Medic - Artificially Intelligent System for Healthcare Services

Arockia Panimalar.S[1]

[1]*Assistant Professor, Department of BCA & M.Sc SS, Sri Krishna Arts and Science College, Tamil Nadu*

Abstract - *The applications of Artificial Intelligence has been increased day-to-day. This is because of the inclusion of high-tech gadgets in our lives. These gadgets provide high computational capabilities and geographical reach. These two features could be exerted to provide medical services to the society. This paper emphasizes on creating a software infrastructure which would provide healthcare services like diagnosis of diseases, advising medical tests to patients, providing medical prescription to patients by making use of personalized medicine problem solving algorithms and providing medical assistance to doctors. This project, Medic, makes use of natural language processing, fuzzy logic, deep learning and a constantly evolving knowledge base to correctly diagnose diseases. It also provides various services to doctors which would help them while making decisions regarding any patient's medical treatment.*

Keywords: Artificial Intelligence, Natural language processing, Image recognition, Convolution neural networks, Deep learning, artificial neural networks

1. INTRODUCTION

Having a healthy body is the biggest blessing we can possess. It is because of our good health, that we can overcome all the negativities around us and help the society in various possible ways. However, maintaining a healthy body is not a simple task since it requires large number of precautions to be taken. In addition to this, because of the fast moving lifestyle, people have started to overlook their health concerns. According to an analysis based on the 2013 Global Burden of Disease Study, it was found that more than 95% of the world's population is ill [4]. It clearly indicates the negligence shown by our society towards healthcare. We can however take advantage of healthcare services offered by doctors and hospitals to retain our mental and physical health from illness. But, according to a research study published by One Medical Group that was gathered by Kelton Research in 2012, it was seen that majority of the respondents avoided doctors or hospitals despite being sick just because they believed it was a time consuming process. In the same research study it was seen that 45% of the respondents felt that visiting a doctor or hospital was a costly affair [4].

This project focuses on providing healthcare services to the people at their fingertips and that too at a low cost. Therefore Medic will not only save people's time, but also money. Hence, eventually more and more people will be attracted towards making use of this service and we will have a healthier society as a result of it.

The motto of Medic was to create cross-platform (major platforms such as smart-phones, tablets and personal computers) compatible software which would offer healthcare services to public. By making use of Medic, users can enter their health related problems along with some basic information and then the software will diagnose the disease and will provide a prescription for the same. In some of the cases Medic may ask its users to visit a hospital or a medical laboratory, if it thinks that certain medical tests are required to be taken by the users for proper diagnosis. If the software identifies any highly severe disease, then it will ask its users to visit a doctor for confirmation of that disease and proper physical treatment. Medic will provide basic information, medical history and previous prescriptions of the user to the doctor once the meeting is set up by it. This information would assist the doctor to cure the patient much faster than the normal approach. Medic will also offer patient records from all around the world to doctors, which would help them while making decisions.

2. RELATED WORK

In past, several efforts have already been taken in order to implement the knowledge of artificial intelligence in the field of medicine and healthcare (projects synonymous with [6] & [7]). There are computational devices and software which make use of AI algorithms to solve or assist in solving health problems. Out of these projects, some work at very detailed level by accepting features highly specific about certain disease or health problem.

Gavin Robertson et al. [8] have written an article which describes a project falling into this category. They have accepted various features required for predicting blood glucose level. This project and other such similar projects are generally domain specific i.e., they can only be used to carry out tasks related to a limited number of health problems, but their results are found to be highly accurate. There are some other projects as well which accept primary symptoms from their users and use them to solve personalized medicine problem.

Jamilu et al. [1] have described a project falling into this category in their research article. Such projects are much broader in scope. But lately, many researchers have doubted the accuracy of projects falling into this category, since some diseases need various parameters which can only be obtained from medical tests to be exactly diagnosed. Medic however, takes the positives from both the types of projects. It can diagnose nearly 1000 different diseases by making use of its default database. This count increases when Medic comes across unknown scenarios, when it makes use of its self-learning capabilities and a web crawler module to diagnose the unknown disease. This project can also get at detailed level by asking its users to undergo certain medical tests to obtain the data required to diagnose a disease which cannot be accurately diagnosed just by taking the primary symptoms into consideration.

3. ARCHITECTURE

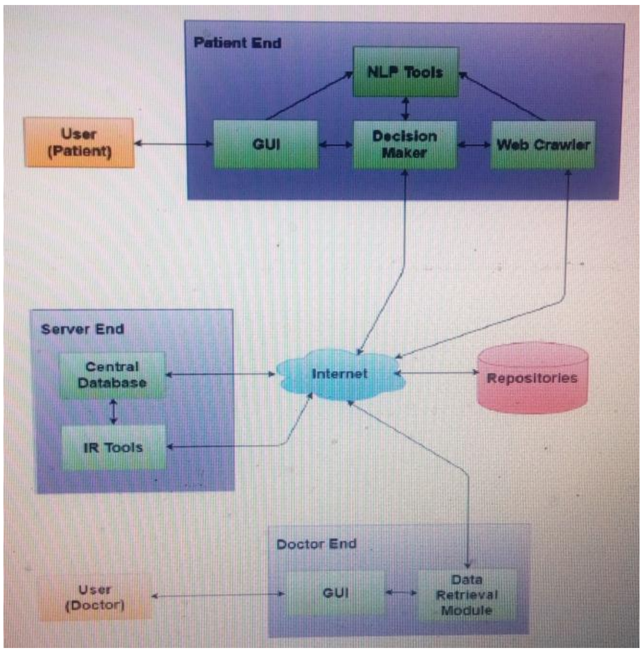

Fig 1: Architecture Diagram of Medic

Fig.1 shows the diagrammatic representation of the architecture of Medic. The project is architected such that it can implement self-learning procedures easily and that too without affecting any other module. The architecture can be divided into 3 parts based on the working of this project and its interaction with the users. These 3 parts are sequentially discussed in the subsequent sections.

A. Patient End

This is the end at which patients provide their health concerns and get health assistance in return. It has 5 modules.

1) Graphical User Interface (GUI): It is an interface which interacts with users to obtain their basic information and health related problems. Final results are shown over this module. It provides basic inputs to NLP module and decision maker module.

2) Natural Language Processing Tools: These tools are required for transforming GUI or web crawler provided terms into the system known terms. It implements a sequence of procedures for carrying out this transformation. These procedures are, part-of-speech tagging, finding out unnecessary terms, neglecting unnecessary terms, building sets of terms by finding out synonyms of the terms and matching these sets with system known terms. Once this transformation is complete, the system can make sense out of the data provided by the user or the web crawler. NLP module of this project makes use Natural Language Toolkit [5] available in python to carry out intended tasks.

3) Web Crawler: This module plays a vital role in providing self-learning abilities to this project. Web crawler is invoked only if the system comes across any unknown set of health problems. It is a simple python script which carries out search operations over a number of well-known healthcare web repositories. The results of these operations are generally data tuples, web pages and images related to user entered health problems. The data tuples are directly stored in the central database. The images are fed to the decision maker module along with some data for carrying out further operations. The web pages are fed to the NLP module to get sense out of the raw data. The decision maker module is provided with the results from the NLP module for updating the central database.

4) Decision Maker Module: This module can be imagined to be the brain of the patient end. It makes use of Fuzzy logic to make decisions. Once it receives inputs, first it decides whether to invoke the web crawler or not. This decision is made by fuzzy matching of the input with available data sets in the database. If the result is below a preset threshold, then web crawler is assigned a task to fetch information regarding the current input over the internet. Otherwise, the results are obtained from the database & existing convolution neural networks from IR module, aggregated and presented to the user.

B. Server End

The main objective behind having 'Server End' is centralization of data. Since all the high volume data and convolution neural networks are stored at this end, other 2 ends of Medic have very little memory requirement. This end consists of 2 modules.

1) Central Database*:* This database contains records of number of different diseases, their symptoms (stored as text fields as well as images) and drugs required to treat them. Medical history and previous prescriptions of every Medic user are also stored in the central database.

2) Image Recognition Tools: These tools are required for identifying the disease from visible symptoms provided by users. This module makes use of trained convolution neural networks for identifying the disease accurately. All the artificial neural networks of Medic are feed forward neural networks having multilayered architecture. GUI backend (at patient end) scales down the user entered image to some extent and then gives it to its decision maker module.

The decision maker module makes a remote call to the IR module present at the server end. Then it feeds the input image to the CNNs present in the IR module. It should be noted that all the convolution networks are stored on GPUs (Nvidia GTX-970) present at the server end. These CNNs then calculate results by simple matching. These results are provided to the decision maker module. The IR module can be called by decision maker module for implementing the self learning procedure also.

During this procedure, IR module is provided with an image or set of images along with the output they should yield. The IR module then first checks whether the input data is already present in the temporary memory (at server end). If a match is not found then it checks whether the temporary memory has enough space to accommodate input data. If enough space is available, then the data is stored in the temporary memory and the self-learning procedure halts. Otherwise, a new CNN (having architecture similar to the pre-existing CNN) is created in one of the GPUs. Then all the data stored in the temporary memory is extracted. This data is nothing but set of images and their corresponding outputs. Then this data is fed to the newly created CNN to carry out learning. This learning may take up to 2 to 4 hours.

C. Doctor End

This is the end at which doctors interact with Medic. Doctor end offers a GUI for presenting the requested data to the doctors. This end also has a data retrieval module, whose job is to provide user requested data, find out datasets stored in the database similar to the input dataset & publish the results.

4. IMPLEMENTATION

A. Convolution Neural Network

Convolution neural networks offer large number of features at the cost of memory. These networks can carry out highly intricate tasks with ease, since they have large number of neurons. As a result of this, they can outperform shallow neural networks (networks having 2-3 layers) while finding relationships between convoluted input patterns (Comparison between shallow neural networks and convolution neural networks is well explained by [2]. Convolution neural networks are essential in Medic for identifying diseases from their visible symptoms. Medic comes with an inbuilt CNN having a capability of recognizing visible symptoms of 965 different diseases.

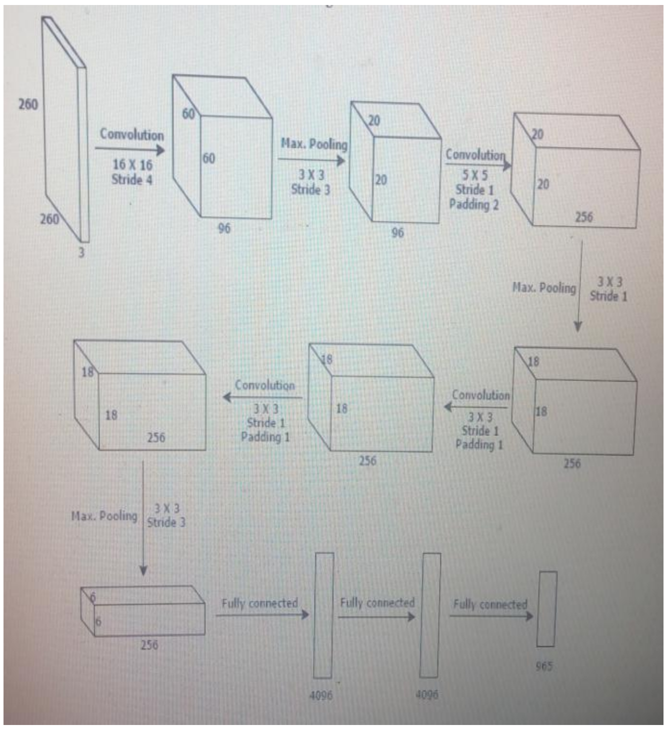

Fig 2: Convolution Network Used in Medic

Fig. 2 shows the diagrammatic representation of the convolution network used in Medic. It has 10 layers. It can be seen from the diagram that, the network makes use of convolution and maximum pooling (Similar to [3]). During convolution, the input data is split into number of parts, where size of each part is determined by size of the filter, and a set of outputs is generated. During maximum pooling, the input data is divided into number of parts, where size of each part is determined by size of the filter, and the output is determined by selecting the maximum value from each of the parts. The input to the convolution network is a colour image having a resolution of 260x260. The input transformation over different layers of the network is shown in the diagram over the arrows. The output of this network is classified into one of the 965 different categories.

For implementing this convolution network, firstly the code was written in python environment by making use of TFLearn. TFLearn is a wrapper around Google's Tensor Flow library [9]. By making use of this library, we can make use of various deep learning features without getting into their complex insight. The network was then loaded onto a GPU for carrying out training and testing of it. It should be noted that, GPUs are always preferable over traditional CPUs while performing deep learning, since they can carry out matrix related operations much faster than traditional CPUs. The GPUs however have much smaller memory capacity as compared to CPUs. Therefore, instead of making use of single GPU, numbers of GPUs were used for Medic. It confirmed that, even if the memory requirement of any convolution network exceeded the available memory size of any GPU, it could still be accommodated by making use of another GPU. However, because of the usage of multiple GPUs a mechanism for inter-process communication between 2 GPUs was required. For implementing this mechanism firstly it was made sure that every GPU at the server end was Nvidia GTX970 in order to maintain uniformity. Since all the Nvidia GPUs have CUDA cores, PyCUDA developed by Andreas et al. [10] was used for establishing the aforementioned mechanism.

B. Natural Language Processing

NLP has 2 key areas of implementation in Medic.

1) NLP in transforming user input to known format

Once the user input is obtained, all the symptoms are temporarily stored in a list. Before the input transformation takes place it is assumed that every symptom entered by the user is separated from the other with a ','. The transformation takes place for every input in the list. First step in the transformation is part of speech tagging. During this step, NLTK (provided by python, explained in detail in [5]) gives a tag to every word in the input string depending upon their parts of speech. The result of this step is stored in a hashmap (by making use Python Interpreter class). Words are stored as keys and their tags are stored as values in this hash map. Then, all the keys in the hash map having values DET (article), PRT (particle), (punctuation marks), PRON (pronoun), NUM (numeral) and CONJ (conjunction) are straight away removed. Then set construction phase starts. During this phase, synonyms of every key of the hash map are determined. Then every synonym of every key is combined with every synonym of the other keys in the hash map to produce phrases.

The phrases are formed in such a way that every phrase contains only a single word having VERB (verb) tag. These phrases are then stored in another list. Every element of this list is compared to all the symptoms in main database (by making use of Lucene) and the process is halted the moment a match is found. It should be noted that, the order of words is not considered during matching and it is a fuzzy matching process. If match is successfully found, then the system stored version of the matching phrase is returned otherwise 0 is returned. The presence of vague and unnecessary keywords the system comes up with a correct answer. However, as the percentage of vague data in input increases, the time taken by the system to come up with an answer also increases.

2) NLP in transforming web crawler results to known format

In this case the input to the NLP module given by the web crawler module after carrying out its intended task. The input is a webpage or set of web pages consisting of information regarding some health problem. Once the input is obtained, the web pages are stored in temporary memory and operations are performed over one webpage at a time.

Fuzzy matching is carried out over the contents of webpage. Then all the sentences having matches whose magnitude of matching is greater than certain threshold value are selected for further transformation. Each of these sentences is processed one by one until a disease or a set of diseases is found for given symptom. For carrying out this searching, firstly every sentence is POS tagged and stored in hash map with words as their keys and tags as values. Then all the key-value pairs having values x (other) and NOUN (noun) are considered for further searching and rest of the key-value pairs are removed. The remaining keywords in the hash map are searched again on the web repositories to make sure that these keywords are diseases which are caused by the input symptom. For this process of confirmation, keywords from hash map are considered as inputs and input symptom is the phrase to be found in the search results performed on keywords from hash map. This is generally a time consuming process and can take up to 5 minutes.

C. Similar Scenario Finder

Similar scenario finding algorithm is required to be implemented at the doctor end since the doctors might request patients' data similar to that of the data of the patient currently undergoing treatment at their clinic or hospital. For carrying out this matching, parameters like age group, gender and symptoms are considered. Age group (age group patient) and gender (gender patient) of current patient are used for direct matching whereas; symptoms (symptoms patient) are used for fuzzy matching.

This project makes use of python's Fuzzy Wuzzy library which in turn makes use of Levenshtein Distance to calculate the differences between sequences, essential for fuzzy matching of 2 strings. Once the initialization is complete, tuples are appended to the result in a descending order of their priority. Threshold values are taken in order to filter out non-required information. For this project, the value of *BIG_THRESHOLD* (assuming user enters more than one symptom otherwise the two thresholds will have value equal to 1) was set to be, [*total_user_entered_symptoms* – 1] and the value of *SMALL_THRESHOLD* was taken as 1.

5. LIMITATIONS

The following are the limitations of Medic:

i. The natural language processing module of Medic supports English language only.

ii. If any patient provides large number of inputs (beyond system's capacity), then instead of coming up with a highly precise answer, the system crashes.

iii. If the results from image recognition module and decision maker module differ drastically, then system yields less precise or sometimes no output.

iv. Medic cannot work without internet connectivity.

v. If every web-repository falls short of providing web crawler requested data, then Medic will fail to diagnose the diseases of its patient.

6. CONCLUSION

Vast developments in the field of technology can be implemented in medical science, in order to help the society. As far as Medic, this implementation heavily relies on artificial intelligence algorithms and its techniques. Although, Medic is in its early phases of development, it can diagnose up to 1000 different diseases. This highlights the massive strengths possessed by AI techniques, points up extensive scope in the fields of AI to researchers and shows a glimpse of futuristic technology. Many forecasters think application of AI techniques could replace most of the jobs traditionally done by humans. However, from Medic we can proclaim that these technologies can assist us in our jobs and thereby make our daily lives easier and safer. Since Medic not only helps patients but also doctors, we can assert that Medic does not try to replace human doctors but rather tries to assist them. Finally, a successful implementation of AI techniques in the medical field would save many lives and help society being healthier.

7. SCOPE FOR FUTURE ENHANCEMENT

Since the logical area of implementation of Medic is widespread, there is a lot of scope for future work in it. The software can be expanded to work on various medical devices such as, CT scan machine, X-Ray machine, MRI machine and blood pressure monitor, in order to get accurate patient data and get rid of the hassle caused by manual feeding of information. Various caching techniques can be used for reducing the memory requirements of the software. NLP module can be modified to generate accurate results in lesser time. Adaptive strategies can be used at the IR module for improving the self-learning procedure. Since Medic is highly dependent on AI techniques, its tools can be upgraded whenever better AI techniques come into existence.

8. REFERENCES

[1] Jamilu Awwalu, Ali Garba Garba, Anahita Ghazvini, Rose Atuah "Artificial intelligence in personalized medicine application of AI algorithms in solving personalized medicine Problems".

[2] Michael Nielsen. (2016, January). Neural networks and deep learning.

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton. "Imagenet Classification With Deep Convolutional Neural Networks". (2012).

[4] The 3 Most Common Reasons We Avoid Going to the Doctor.http://www.fool.com/investing/general/the-3-most-commonreasons-we-avoid-going-to-the-do. Aspx

[5] Bird, Steven, Edward Loper and Ewan Klein (2009), Natural Language Processing with Python. O‟Reilly Media.

[6] Anthony Farrugia, Dhiya Al-Jumeily, Mohammed Al-Jumaily, Abir Hussain, David Lamb. (December 2013). "Medical Diagnosis: are Artificial Intelligence systems able to diagnose the underlying causes of specific headaches?".

[7] Shih-Chung B. Lo, Shyh-Liang A. Lou, Jyh-Shyan Lin, Matthew T. Freedman, Minze V. Chien, Seong K. Mun. (December 1995). "Artificial Convolution Neural Network Techniques and Applications for Lung Nodule Detection".

[8] Gavin Robertson, Eldon D. Lehmann, William Sandham, and David Hamilton, "Blood Glucose Prediction Using Artificial Neural Networks Trained with the AIDA Diabetes Simulator: A Proof-of-Concept Pilot Study".

[10] Andreas Klockner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, Ahmed Fasih, PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation, Parallel Computing.