Medical Expert System

Group Members

1. Prasann Patel - 100833325
2. Samarth Patel - 100827936
3. Shail Patel - 100846377
4. Tejas Devani - 100846988

Github Link: https://github.com/Shail079/Medical-Expert-System

Introduction

To let users know about a disease which they suffer, we came up with the idea to develop an expert system that is named **Medical Expert System.** This system can be used for diagnosing respiratory diseases, and also can be used for detecting illnesses like flu or cold.

Task and Purpose of the Expert System

Medical Expert Systems have come into existence to assist Physicians and Patients in the diagnosis and treatment of diseases and even to send alerts and provide notifications. Also, having the ability to assist their users at any time could improve their quality of life. Information technology and clinical knowledge are then intended to improve healthcare safety, efficiency and reliability through their synergy. The fact that Healthcare is a sensitive and constantly evolving field, requires the expert system to be updated, expanded or modified in line with the changing knowledge base and the rules.

Knowledge Sources and Knowledge Acquisition

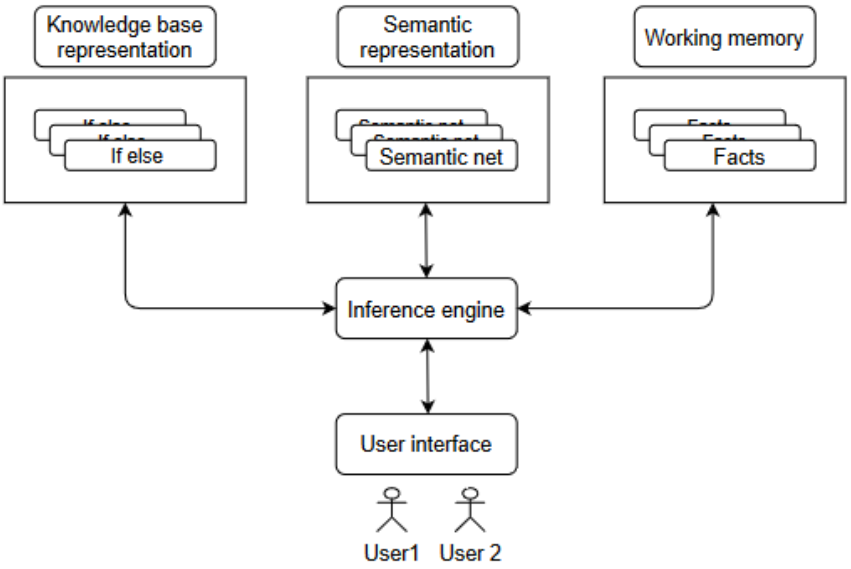
Domain knowledge is vital for any knowledge based expert system. The process of acquiring knowledge involves a human expert, a group of experts, interviews, internet sources or books. We explored the internet sources such as various government resources and any healthcare web resources to gather common symptoms of cold and flu.

There are some resources listed below from where we have acquired the domain knowledge.

- https://lunghealth.ca/lung-disease/a-to-z/common-cold/
- https://lunghealth.ca/lung-disease/a-to-z/flu/?gclid=EAIaIQobChMI_d3h7Z-R9wIVDFNyCh2ivQEZEAAYAiAAEgJDj_D_BwE
- https://www.cdc.gov/flu/symptoms/symptoms.htm
- https://www.cdc.gov/features/rhinoviruses/index.html

Knowledge Design and Engineering

In this Project, we aim to develop a generic knowledge-based medical expert system. This medical expert system has the scope of representing some major diseases' symptoms and properties. One of the main focuses of this medical system is to develop a graphical user interface to visualize decision-making and provide a knowledge-based GUI edit.



An expert system is composed of two major parts, a knowledge base and an inference engine. The inference engine reasons about the knowledge base like a human to run an expert system. A dialogue interface is used to communicate between users and the system, which has the ability to conduct a conversation with users via "conversational" interface. This system has all of the necessary components that an expert system must have. The major components are:

- **Knowledge base** - A declarative representation of the expertise, can either be the IF-THEN rules based or knowledge base representation such as semantic net and frame based representation.
- **Inference engine** - It is the core of the system, which derives recommendations from the knowledge base and problem-specific data in working memory.

- **Working memory** - It is used to store user inputs and for further matching queries, which help to not ask duplicate questions.
- **User interface** - It is the interface that controls the interaction between the user and the core expert system. This component has been built using python in the terminal.

1. Knowledge structure

The knowledge construction was built on a concept of rules that were performed in a hierarchical tree structure. For this part, rule based knowledge with a hierarchical tree structure is used to represent the knowledge about different diseases and its symptom. This system has its own internal rule format with header information and all premises are connected by AND or OR conditions. This format has a section for representing a certainty factor of rules. This certainty factor will be used by the inference engine to make decisions about a possible solution. The rule can be nested, which means a rule can have nested sub-goal rules and different facts.

2. Inference Engine

An inference engine is used to derive answers from the knowledge base. A finite state machine in the inference engine can be described with a cycle of three action states as match rules, select rules, and execute rules. Rules are designed to be a tree-like hierarchical structure to represent the knowledge about different diseases and their symptoms. With the first state of match rules, the inference engine searches all of the rules that are satisfied by the current contents of the data target. In the second state, the inference engine applies some selection strategy to determine which rules will actually be executed. Finally, the inference engine executes or fires the selected rules, with the instantiating data items as parameters.
Using this format of rules, the inference engine of this system can reason knowledge about diseases by several symptoms. The main desired behaviours of the inference engine are:

- combine certainty factors as indicated previously.
- maintain working memory that is updated as new evidence is acquired.
- locate all information specific to an attribute when it is asked for, and store that information in working memory.

Backward chaining approach

The system works backward from the ultimate goal until all the sub-goals in working memory are known to be true, indicating that the hypothesis has been verified. This process is known as “backward chaining”.

Steps in backward chaining: System has a hypothetical solution(s) (e.g. “The patient has type I diagnosis(X)”), and tries to prove it.

- Find rules that match the goal.
- If antecedents match the facts, stop.
- If not, make the antecedents the new sub-goals, and repeat.

Certainty Factors: The most common scheme for modelling uncertainty is to assign a certainty factor to each piece of information in the system. The inference engine automatically updates and maintains the certainty factors as the inference proceeds. The certainty factors (CF) are integers from 0 (for definitely false) to 1 (for definitely true). The rule format also allows for the addition of certainty factors. When the premise of a rule is uncertain due to uncertain facts, and the conclusion is uncertain due to the specification in the rule, the following formula is used to compute the adjusted certainty factor of the conclusion:

$$CF = RuleCF * PremiseCF$$

3. Working memory

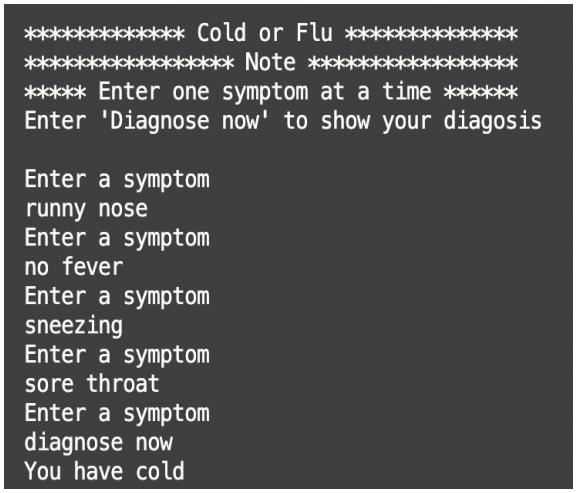
One can treat working memory as the brain processes used for temporary storage and manipulation of information. It operates over a short time window which is usually a few seconds, and it facilitates users to focus attention, resist distractions, and guide the decision-making.

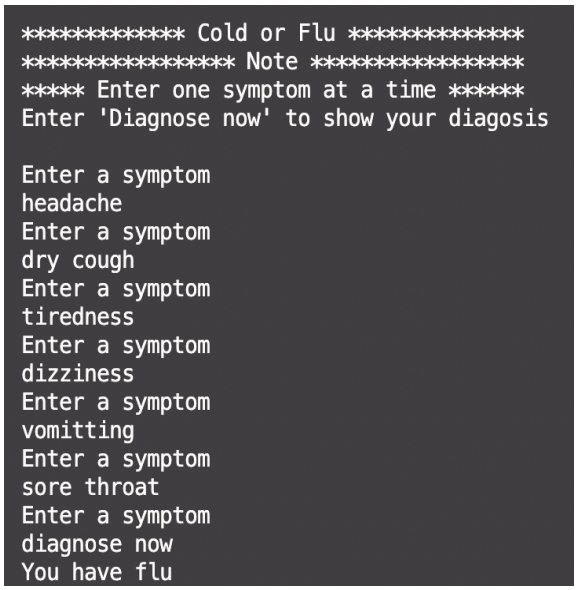
Working memory simply contains the known facts about attribute-value pairs. Known facts, represented as $Known(Goal, CF)$, are stored in the Prolog database for further use. If a current fact and its certainty fact have been already stored in working memory, the system would not ask users about it further and it will calculate the certainty fact using the stored data.

User Interface

Interaction between user and system is done using the command-line interface. The expert system requires users to enter symptoms one by one. The user needs to enter “Diagnose now” to show the diagnosed disease. Users are needed to input their symptoms in their terminal, For Example: If the person is having a fever right now, similarly if the person is suffering from headache, sore throat or any other symptoms, then the user needs to enter those symptoms in the terminal. The user is needed to input **at least 3 symptoms** that he/she is suffering for getting an accurate diagnosis of the disease. Thus, after successfully entering the symptoms the user needs to enter ‘Diagnose now’ and then It will print “You have cold” or “You have flu”.

Our system’s output will be shown in the below images.

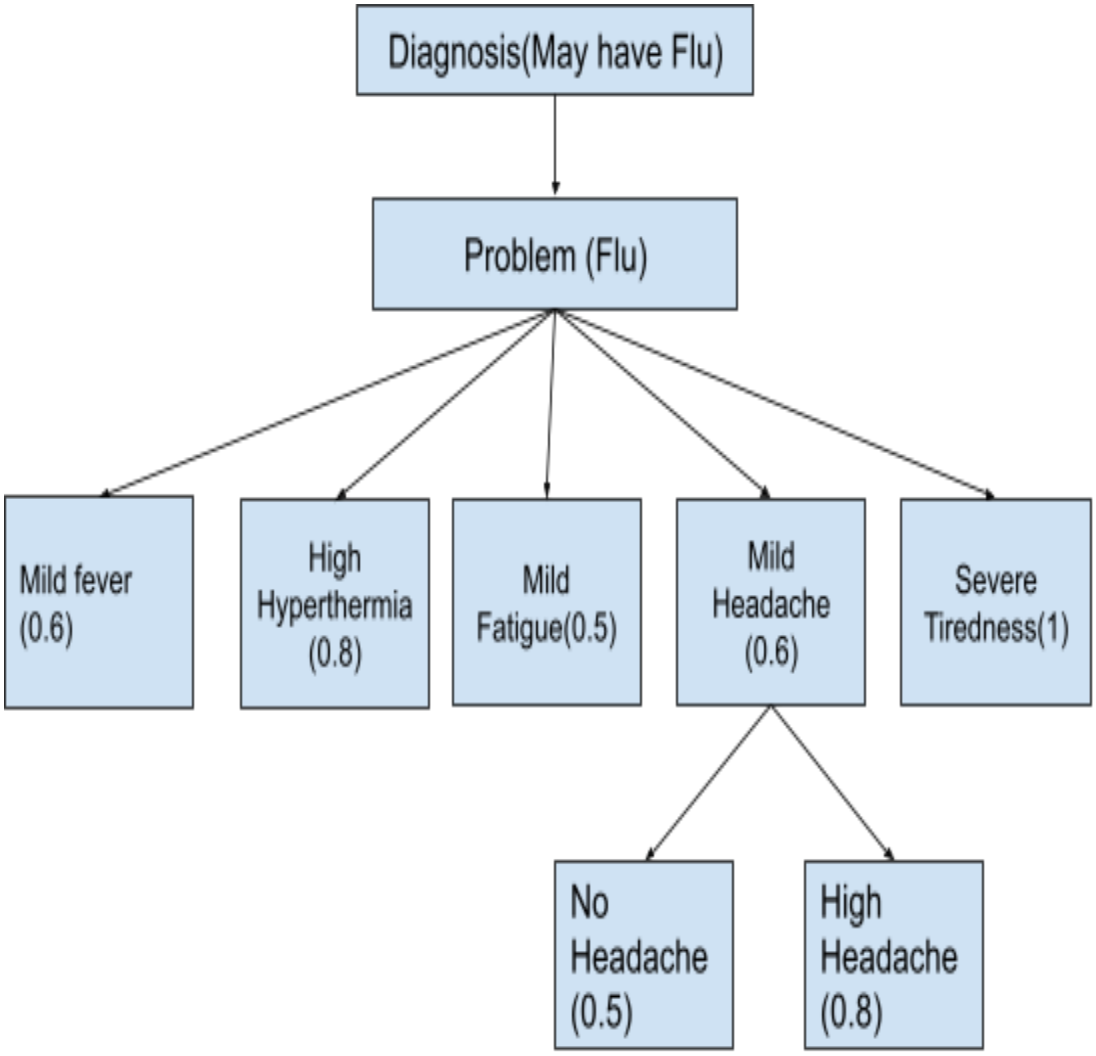




Framework and Implementation

To implement the rule-based expert system we have used **pyknow** library. We implemented an interface engine by defining rules for different symptoms and then we will run the interface engine to make a decision based on predefined rules.

Languages: Python
IDE: Google Colab
Third-party libraries: PyKnow



Graphical Visualization of Diagnosis of Flu with Certainty Factor

```
# Output
flu_symptoms = 0
cold_symptoms = 0
print("")
print(" Cold or Flu ".center(40, "*"))
print(" Note ".center(40, "*"))
print(" Enter one symptom at a time ".center(40, "*"))
print("Enter 'Diagnose now' to show your diagosis ".center(40, "*"))
print(" ")
engine = DiagnoseDisease()

while(1):
    print("Enter a symptom")
    symptom = input()
    # Exit from loop if the input equal " i do not have"
    if symptom.strip().lower() == "diagnose now":
        break
    engine.reset()  # Prepare the engine for the execution.
    engine.declare(Fact(symptom=symptom.strip().lower()))
    engine.run()  # Run it

# Check if user enter less than 3 symptoms
if ((flu_symptoms < 3) and (cold_symptoms < 3)):
    print("You must enter at least 3 symptoms")
# Cann't diagnose user state when the flu and cold symptoms are equal
elif(flu_symptoms == cold_symptoms):
    print("Can't diagnose your state")
# If the flu counter greater than cold counter, then the user has flu
elif ((flu_symptoms > 3) and (flu_symptoms > cold_symptoms)):
    print("You have flu")
# If the cold counter greater than flu counter, then the user has cold
elif ((cold_symptoms > 3) and (flu_symptoms < cold_symptoms)):
    print("You have cold")
```

Group Members' Task

Member	Task
Prasann Patel	Project Ideation, Knowledge Acquisition, Planning, Presentation
Samarth Patel	Project Report, Presentation
Shail Patel	Knowledge Representation, Rule Definition, Report, Coding
Tejas Devani	Knowledge Acquisition, Rule Definition, Report, Testing