MINISTRY OF EDUCATION OF REPUBLIC OF MOLDOVA

Technical University of Moldova

Faculty of Computers, Informatics and Microelectronics

Department of Software Engineering and Automatics

Report

on Artificial Intelligence Fundamentals

Laboratory Work nr. 1

Performed by: **Vasile Boaghi, Ion Dodon**, FAF-172

Verified by: **Mihail Gavrilița**, asist. univ.

Chișinău, 2021

Contents

Expert Systems

Expert systems (ES) are one of the prominent research domains of AI. It is introduced by the researchers at Stanford University, Computer Science Department.

The expert systems are the computer applications developed to solve complex problems in a particular domain, at the level of extra-ordinary human intelligence and expertise.

Characteristics of Expert Systems:

- High performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems:

The expert systems are capable of

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem

They are incapable of

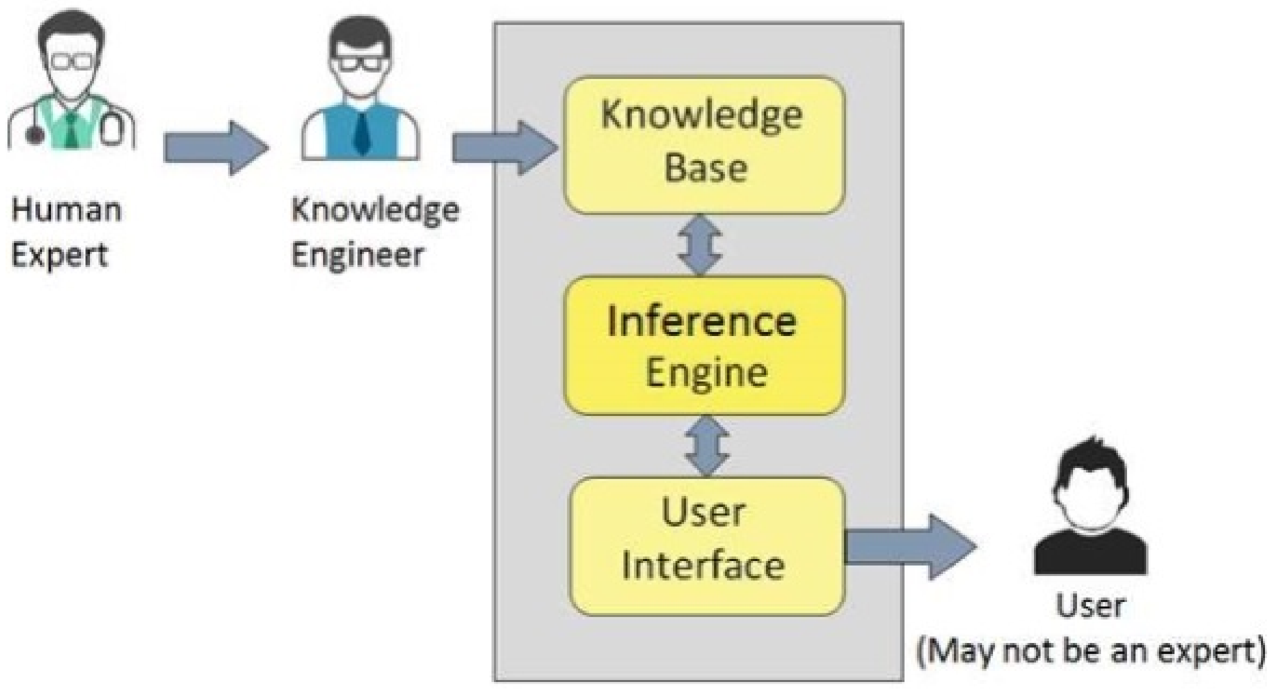
- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Components of Expert Systems

The components of ES include

- Knowledge Base
- Inference Engine
- User Interface

Figure 1: Components of Expert Systems



Knowledge Base

It contains domain-specific and high-quality knowledge.

Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

Inference Engine

Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution.

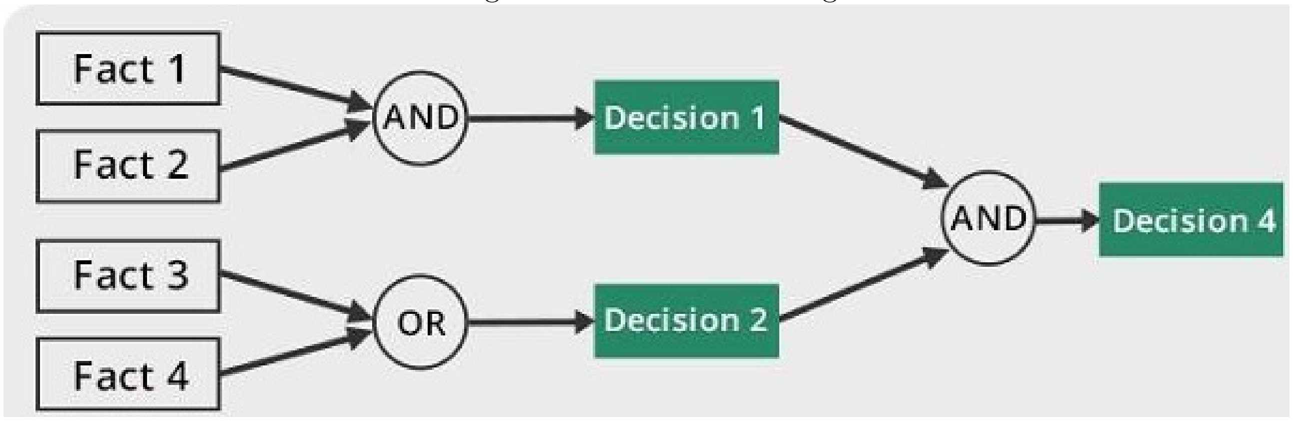
In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

Forward Chaining

It is a strategy of an expert system to answer the question, “What can happen next?”

Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

Figure 2: Forward Chaining



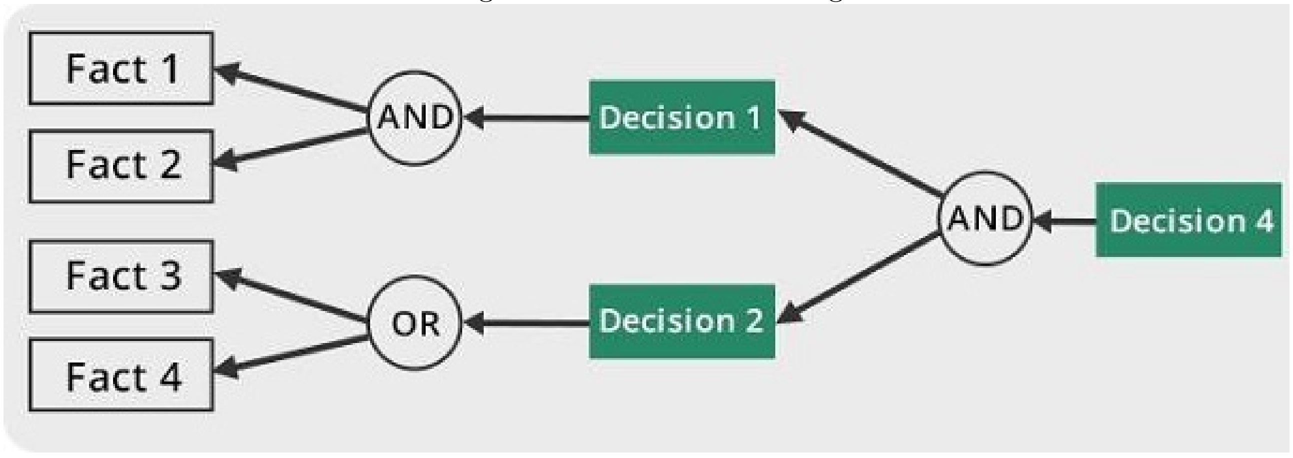
This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.

Backward Chaining

With this strategy, an expert system finds out the answer to the question, "Why this happened?"

On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.

Figure 3: Backward chaining



User Interface

User interface provides interaction between user of the ES and the ES itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.

The information from above is taken from tutorialspint.com [2]

The Task

Fighting with sapient rocks

The task of this laboratory work is to create an expert system that will be able to detect the types of tourist that are at the Luna City.

It is needed to create a system that would allow the user to answer some questions about a possible tourist. If the set of given answers matches a type of tourist from the database, this should be the system's answer. If on the other hand, the system determines that the person in question is a Loonie, the answer should be returned accordingly. Make sure to consider the case when the set of answers does not find a match in the database (highly improbable if you've done your research well).

Another thing to consider is how the system will be deployed. The machines that are at your disposal are only capable of running Python code or Docker images. So if you choose to write your system in something exotic, like Prolog, make sure to package it all accordingly. To aid you in your task, the mentor gave you some ancient "lab" document and some Python code that he found in the library.

Solution Description

We used Python to create our Inference Engine for the expert system. The engine will determine the type of tourist by traversing through the and/or tree backwards and when it reaches the leaves the engine will ask questions. Each one of the tree we name is a fact. The leaves of the tree are kind of primitive facts that can be considered questions and can be answered with yes or not.

Before using the engine to traverse the tree in order to find the types of tourists, first, the system should read data from a *csv* file, which in our case is the database. The **csv** file will contain three columns: first column will represents a child node, the second column will be the parent of this child node, and the third column represents the operand - the relation that should be respected in order for the child node to be valid for the parent node. This operand can be **or** or **and**.

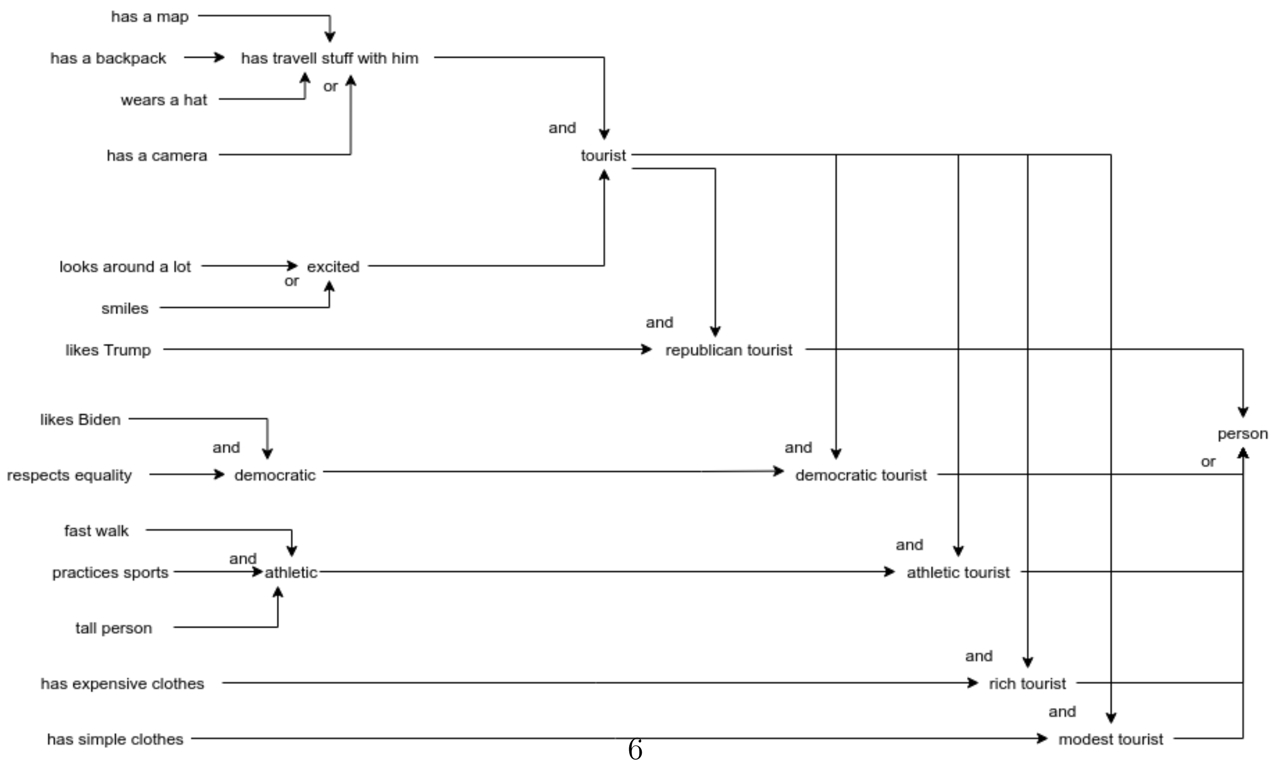
In table 1 we can see the data itself and in figure 4 we can see the structure of the tree.

The root of the tree will be a node - fact name **person**. From this fact the backward traversing will begin in order to find the types of tourists. It table 4 we can see the structure of the tree.

Table 1: Data from the csv file.

child	parent	operand
has a map	has travel stuff with him	
has a backpack	has travel stuff with him	
wears a hat	has travel stuff with him	
has a camera	has travel stuff with him	
has travel stuff with him	tourist	or
looks around a lot	excited	
smiles	excited	
excited	tourist	or
likes Trump	republican tourist	
tourist	republican tourist	and
tourist	person	and
republican tourist	person	and
likes Biden	democratic	
respects equality	democratic	
democratic	democratic tourist	and
democratic tourist	person	and
fast walk	athletic	
practices sports	athletic	
tall person	athletic	
athletic	athletic tourist	and
athletic tourist	person	and
athletic tourist	person	and
has expensive clothes	rich tourist	
rich tourist	person	and
has simple clothes	modest tourist	
modest tourist	person	and
person		or

Figure 4: The tree structure



Code and Mentions

Each fact is represented as a class in Python. This class has the following attributes: *fact_name*, *parent_fact_name*, *child_facts*, *operand*. Is a fact does not have an operand and it has 0 child fact then it is a primitive fact that can be a question answered with yes or no.

Listing 1: The fact class

```python
class Fact:
    fact_name = None
    parent_fact_name = None
    child_facts = []
    operand = None

    def __init__(self, fact_name, parent_fact_name, operand):
        self.fact_name = fact_name
        self.parent_fact_name = parent_fact_name
        self.operand = operand
```

The *read_knowledge()* and *set_fact(new_fact)* functions are used to read the data from the csv file and create the tree - knowledge. The knowledge will be a dictionary in which the key will be the fact name and the value will be an object of the type Fact.

Listing 2: Function to create the knowledge

```python
def set_fact(new_fact):
    if not (new_fact.operand is None):
        children_list = get_children_list(new_fact)
        new_fact.child_facts = children_list

    knowledge[new_fact.fact_name] = new_fact


def read_knowledge():
    with open('data.csv', newline='') as csv_file:
        reader = csv.reader(csv_file, delimiter=',')
        for raw in reader:
            fact_name = raw[0] if raw[0] != '' else None
            parent_fact_name = raw[1] if raw[1] != '' else None
            operand = raw[2] if raw[2] != '' else None
            new_fact = Fact(fact_name, parent_fact_name, operand)
            set_fact(new_fact)
```

The *ask(fact_name)* function is used to answer the questions of type - primite facts.

Listing 3: Function to answer questions

```python
def ask(fact_name) -> bool:
    response = input(fact_name + "? (y/n): ")
    return True if response == 'y' else False
```

The *check_fact(fact_name)* is one of the most important ones, because it is used to traverse the tree backwards. It is the *Inference Engine*.

Listing 4: Function to traverse the tree

```python
def check_fact(fact_name) -> bool:
    fact_state = True
    fact = knowledge[fact_name]
    if fact.operand == "and":
        for child_fact_name in fact.child_facts:
            fact_state = fact_state and check_fact(child_fact_name)
        return fact_state
    elif fact.operand == "or":
        aux = False
        for child_fact_name in fact.child_facts:
            next_fact_status = check_fact(child_fact_name)
            if next_fact_status is True:
                print(fact_name + ", because " + child_fact_name)
                aux = True
        return aux
    else:
        return fact_state and ask(fact_name)
```

Conclusions

Working on this laboratory work we have understood what an Expert System is what are the main component of an expert system: Knowledge Base, Inference Engine, User Interface; and how to implement a simple one. By doing this laboratory work we've practiced how to collect data and with it to create our decision tree, then we thought how to create such an algorithm for our tree so that it will be able to ask question and based on answers to determine specific fact about the given topic.

We have come to the conclusion an expert system does not have special that would make it intelligent, instead is just follows rules, conditions and using a database of knowledge it can offer specific information. To make a better "Expert System" one must have a better database with more data and well defined rules.

References

[1] Vasile Boaghi, Ion Dodon. Source code for the laboratory work. Accessed February 9, 2021. https://github.com/iondodon/FIA/tree/main/Lab1.

[2] tutorialspoint.com https://www.tutorialspoint.com