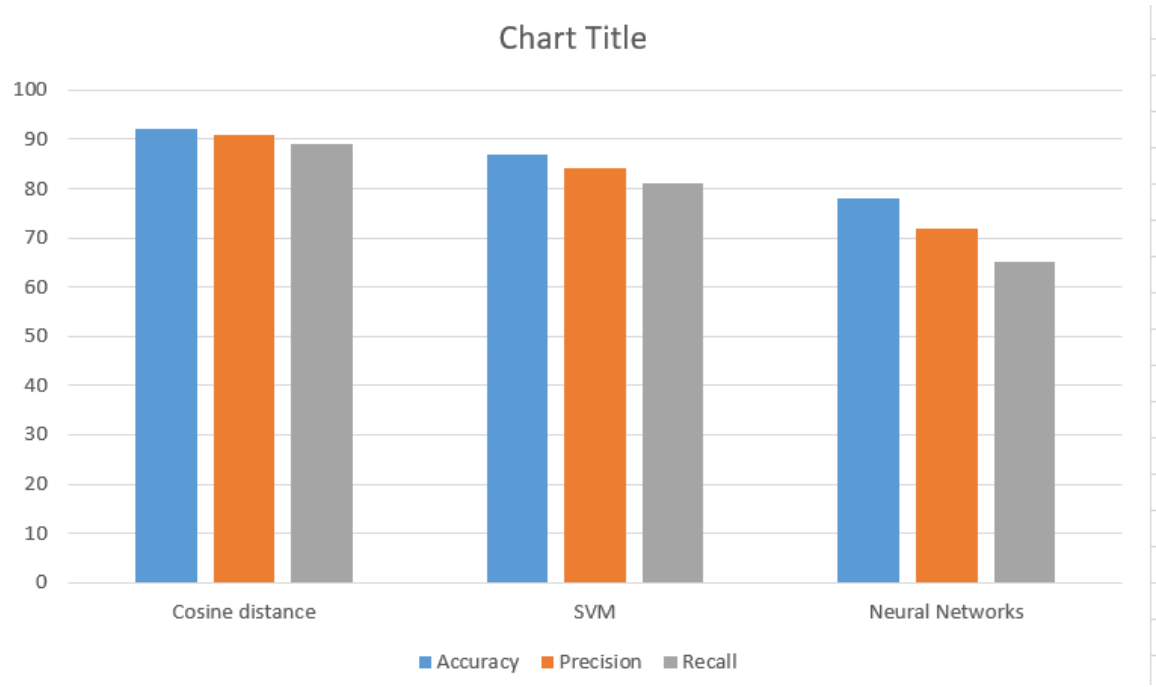
Implementation and Design(Frontend)

Design:






MediCo
Greetings! How can I assist?
who is the best doctor for diabetes
MR. RAM BARAL from NORVIC HOSPITAL

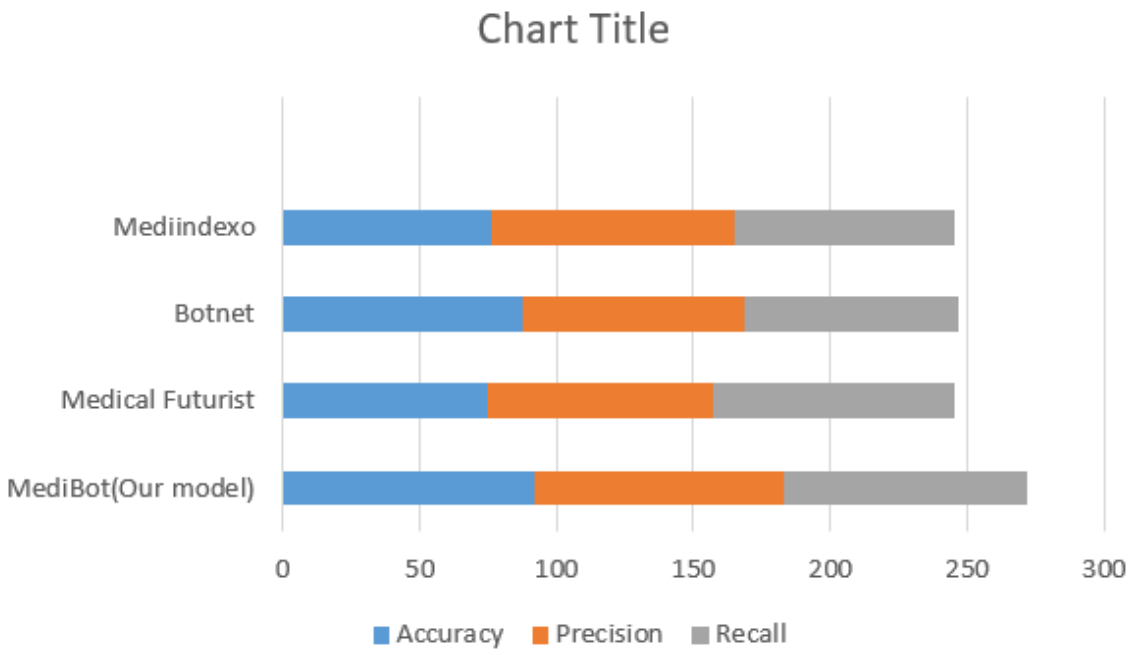


Comparison of various models that we used.

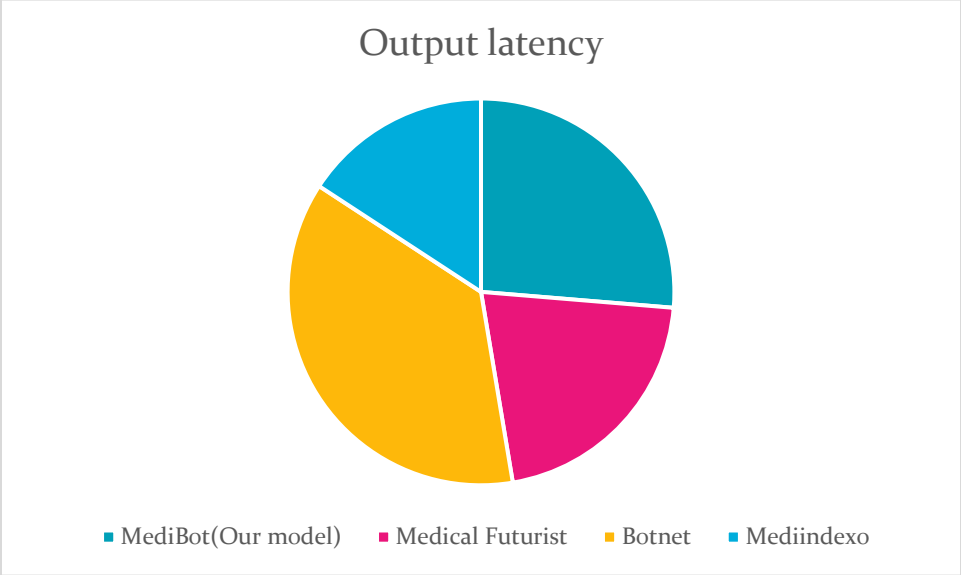
Cosine distance is fairly ahead of other in almost all the matrices. And further for these type of sensitive information, a false positive would be more safer than false negative classification.

WHY OUR MODEL IS BETTER?

GRAPH SPEAKS



OUR MODEL STANDS APPART IN ALMOST MANY METRICES



Latency graph. Our model is little slower than mediindexo bot better than other two, with output delay of 1.5 s

Conclusion

Based on the research and test results, it can be concluded that the chatbot that was developed was able to provide a response to user queries with a response rate of 92%. The response mismatch of 8% is caused by the limited number of medical dataset and a combination of patterns with different sentence structures so that some queries produce the same value of closeness patterns, as well as answers that are expected to be unavailable. The system responds to each user query on average less than 20 milliseconds and displays a response of about 1-3 seconds after adding the waiting time depending on the response length.

We were successful in implementing our version of the chatbot with the UI that enabled us to easily detect and recommend precautions and medicine for the disease/illness that the user had as per the responses given by the user. The improvements that we have proposed have also shown how the chatbots architecture can be improved in order to provide better care for those in need. For the future, as the technology and concepts develop we are sure to see a boom of ai chatbots for various purposes that makes the day to day life of humans easy.

Here we developed the application using the N-gram, TF-IDF for extracting the keyword from the user query. Each keyword is weighed down to obtain the proper answer for the query. The Web-interface is developed for the users, to the input query. The application is improved with the security and effectiveness upgrades by ensuring user protection and characters and retrieving answers consequently for the questions.

Future Work

Most of the existing chatbots are text based, where the response is given on the basis of the text processing and classification. We are aiming to implement a chatbot that gives the response on image entry too. The state of the art CNN model combined with LSTM and RNN can help us achieve it. For eg: when a user inputs his X-RAY report, the chatbot can give the result of the image, and also the image can be stored in the database, for the doctor to review when available. Our chatbot fails to understand and communicate longer by understanding the previous messages. Thus a sequential model with time stamps is what we are trying to implement in the future.

References

[1] C.S. Kulkarni, A.U. Bhavsar, S.R. Pingale, and S.S. Kumbhar, “BANK CHAT BOT – An Intelligent Assistant System Using NLP and Machine Learning,” IRJET (International Research Journal of Engineering and Technology, vol: 04 Issue: 05 | May -2017 [Online]. Available:https://www.irjet.net/archives/V4/i5/IRJET-V4I5611.pdf [Accessed: 9-Jan-2018]

[2] S.H. Aparaj, C. Shashank, and R. Bhagat, “Smart Bot - A Virtual Help Desk Chat Bot,” ICAICTE2013 (International Conference on Advanced Information Communication Technology in Engineering, 2013.

[3] A. Dole, H. Sansare, R. Harekar, and S. Athalye, “Intelligent Chat Bot for Banking System,” IJETTCS (International Journal of Emerging Trends & Technology in Computer Science, vol 4, Issue 5(2), September - October 2015 [Online]. Available: http://www.ijettcs.org/Volume4Issue5(2)/IJETTCS-2015-10-09-16.pdf [Accessed: 20- Apr-2018]

[4] R. Shah, S. Lahoti, and Lavanya, “An Intelligence Chat-bot using Natural Langauge Processing,” International Journal of Engineering Research, vol no. 6, Issue no. 5, pp: 281-286, 2017.

[5] A. Parab, S. Palkar, S. Maurya, and S. Balpande, “An International Career Counselling Bot: A System for Counselling,” IRJET (International Journal of Engineering and Technology, vol: 04, Issue: 03 | Mar – 2017 [Online]. Available: https://www.irjet.net/archives/V4/i3/IRJET-V4I3604.pdf [Accessed: 1-Dec-2017]

[6] B. Setiaji, E. Utami, and H.A. Fatta, “Membangun Chatbot Berbasis AIML dengan Arsitektur Berbasis Modular,” Seminar Nasional teknologi Informasi dan Multimedia 2013 STIMIK AMIKOM Yogyakarta, 2013.

[7] A. Augello, G. Saccone, and S. Gaglio, “Humorist Bot: Bringing Computational Humour in a Chat-Bot System,” International Conference on Complex, Intelligent and Software Intensive Systems, 2018 [Online]. Available: https://ieeexplore.ieee.org/document/4606756 [Accessed: 1-Jun-2018]

[8] V.R. Prasetyo, and E. Winarko, “Rating of Indonesian Sinetron Based on Public Opinion In Twitter Using Cosine Similarity,” 2 nd ICST (International Conference on Science and Technology-Computer, 2016 [Online]. Available: