REPORT FOR:

Lab Oriented Project (CS F367)

TOPIC:

Interactive Android Application for First-Hand Medical Assistance

Submitted By:

ADITYA GARG	2015B3A70618P
HITESH SAGTANI	2015B3A70655P

Supervised By:

Dr. Atanendu Sekhar Mandal
IC Design Group
CEERI Pilani

Acknowledgement

We would like to thank Dr. Atanendu Sekhar Mandal, Scientist 'G', IC Design and Perception Engineering, for his continual guidance throughout the project. We would also like to thank the CSIS Department, BITS-Pilani who gave us a chance to work in collaboration with CEERI.

Contents

I. ABSTRACT

The tasks of medical experts are becoming increasingly difficult with increasing population and consequently diseases. The people from rural area are at even greater difficulties as there is lack of availability of medical experts too. With increasing population, and low medical support when compared to rise in population, especially in developing countries, there is a need to find good medical expertise in these conditions. This problem can be solved by developing an automated medical expert system which when given symptoms and conditions can diagnose a disease. Common diseases like seasonal viral fever, malaria, etc. are becoming more dangerous especially in rural areas where medical support is poor and reachability is less. This project aims at developing an interactive system to provide a provisional diagnosis directly to the patients where there is lack of availability of doctors and physicians for one of the diseases – malaria, dengue or typhoid. It takes as input common natural utterances and outputs the best possible action to take, without any external help.

II. INTRODUCTION

Computer scientists are researching human interactive medical assistance models. Due to less number of experts in specific domain of medical diagnosis it is important to help doctors by assisting them in case frequently occurring diseases. Diagnosing malaria, typhoid, seasonal fevers and cold etc. using interactive medical systems helps doctors with the load of work. It could also play important role in rural areas where the medical facilities and reachability of medical experts is tough. A medical assistance mobile application also helps to maintain health records of individuals which can be further used for extracting information. The first step of diagnosis process consists of patient describing his symptoms to doctor. In the next step doctor investigates about more symptoms and condition from the patient. This investigation process is a step by step questioning process. It also includes measuring blood pressure, temperature of body. The patient’s status can also be considered while diagnosing the disease. In the mobile app input taken through speech, converted to text, text is processed, keywords are identified and used for deciding most probable disease. The speech input is converted to text on mobile device and it is sent to the backend program residing in the cloud. This text is processed and the next question is sent to mobile device. There are certain limitations such as we cannot measure temperature or blood pressure through a mobile device. The mobile app did not handle usage of patient’s history, diverse cases, rare diseases etc. The mobile application contains database for three diseases malaria, typhoid, and dengue, which can be extended.

III. PREVIOUS WORK

Knowledge base in the project is built with the help of experts and medical textbooks. There will be symptom matrix for each disease. This matrix represents presence or absence of a particular symptom with respect to a particular disease.

Input is given as a transcript of patient's condition and symptoms which is very similar to explaining to a real doctor. Then preprocessing of this obtained text is done. This is done using NLTK python library. Several techniques are used to process the input text such as sentenizing, tokenization, spell corrector, keyword extraction and stemming represented in order of their execution. Each of the above has different tasks. Sentenizing breaks the text into different sentences. Tokenization breaks the sentence into tokens. The spell corrector corrects them .then the keywords are extracted using removal of stop words and parsing the text using AIML (Artificial Intelligence Markup Language). These key words are stemmed using porter stemming algorithm.

Symptoms are obtained from the keyword with the use of database of symptoms. The disease matrix will be constructed with the help of above symptoms and their values in the knowledge base .After getting disease matrix with respect to a disease in knowledge base, for further analysis we need to get the mutual information between these two. With the help of mutual information theory, weights are provided to different situations

$$I(X;Y) = \sum_{y \in Y} p(\bar{y}) \sum_{x \in X} p(x|\bar{y}) \log \frac{p(x|\bar{y})}{p(x)} =$$

$$\sum_{x,y} p(x,\bar{y}) \log \frac{p(x,\bar{y})}{p(x)p(\bar{y})}$$

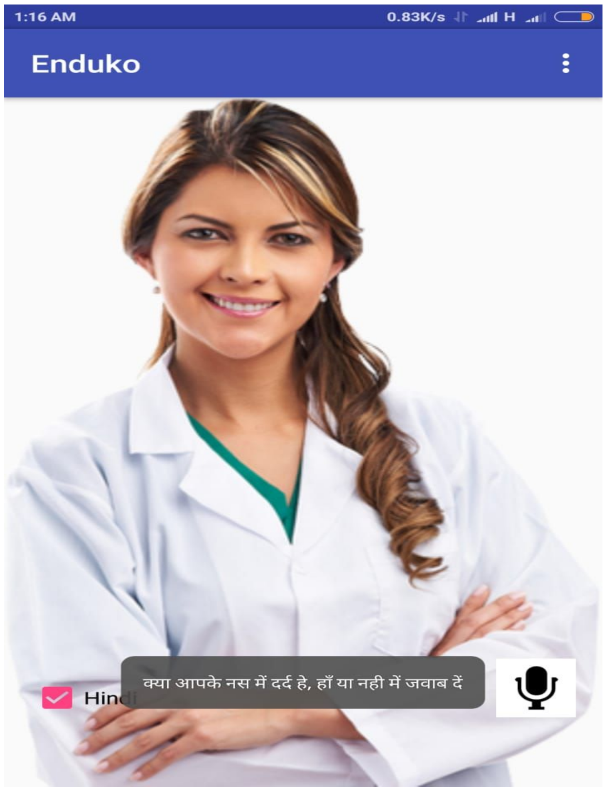
Where, X is represented as the disease and Y as the symptom

From this, we get the main symptom of the most probable disease. Then questions were asked to the patient depending up on the main symptoms obtained above and after each query, the disease matrix is updated. When there are no more further symptoms and all the analysis is done, the procedure is stopped and the most probable disease is obtained as output.

IV. ANDROID INTERFACE

Our Android App is developed using Android studio. It takes input through both speech and text. The speech recognition is done using Google speech recognition API. Google speech API convert audio to text. It is powered by neural networks and high end machine learning algorithms. When the symptoms are first uttered by the device user it is converted into text and stored. This piece of text is sent over the server. We can provide input in Hindi as well as in English.

The server side code processes the text and extracts keywords relating to medical conditions and symptoms. The server then creates a symptom array. This symptoms array is attached to the header file and sent to the user device. The server remains stateless about user throughout the process. The symptom array acts a cookie throughout the entire process of diagnosis. This is done for accommodating many users which are accessing the server. The device then asks a question choose based on the symptom array (where symptom flagged -1).The answer from user could be taken using both speech and text which is either yes or no. This answer sent to server along with the symptom matrix to calculate disease probability. This process is repeated until probability of any disease crosses the threshold.



क्या आपके नस में दर्द हे, हाँ या नही में जवाब दें

Fig2: Mobile application interrogating the user for further symptoms and conditions

Finally the mobile application prints the disease on device screen. The device also shows the route to the nearest consultant for the disease using google maps API. Using google maps API the current location of user is extracted and is used to suggest, show routes of the nearest medical clinics and hospital who are experts in the specific disease. The Google map API requires credit card info entering which we get access to this facility.

V. CLOUD PROCESSING

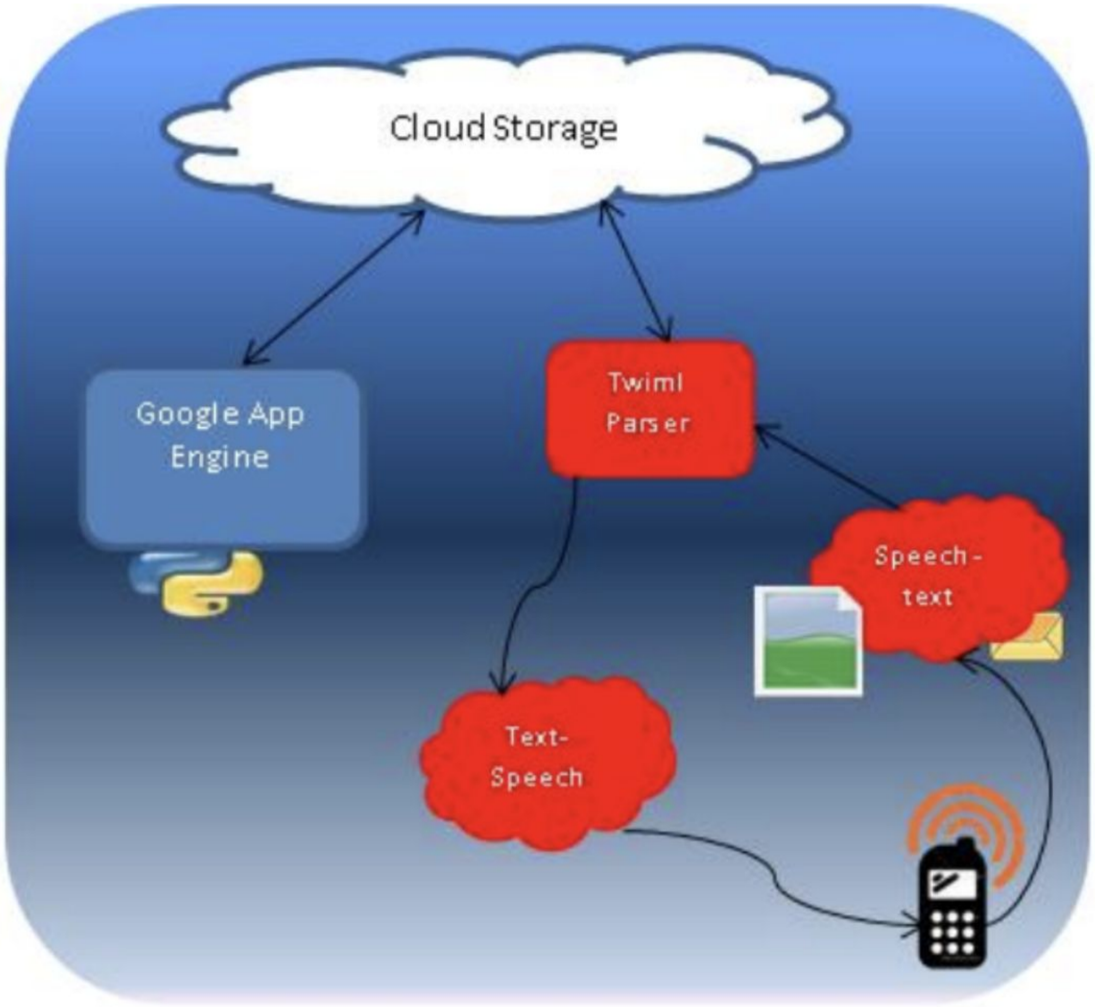
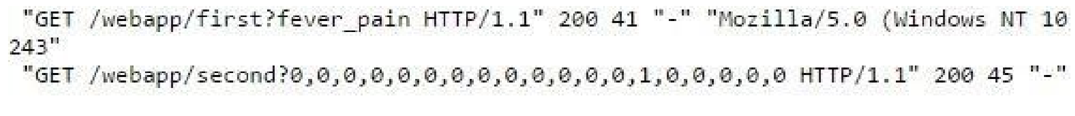


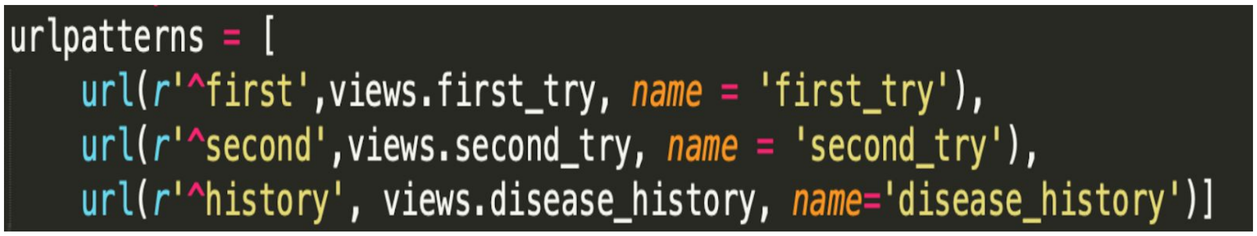
Fig. 6 Figure showing the overall picture about the interactions between users the api and python code run on Google app engine.

The application is deployed in cloud which could accessed by a user from any place. We have used PythonAnywhere to host the Django code. Deploying the application in cloud makes it more scalable and reduces overhead on the device. Helps in good management of medical databases and history. Our application is developed in Django. Django is a high level web framework using python language. This django application could deployed on a personal server or some cloud storage. We deployed this application on pythonanywhere cloud system. The pythonanywhere platform has a simple configuration settings for deploying and constructing our application in desirable way. The cloud platform receives messages from user i.e. mobile application. It processes those messages and return a message to mobile application. This returned message may be the diagnosed disease or interrogative question to gain more information of patient's condition. The mobile device just acts as an interactive interface between the server program and user. The server deployed in the cloud does not store state of user conditions at any point of the interaction. The user conditions are stored in a cookie. The cookie helps in storing user symptoms and helps for further processing. The cookie is implemented as symptom array.



```
 "GET /webapp/first?fever_pain HTTP/1.1" 200 41 "-" "Mozilla/5.0 (Windows NT 10
243"
 "GET /webapp/second?0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0 HTTP/1.1" 200 45 "-"
```

Fig4: Messaging between user device and cloud server using a cookie.



```
urlpatterns = [
    url(r'^first',views.first_try, name = 'first_try'),
    url(r'^second',views.second_try, name = 'second_try'),
    url(r'^history', views.disease_history, name='disease_history')]
```

This way many users could be handled by the application with less overhead. The cloud also helps in storing of user medical history for deciding the probabilities of a disease. The application could be deployed in any django cloud system when scaled in real world. Each user could be given a login id for identifying the patient. We basically call 3 functions , first, second and disease history . First is used to get the inital symptom array, second gives the subsequent updated symptom array and disease history is obtained using the call on disease history in views.

VI. CONCLUSION

We have successfully implemented an android application based on cloud server which can provide medical diagnosis from even remote places. We diagnose the disease and then locate the nearest hospitals present. The diagnosis is done by asking questions for more details based on the highest probable disease. We can diagnose diseases from fever, heart attack and some common digestive problems.

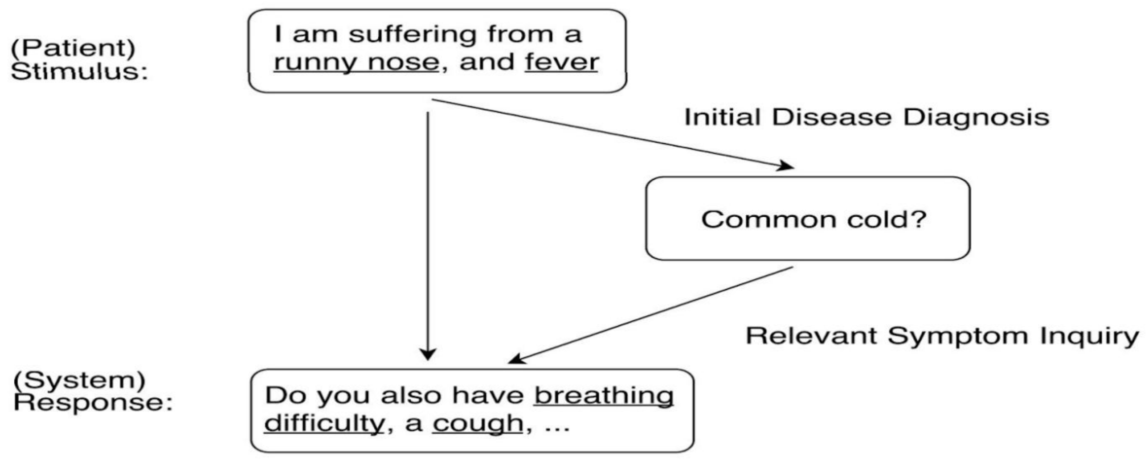
The system is tested over many test cases considering the symptoms based in knowledge base for diseases malaria, dengue, typhoid and other diseases. The project is more focused on providing diagnosis for common diseases. To accommodate more diseases we should extract conditions and information about these diseases and add it to the knowledge base.

VII. SCOPE FOR IMPROVEMENT

In future, the application could also help patients to connect with doctors with the help of hospitals and medical organizations. Advancements in field of human interaction, cognitive systems, sensors, computer vision and natural language processing would improve the research capabilities in medical diagnosis. This could also ensure more marketability for the medical assistance applications.

We can also try to integrate more diagnostic applications like blood pressure and heart beat checker, images to identify the symptoms and many other modules. We have implemented the entire project in English but, for it to be useful for an ordinary person in the rural areas English may not be the suitable language. We can try to implement modules that can recognize their native languages like Hindi, Telugu, Tamil etc. and all the symptoms in that native language also.

The mobile application that we implemented has the map facility. But this facility requires google maps to tag hospitals present in the region. If there are no tags to hospitals in the nearby area then we do not get any hospitals like in the case of Pilani. We can improve this by partnering with an external company that has all the list of hospitals.


(Patient) Stimulus:
I am suffering from a runny nose, and fever
Initial Disease Diagnosis
Common cold?
Relevant Symptom Inquiry
(System) Response:
Do you also have breathing difficulty, a cough, ...

The main other part that is required is improving the intelligence. This can be done as follows :

(1) How to generate most relevant symptoms given patient described symptoms and an initial disease hypothesis? Such to-be-generated symptoms shall be utilized to ask patients and decide the disease ultimately. The problem is challenging in the sense that: (i) Both correlations among symptoms and those between diseases and symptoms should be captured in order to generate a high-quality response; (ii) The symptoms shall be generated in a decreasing order in terms of priority to ask patients, which is reflected in a real situation.

(2) We propose an augmented symptom generative model based on LSTM, whose network architecture can maximize information utilization for the relevance inference. It naturally incorporates the inputs from embedding vectors for patient described symptoms and an initial disease hypothesis given by a predictive model,to generate relevant symptoms for further disease prediction. The generation process essentially models the conditional probability of observing a

symptom given a set of patient described symptoms, an initial disease hypothesis, as well as newly generated symptoms. Therefore, it meets the above two challenges in dealing with the problem.

(3) Our work pave the path towards building a medical self-diagnosis Android framework, which mutually incorporates relevant symptom generation and disease prediction. The framework could also potentially answer health-related questions and provide people valuable information regarding medical suggestions, diagnosis, treatment, drug, etc., by extending the relevance inference to these topics.

VIII. REFERENCES

1. 2374-8486/16 $31.00 © 2016 IEEE , Augmented LSTM Network to construct Medical Self Diagnosis Android.
2. Mutual Information homepage on Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Mutual_information
3. PythonAnywhere website [Online]. Available: https://www.pythonanywhere.com/
4. Android Studio home page. [Online] Available : http://developer.android.com/sdk/index.html