Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 04, April-2020

Symptoms Extraction from a Voice Input using Natural Language Processing

Chethana Saligram[1], Deepali B K[2], Gowri K S[3], J Ajay Kumar[4], Dr. Jyoti R Munavalli[5]
[1,2,3,4] Student, Dept. of ECE, BNM Institute of Technology, Bangalore
[5]Associate Professor, Dept. of ECE, BNM Institute of Technology, Bangalore

***Abstract*— Natural Language Processing is a field that is concerned with interactions between humans and computers using the natural language or the language that humans use. It focuses on making the computers understand the language of humans and to process it to collect information. This field has proved to be very useful in several areas. In this paper we intend to explore the application of natural language processing in the field of primary healthcare. This system is provided with an input regarding the health issues of a person. The system understands the given input and pulls out only the symptoms that are provided by the user which is further utilized for classification purposes. This system was implemented using NLTK Python package and results were displayed using the UI window created using tkinter. The accuracy of this system was found to be 0.8752.**

Keywords*— Preventive healthcare; Virtual Assistants; Medical information extraction, Natural Language Processing

I. INTRODUCTION

Sickness is not a choice but an uncertain event that happens without knowledge. At times, this illness might be trivial or serious. People, often cannot afford to spend time travelling to a doctor in their busy metropolitan life. To help such people, symptom checkers that provide fast and good aid in improving the health are used [1]. Though there are several symptom checkers yet research in this field is still active as more and more accurate systems are in demand. We set out to develop one such system which is interactive, learn the problems of the user and provides with close to accurate diagnosis along with some remedies.

The vital part of any conversational device is the ability to understand and comprehend what the user is speaking. In other words, defining a set of rules or to be able to teach the device to take in the input, parse it and deduce its meaning. Our project is to build a simple voice-based disease prognosis device (symptom checker). Hence, applying Natural Language Processing (NLP), we create an interface between the user and the core part of the symptom checker (which is the classification of diseases). However, in this paper we focus only on the NLP part of our device.

II. RELATED WORKS

In this section we present literature survey carried out for this work. Literature work from Google scholar and Google search engine were searched with the keywords: preventive healthcare, Virtual Assistants and medical information extraction using NLP. Relevant papers are summarized here.

Emad S. Othman [2], implements a Personal Assistant using Raspberry pi. This Personal Assistant performed numerous tasks like reading out the news, giving information regarding the weather and much more. Primary hardware used is Raspberry Pi to which voice commands are the primary inputs. This device has to be connected to internet continuously, machine learning is also implemented. In the proposed methodology, online modules of speech-to-text and text-to-speech by the AVS are used. The text is query processed online on the wolfram alpha site based on the keywords uttered by the user. Few main keyword modules are wiki, news, joke, weather, etc. This system claims to be customizable as it is using open source modules. Internet is continuously required for performing the tasks. In another work a symptom checker is built using a medical knowledge base and natural language processing. Its medical knowledge base consists of data from open source medical resources, texts, and claims data. This method provides diagnosis using the historical data. Knowledge base is created automatically using structured clinical resources, medical texts, and information extraction engine. Unified Medical Language Systems (UMLS) is used to integrate all the data collected into a knowledge base. The symptom checker includes a front end that collects the user information and a Web server that links all the components present in the checker, it sends the data collected by the front end to the NLP component. NLP component does three tasks, paraphrase generation, negative detection and phrase extraction. A personalize component which provides personalized output based on clinical claims. This symptom checker found discrepancies in classification as KB had more data for one particular class while less for the other. The symptom checker demonstrated an accuracy of 65.9% on an average [2].

Arlene E. Chunget.al.[3], examines the characteristics of health and fitness apps. A number of health-related Voice Activated Assistants (VAA) that are used in a variety of settings for many different functions, such as home monitoring of symptoms or health education. Some studies have suggested using Amazon Alexa to asses deaf speech, provide task support for individuals with cognitive disabilities, and receive voice input from patients to determine “unexpected changes in mood”. Such VAA vendor apps are different from the conventional mobile apps, in that the types of interactions are not fully delineated by voice interface because it does not have the same transparency as interacting with a physical user interface. There is also no menu of features or functionalities such that the user could understand the full spectrum of the types of commands one could ask or what kind of information could be provided by the app.

A study was carried out to conduct the descriptive content analysis based on the information provided by the vendors to determine the types of apps in health and fitness categories for commercially available VAAs. As per the review, Amazon and Google were the only companies with commercially available hands-free VAAs. The selected apps of hands-free voice activated assistants are screened for health and fitness apps and the filtered ones are taken for review. After evaluation it was found that 309 apps were categorized as "health and fitness apps" by vendors. Their analysis revealed that there were only a limited number of health monitoring apps. This may be, in part, due to the restrictions within Amazon and Google policies for publishing health-related apps. Amazon and Google Home devices have a button that can be pressed to "mute" the device from listening, but instruction manuals do not have explicit language describing that the devices are always listening and that they are recording and storing audio. Also, device and usability should be improved, and focus should be given on interactive user-based interfaces. Apps are released, modified, updated, and discontinued on a regular basis. As a result, there may be apps that were reviewed in this study that are no longer available, and there may have been modifications in app descriptions since data extraction. Despite these limitations, the paper contributes towards an understanding of the characteristics of health and fitness apps available for commercially available, hands-free VAA [3].

Gunjan Dhole et.al.[4], published a paper "Medical information extraction using Natural Language Processing interpretation" which is based on retrieval of medical data from narrative clinical documents using Natural Language Processing (NLP). It explains about how NLP is used to extract medical information from a narrative text by using tools such as tokenizing, noun entity recognizers, parts of speech taggers and relationship extractors. However medical text is different from normal text as it contains complex medical terminologies due to the synonym and disease names. The system that is proposed in this paper tries to understand the text relating to a list of symptoms and it will try to give out the relevant answer.

The authors have clearly stated the need for using NLP in this case. According to them most information retrieval systems are rule based systems as against NLP which can extract data from free text form. Several medical extraction systems have been researched and mentioned about in the paper.

Information Retrieval:

NLP stores and interprets meaning at both query and the document. A general text retrieval system would have the following flow:

1. Document processing
2. Query processing
   a) Query matching
   b) Ranking and sorting

The knowledge base stores all the medical documents. The system consists of two modules, document and query processing. The answer processing module will check which type of query is coming in. If its "Definitive" it will follow the flow for definitive queries. If it's "Descriptive" query, it will follow the flow for descriptive queries. Document Processing and retrieval is done from the Query processing module which gives noun entities as output.

Lots of research is ongoing in the field of extraction of medical data using NLP. It needs different tools as compared to the normal ones. The system described in this paper does not store the information or the medical history anywhere and the extraction techniques can be improved further [4].

Gustav Cederblad [5], in "Finding synonyms in medical texts – Creating a system for automatic synonym extraction from medical texts" focuses on bridging the gap between laypeople and medical professionals by extracting the synonyms from a corpus consisting of medical texts and to evaluate whether these words are actually related. Because, sometimes the more specific medical words are described by a more commonly used familiar word. The aim is to find synonyms and related words for different diseases. In the unigrams that are considered after tokenizing only the noun entities are retained since synonyms belong to the same part of speech (noun).

They give a stark description about the semantic relationships and weighting, which deals with giving importance to words occurring in the same context, close to the target word, giving a hint about the word's true meaning. Work that has been already done in synonym extraction has been reported.

The project is implemented in stages. At the first stage unwanted data is got rid of, in the medical corpus, leaving only the necessary information. Misspelled or unknown characters are taken as noise which should be reduced, and hence regular expressions are used which filters out irrelevant data. The second stage is a part of speech filtering. They make use of a SNOMED dictionary describing diseases, words are chosen from that, which are all nouns. They only focus on unigrams. Hence, filtering out all the non-noun (NN) words would lead us to the synonyms that are actually required. The pos tagger – NN is made use of. However, it is not 100% accurate.

Since, in large pieces of text, it is very likely that the same word will appear in different forms, NLP techniques like stemming and lemmatization are utilized. Distinguishing between words having the same root word is unwanted since it is only the meaning that is of interest to the analysis. Since Stagger has already been used for the part-of-speech tagging, lemmatizing the corpus is not more computationally costly than stemming. That is because all the lemmas have already been extracted and it is just to replace its original word in the corpus.

Later, they go about extracting the semantically related words getting a csv file as an output consisting of the most similar words for each SNOMED term. The analysis is further done. The use of a questionnaire is stated to interact with the user to narrow down to disease they are conveying. A human evaluation on a few selected members is performed and the results and the statistics are explained in the paper.

The limitation of this system was that it only handles unigrams which could decrease the overall performance of the system. Bigrams or further ngrams would also describe the adjectives for a particular symptom, like "runny nose" or "severe pain". This emphasizes the need for a model that can manage n-grams larger than unigrams. The noise removal is also not very efficient in retaining only the noun parts of the user input. When replacing synonyms in texts, it is imperative that no information gets lost, especially in a medical context where correct information could be the difference between life and death. Dealing with this sensitive and personal information is difficult and entails many ethical questions that have to be considered. Hence there is much room for improvement on the aforementioned system [5].

III. PROPOSED METHOD

For a device to accept voice input and understand it, certain steps have to be followed to parse the input. The proposed method includes the following steps:

1. Speech synthesis
2. Modifying the dataset
3. Interpreting the input
4. Review

We have made use of NLTK package to perform the tasks, due to its easy to use interfaces such as WordNet, and a suite of text processing libraries for tokenization, stemming, tagging, parsing and so on, all of which shall be explained below. It is a free open-source package. Pandas library is a fast, flexible and a powerful tool that can be used for reading datasets in our project.

The block diagram of the proposed method is as shown in Fig.1.

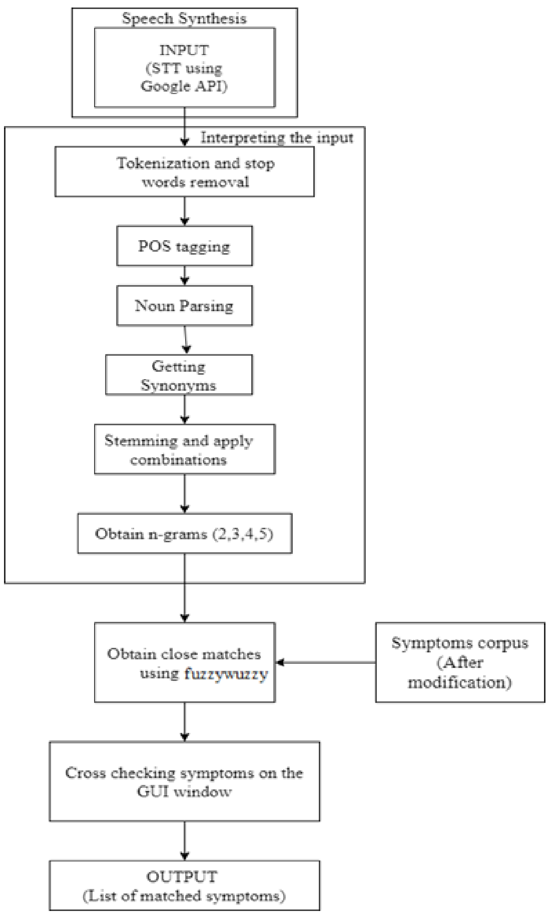

Speech Synthesis
INPUT (STT using Google API)
Interpreting the input
Tokenization and stop words removal
POS tagging
Noun Parsing
Getting Synonyms
Stemming and apply combinations
Obtain n-grams (2,3,4,5)
Obtain close matches using fuzzywuzzy
Symptoms corpus (After modification)
Cross checking symptoms on the GUI window
OUTPUT (List of matched symptoms)

Fig.1: Block diagram of the proposed system

A. SPEECH SYNTHESIS:

Starting off with the process, the prime component is the speech. It is random and cannot be directly used. Hence, an analog to digital conversion is done to digitize it to convert it from the raw physical sound to an electrical form with the help of a microphone. Once this is done, there are several models that transcribe the audio into the textual form. There are several packages/ APIs to perform this.

We have made use of one of the popular ones, that is, Google Text to Speech API (gTTs). It is a convenient tool. The Speech Recognition library retrieves the audio input and is very flexible. The Recognizer class of Speech Recognition recognizes the speech and has many methods that it makes use of, using different APIs, like gTTs. It requires an internet connection but is quite facile.

We would also need an additional tool for the reverse, that is text to speech. The pyttsx module does text to speech conversion and also works offline.

B. MODIFYING THE DATASET:

Any symptom checker would have a dataset of a list of diseases and symptoms as a basis for prognosis. We start out by considering a small dataset, which could be a .csv file and is modified slightly. Since the user input will not be in the exact terms as in the symptom's dataset, we take the root words of the symptoms. Hence, if the user's words have the same root word(s) of that of the dataset, it matches. Taking the root word is done by processes like stemming or lemmatization. They are collectively called as text normalization processes of NLP.

Example: The words 'suffering', 'suffered', 'suffers' have the same root word i.e., 'suffer'.

Stemming deals with reducing the word to its root, even if the root isn't a meaningful word. 'Porter Stemmer' of the 'nltk.stem' package is one of the oldest and popular one that is in use. It is known for its simplicity and speed.

Lemmatization, however, reduces the word to a meaningful root word which is called as lemma. Python NLTK provides 'WordNet Lemmatizer' that uses 'WordNet Database' to lookup lemmas of words. This modified dataset will be kept as a reference for further steps.

C. INTERPRETING THE INPUT:

Here comes the interesting part! Making sense out of the input and obtain the relevant information (i.e. symptoms) that can be made use for classification. The raw input is a string from the speech synthesis. All the actions are performed on this string.

One of the important steps of any NLP method is text tokenization. It involves splitting words or sentences from a body of text into entities. It is carried out by 'word_tokenize' module.

Example: Input – "I am facing severe pain in the chest"

Output – ["I" , " am" , "facing" , "severe" , "pain", "in" , "the" , "chest"]

Stop words are those words that are insignificant and are usually filtered out because they contain a vast amount of unnecessary information. The common ones used in English language are 'as', 'that', 'of' and so on. The 'stopwords' corpus of nltk gives the collection of such words. Hence, they are removed from the tokenized list.

Next, we do what is called as part of speech (pos) tagging. It is also called as grammatical tagging. The goal of a pos tagger is to apply a linguistic tag or information to the tokens. Since we're interested in only extracting the symptom part of the text, which are commonly forms of nouns or adverbs (mostly), we filter in only such tagged words. Our process becomes more efficient step by step.

Some common WordNet methods are applied like that of 'lexeme', which gives the tense forms of a particular word.

Another important factor to be considered is a lexical relation called synonyms. Since the limited dataset obviously cannot cover all the similar meaning words that it contains, we need to simply find the same for the input and compare it with the dataset. The 'synset' package provides a list of synonym words.

Example: Input - "ache"

Output - {'ache', 'aching', 'hurt', 'languish', 'pine', 'smart', 'suffer', 'yearn', 'yen'}

We will also be considering two, three, four or n-tokens together and apply all of the above-mentioned methods, because the symptoms could also be a group of words. They are called as n-grams and n could be any positive integer. The commonly used as bi and tri-grams. They are compared against the dataset.

To be on the safer side, we will use a comparing algorithm on the input n-grams with the dataset symptoms, to get the closest matching words. Such an algorithm is made use in the 'fuzzywuzzy' module of Python. It is based on the Levenshtein distance to calculate the difference between the given two texts or strings. It has a value that is used as a threshold to compare and provides all the closely related words, completing our text parsing.

Several other methods can also be applied to make the output more accurate; however, the efficiency and the time taken must be kept in mind. The output from each and every stage of all the NLP processes is compared with the dataset symptom and any closed match is recorded in an output list.

D. REVIEW:

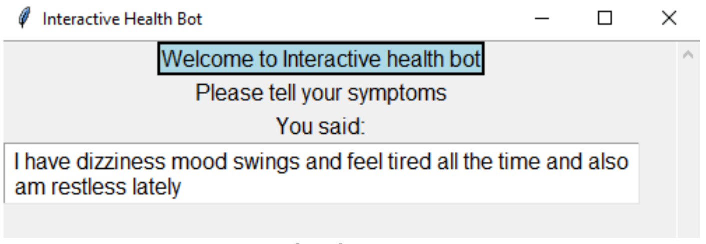
Since we always must err on the right side, the recorded output list of symptoms is cross-checked with the user. A yes/no question-answer type interface is made for each symptom and recorded again in a list. If the output is an empty list, the user is asked if they would want to start over or exit. Finally, the verified output list is the result of the NLP performed on the input and is passed on to further classification processes.

Additionally, a general user interface (GUI) is built with the help of 'tkinter' Python library. The NLP will be running in the backend while a GUI is present in the frontend providing the user a visual experience along with audio updates.

IV. RESULTS

The proposed methodology was implemented using Python (version 3.7.3) and user interface was developed using tkinter package and the results obtained are presented in this section.

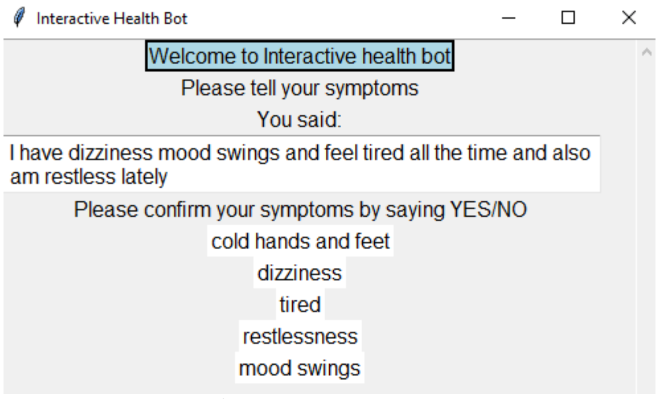
Example 1:



Welcome to Interactive health bot
Please tell your symptoms
You said:
I have dizziness mood swings and feel tired all the time and also am restless lately

Fig.2: Recording the user input

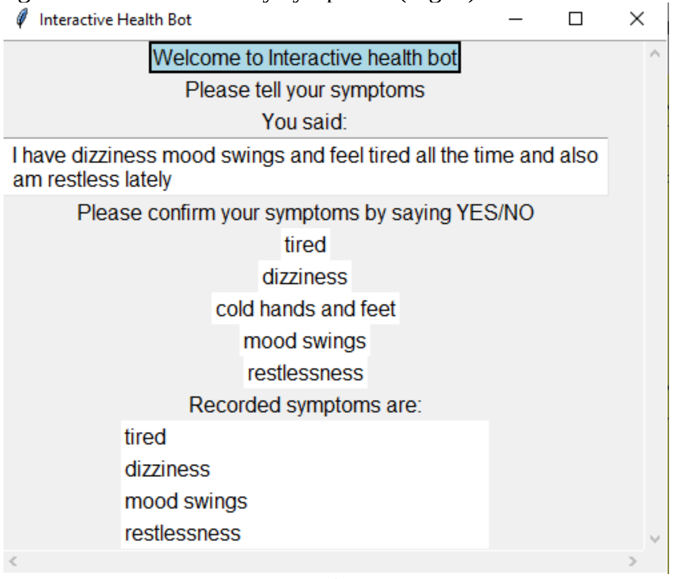
The voice input is taken into the system. Fig. 2 shows an example of an input sentence describing the symptoms of the user.



Welcome to Interactive health bot
Please tell your symptoms
You said:
I have dizziness mood swings and feel tired all the time and also am restless lately
Please confirm your symptoms by saying YES/NO
cold hands and feet
dizziness
tired
restlessness
mood swings

Fig.3: Verifying the extracted symptoms

The input sentence is processed and the possible symptoms are extracted as shown in Fig. 3. The possible symptoms are displayed and asked for confirmation from the user. This step double checks for only the required symptoms and ignores the not necessary symptoms (Fig. 4).



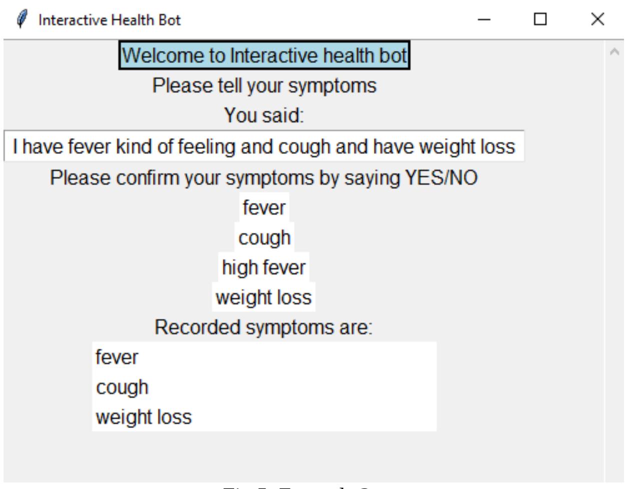
Welcome to Interactive health bot
Please tell your symptoms
You said:
I have dizziness mood swings and feel tired all the time and also am restless lately
Please confirm your symptoms by saying YES/NO
tired
dizziness
cold hands and feet
mood swings
restlessness
Recorded symptoms are:
tired
dizziness
mood swings
restlessness

Fig.4: Displaying the confirmed symptoms

The confirmed symptoms are displayed. These are the symptoms that will be used further for the classification process (This is out of scope of this article).

Example 2:

Here is an extract of another example (Fig. 5) that was considered and the corresponding results obtained.



Welcome to Interactive health bot
Please tell your symptoms
You said:
I have fever kind of feeling and cough and have weight loss
Please confirm your symptoms by saying YES/NO
fever
cough
high fever
weight loss
Recorded symptoms are:
fever
cough
weight loss

Fig.5: Example 2

Upon considering several test cases, the output symptoms were compared with the expected ones and the mean accuracy was found to be 87.53 %. Different variations of the input text were considered and compared with the symptoms that were in our dataset. Fig. 6 shows a plot of a few cases along with its accuracy levels.

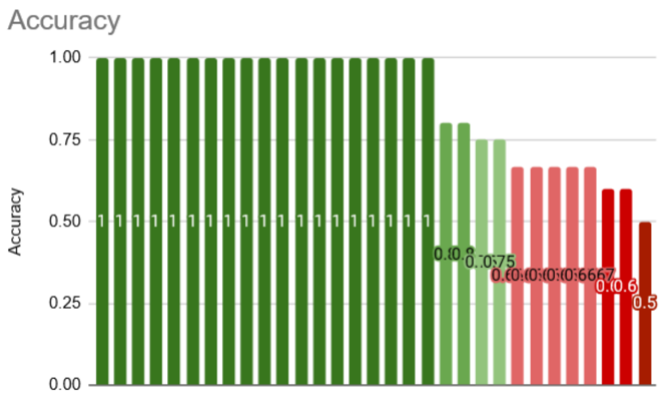


Fig.6: Accuracy plot of the proposed system

Similarly, another statistical analysis is done based on the precision and recall which is the F-measure or the F1 score. It is a better measure to analyze the results since it takes a balance between precision and recall. Hence, it provides a realistic measure of the performance of the system. The Fig. 7 shows the radar chart of the F-measure. It indicates that most of the values are closer or equal to 1 (which is the ideal value), indicating that the system's performance is good.

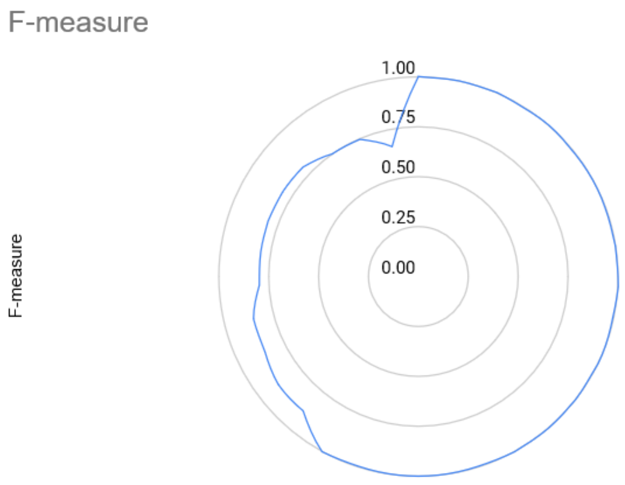


Fig.7: F-measure radar plot of the proposed system

V. CONCLUSION

A system for extracting relevant symptoms from the user voice input was put forth. The primary use of the extracted symptoms is to help in classifying what medical condition the user is facing. The system proposed makes use of many natural language Python packages and implementation steps to parse the input. The testing was done for several cases and the accuracy turned out to be good. Since healthcare is one of the growing industries in the world, our system proves to be a useful method that can be applied to provide a good prognosis prediction. Future work could include various ways to refining the recorded symptoms to improve the accuracy and user dependency.

REFERENCES

[1] Chambers, D., Cantrell, A. J., Johnson, M., Preston, L., Baxter, S. K., Booth, A., & Turner, J. (2019). Digital and online symptom checkers and health assessment/triage services for urgent health problems: systematic review. *BMJ open*, *9*(8), e027743. https://doi.org/10.1136/bmjopen-2018-027743

[2] Emad S. Othman, "Voice Controlled Personal Assistant Using Raspberry Pi", International Journal of Scientific & Engineering Research Volume 8, Issue 11, November-2017.

[3] Arlene E. Chung, Ashley C. Griffin, David Gotz, "Health and Fitness Apps for Hands-Free Voice-Activated Assistants: Content Analysis", JMIR mHealth and uHealth 2017.

[4] Gunjan Dhole and Dr. Nilesh Uke, "Medical information extraction using Natural Language Processing interpretation", Advances in Vision Computing: An International Journal (AVC), Vol.1, No.1, March 2014.

[5] Gustav Cederblad, "Finding synonyms in medical texts – Creating a system for automatic synonym extraction from medical texts", Linköping University, Bachelor thesis, 18 ECTS, Spring term 2018.