1/8/22

Page No.
Date 1 8 22

★ What is human Intelligence ?

⇒ It is composition of abilities like

1) Reasoning
2) Learning
3) Problem Solving
4) Perception
5) Linguistic Intelligence

★ Artificial Intelligence

1) AI is a simulation of human intelligence by machine means.

OR 2) AI is the study of how to make computers do things which at the moment people do better.

OR 3) AI is a branch of computer science concerned with the study and creation of computer system that exhibit some form of intelligence ; Systems that learn new concept and tasks. System that can reason and draw useful conclusion about the world around us, Systems that can understand a natural language of perceive and comprehend a visual scene and System that perform other type of tasks (facts) that require human type of intelligence.

Page No.
Date 2 8 22

2/8/22

- Task domain of AI (R/k)

Types of domain

1) Mundane Task
   - perception
     - vision
     - Speech
   - Natural language
     - Understanding
     - Generation
     - Translation
   - Common sense Reasoning
   - Robot control

2) Formal Task
   - Game playing
     - chess
     - Backgammon
     - checkers
     - Go
   - Mathematics
     - Geometry
     - Logical programming
     - Integral calculus
     - properties of programs

3) Expert Task

- Engineering
  - Design
  - fault finding,
  - manufacturing
  - planning
- Scientific Analysis
- medical dignosis system
- Financial analysis.

★ State Space Search

To build a system to solve a perticular problem following are the steps:

1) Define the problem presizely
2) Analyze the problem
3) Isolate and Represent the tusk knowledge i.e. necessary to solve problem
4) choose the best problem solving technique & apply it to perticular problem

6/9/22

* Water Jug Problem:

You are gives 2 Jugs of 4 gallon and 3 gallon Jug nither has any measuring measures on it. There is a pump that can be used as file the jugs with water. how can you get exactly 2 gallon of water into the 4 gallon jug

⇒ The state space for this problem can be distributed as the set of the order pairs of integers (x,y) such that

$x = 0, 1, 2, 3, \text{ or } 4$

$y = 0, 1, 2 \text{ or } 3$

where x represent no. of gallon of water is 4 gallon jug and y represent no. of gallon of water is 3 gallon Jug

The initial state is (0,0), The goal state is (2,n) for Any value of n = 0, 1, 2, 3, ...

- **Production Rule**

Rule 1 :- $(x, y) \rightarrow (4, y)$ fill the 4 gallons Jug if $x, y < 4$.

~~Rule 2~~ :-

Rule 2:- $(x,y) \rightarrow (x,3)$ if $y<3$ — fill the 3 gallon Jug

Rule 3:- $(x,y) \rightarrow (x-d,y)$ if $x>0$ — Pour some water out of 4 gallon Jug

Rule 4:- $(x,y) \rightarrow (x,y-d)$ if $y>0$ — Pour some water out of 3 gallon Jug

Rule 5:- $(x,y) \rightarrow (0,y)$ $x>0$ — Empty the 4 gallon Jug on the ground

Rule 6:- $(x,y) \rightarrow (x,0)$ $y>0$ — Empty 3 gallon Jug on the ground

Rule 7:- $(x,y) \rightarrow (4, y-(4-x))$ if $x+y \geq 0$ & $y>0$ — Pour water from the 3 gallon Jug into 4 gallon Jug untill 4 gallon Jug is full

Rule 8:- $(x,y) \rightarrow (x-(3-y), 3)$ if $x+y \geq 3$ and $x>0$ — Pour water from 4 gallon Jug into 3 gallon Jug untill 3 gallon jug is full

Rule 9:- $(x,y) \rightarrow (x+y, 0)$ if $x+y \leq 4$ & $y>0$ — Pour all the water from 3 gallon jug into 4 gallon jug

Page No.
Date

Rule 10 :- $(x,y) \rightarrow (0, x+y)$ if $x+y \leq 3$ and $x > 0$ — Pour all the water from 4 gallon jug into 3 gallon jug

Rule 11 :- $(0,2) \rightarrow (2,0)$ — Pour the 2 gallon from 3 gallon into 4 gallon jug

Rule 12 :- $(2,y) \rightarrow (0,4)$ — Empty the 2 gallon on 4 gallon jug on the ground.

Gallon in the 4-g Jug	Gallons in 3-g Jug	Rule Application
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

8/8/22

Production system.

1) A set of Rules
2) One or more knowledge
3) control strategy
4) Rule Applier

Control Strategy

Q. what do you mean by production system?
Q. what do you mean by control strategy & requirements of control strategy?
Q. differentiate of depth first search & breadth first search

The control strategy is one that specifies the order in which the rules will be compared to the database and a way of resolving conflex that arise when several rules matched at once.

Requirement

1) It cause motion
2) It should be systematic

* ~~Heu~~

18/8/22

Differentiate betn blind search & Heuristic search
uninformed informed

* problem characteristics ~~of production system~~

1. Is the problem decomposable into aset of independant or easier subproblems.
2. Can solution steps we ignored or atleast undone?
3. is the problem's universe predictable?
4. is a good solution to the problem obvious without comparison to all other possible solutions

(1) chair → Block wood problem is not decomposable

Integration ~~differentiation~~ is decomposable.

(5) is the desired solution a state of the world or path to a state. ?

(6) is a large amount of knowledge absolutely required to solve the problem or is knowledge important to constrain a search

(7) Can a computer i.e. simply given the problem written the solution or will the solution of problem require interaction betn the computer & a person.

19/8/22

Q Analyze the problem with 7 problem characteristics.

problem.

2nd → Solution are of three types

1) Ignorable (eg. Theorem proving)
2) Recoverable (eg. Eight puzzle)
3) Irrecoverable (eg. chess)

Ignorable - In which solution steps can be ignored.

Recoverable - In which solution steps can be undone.

Irrecoverable - In which solution steps can not be undone

3rd ch → Universe predicatable

1) Certain outcome - 8-puzzle
2) Uncertain outcome - Chess, Bridge

4th ch → Is the good solution as absolute or relative

① Any path problem - facts., 8-puzzle ← absolute answer
② Best path problem - Travelling salesman

5th ch → is the solution a state or a path?

1) Problem whose solution is a state of the world and - Natural language understanding
2) Problems whose solution is a path to a state - Water Jug problem

6th ch → What is the role of knowledge?

1) problems for which a lot of knowledge is important only to constrain the search for a solution - chess
2) Problems for which a lot of knowledg is required even to be able to recognize a

What are the diff^n types of production system characteristics?

Page No.
Date

Solution

25/09/22

7th ch- Does the task required interaction with a person

1) Solitory

⇒ In which computer is given a problem discription and produces an answer with no intermediate communication and with no demand for explanation of the reasoning process.

2) Conversation.

⇒ In which there is intermediate communication between a person & the computer, either to provide assistants to the computer or to provide additional information to the user or both.

Production System Characteristics

	Monotonic	Non-monotonic
Partially Commutative	Theorem proving	Robot Navigation
Not-partially commutative	chemical Synthesis	Bridge

The four categories of production system

★ Monotonic Production System

It is a production system in which the application of rule never prevents, the later application of another rule that could also have been applied at the time rule was selected.

★ Non-monotonic production system

A nonmonotonic production system is one in which these is not true

★ Partially commutative System

A partially commutative production system is a production system with the property that if the application of perticular sequence rules transformation state x into state y, then any applic permutation of those rules that is allowable also tranforms state x into state y

★ Commutative production System

A commutative production system is a production system i.e. both monotonic & partially commutative

30/08/

Date

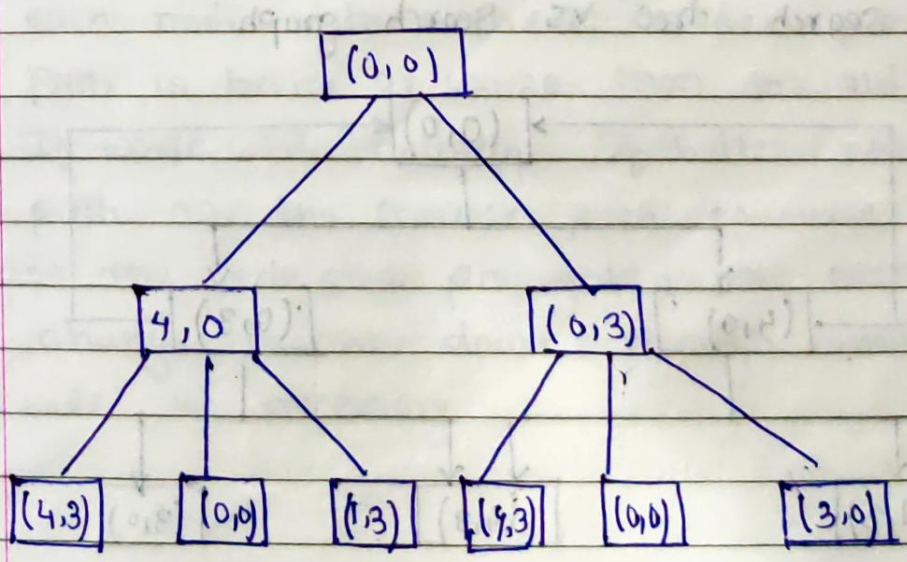
⇒ Partially commutative monotonic production system are useful for solving ignorable problems

⇒ They are good for problems where things do not change., New things get created.

⇒ Non-monotonic partially commutative system are useful for problems in which changes occur but can be reversed and in which ordered of operation is not critical.

⇒ production systems that are not partially commutative or useful for many problems in which irreversible changes occur.

* Issues in design the search problem :



Search Tree for water Jug problem

Every search process can be viewed as a traversal of a tree structure in which each node represents a problem state and each arc represents a relationship betn the states represented by the nodes it connects.

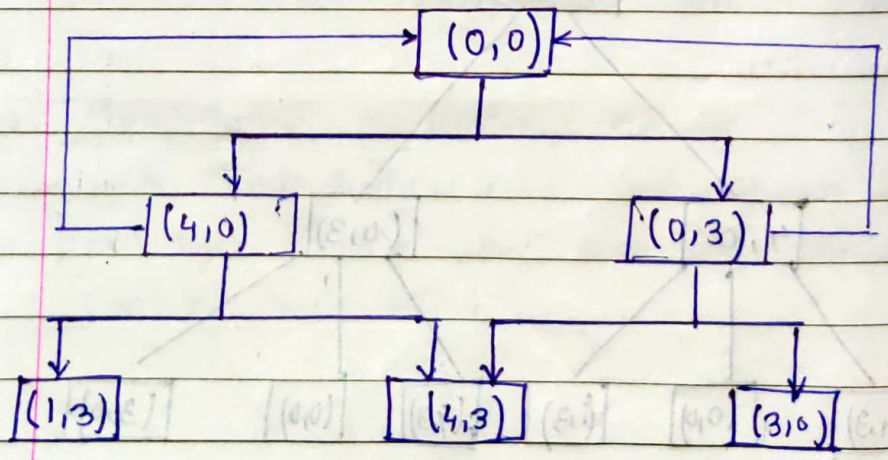
* Some of the important issues that arrise in the design of search program.

1) The direction in which to conduct the search (forward vs backward reasoning)

2) How to select applicable rules (matching)

3) How to represent each node of the search process.

4) Search tree vs Search graph



Search graph for water jug problem

Algorithm : check duplicate Nodes.

1- Examine the set of nodes that have been created so far to see if new node already exists.

2. If it does not, simply add it to the graph just as for a tree

3. If i does already exists, then do the following

a) Set the node that is being expanded to the point to the already existing node corresponding to its successor rather than to the new one The new one can simply be thrown away

b) If you are keeping tracks of the best path to each node, then check to see if the new path is better or worse than the old one. If worse, do nothing. If better, record new path as the current path to use to get to the node & the propagate the corresponding change in cost down through successor. node as necessary.

UNIT - 2

Page No.
Date

* Heuristic Search Problem (Rule of Thumb)

1) Generate - and - test
2) Hill climbing
3) Best-first Search
4) Problem Reduction
5) constraint satisfaction
6) Means - ends analysis

1] Generate - & - test

Algorithm :

1. Generate a possible solution. for some problems this means generating a particular point in the problem space, for others, it means generating a path from the start state.

2. Test to see if this is actually a solution by comparing the chosen point or endpoint of the chosen path to the set of acceptable goal states

3. If a solution has been found, quit. Otherwise, return to step 1.

2/9/22

1) What is difference betn generate & test & Hill climbing
2) Which of the problem encountered in Hill climbing?

Page No.
Date

2] Hill climbing (No back tracking)

⇒ Hill climbing is a ~~very~~ variant tough generate & test in which feedback from the test procedure is used help the generator decide which direction to move in the search space. In a pure generate & test procedure, the test function responds with only a yes or no.

Algorithm :- Simple Hill climbing

1. Evaluate the initial state. If it is also a goal state, then return it & quit. otherwise continue with the initial state as the current state.

2. Loop until a solution is found or until there no new operators left to be applied in the current state

a) Select an operator that has not yet been applied to the current state & apply it to produce a new state

b) Evaluate new state.
   i. If it is a goal state, then return & quit

ii. If it is not a goal state but it is better than the current state, then make it the current state.

iii. If it is not better than the current state then continue in the loop.

Algorithm :- Steepest-Ascent Hill climbing

1. Evaluate the initial state. If it is also a goal state., then return it & quit. otherwise continue with the initial states as the current state.

2. Loop until a solution is found or until a complete iteration produces no change to current state:

a) Let succ be a state such that any possible successor of the current state will be better than succ.

b) For each operator that applies to the current state. do:

i. Apply the operator & generate a new state.

4/9/22

Page No.
Date

- Both basic /simple and steepest ascent hill climbing may fail to find a solution. Either algorithm may terminate not by finding a goal state but by getting to state from which no better states can be generated.

- This will happen if the program has reached either a local maximum, a plateau, or a ridge.

1) Local Maximum

It is a state i.e. better than all its neighbour but is not better than some other farther away. At a local maximum, all moves appear to make things worse. ~~local maxima~~.

Solution - Back track to some earlier node and try going in a different direction.

2] Plateau -

It is a flat area of the search space in which whole set of neighbouring states have the same value, on a ~~plateau~~ plateau, it is not a possible to determine the best direction in which to move by making local comparison

Solution :- Make a big jump in some direction to try to get to a new section of a search space.

3] Ridge

A ridge is special kind of local maximum. It is an area of search space that is heigher than surrounding areas and that itself has a slope. (which one would like to climb)

Solution - Apply the two or more rules before doing the test. This corresponds to moving several direction at once.

13/09/22 3) Best - First search

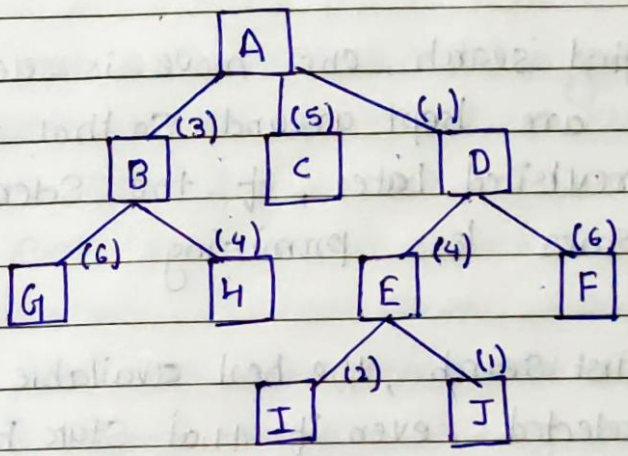
→ DFS is good because it allows a solution to be found without all competing branches having to be expanded.

→ BFS is good because it does not get trapped on dead-end paths.

→ One way of combining a two is to follow a

single path at a time, but switch paths whenever some competing path looks more promising than the current one does.

eg



OPEN1: A

CLOSED1: A

OPEN 2: B, B, C

CLOSED 2: A

OPEN 3: B, E, C, F

CLOSED 3: A, D

OPEN 4: E, C, H, F, G

CLOSED 4: A, D, B

OPEN 5: I, C, H, F, G

CLOSED 5: A, D, B, E

* OPEN and CLOSED list is arranging according to the priority criteria.

What is difference betn the hill climbing & Best first search

→ In hill climbing in one move is selected & all the others are rejected, never to be reconsider

→ In best first search, one move is selected but the others are kept around so that they can be revisited later, if the selected path becomes less promising

→ In best first search, the best available State is selected, even if that State has a value that is lower than the value of State that was just explored

→ In hill climbing search process will stop if there are no succussor states with better values than the current state.

• To implement a graph search procedure, two list of nodes are maintain.

1) OPEN :- Nodes that have been generated & have had the heuristic function applied to them but which have not yet examined. (priority queue)

CLOSED :- nodes that have already been examined

16/9/22 Algorithm : Best-first search.

1) Start with OPEN containing just the initial state.
2) Until a goal is found or there are no nodes on OPEN do :
   a) Pick the best node on OPEN.
   b) Generate its successors.
   c) For each successor do :
      i) If it has not be generated before, evaluate it add it to OPEN, and record its parent.
      ii) If it has been generated before, change the parent if this new path is better than the previous one. In that case, update the cost of getting to this node and to any successors that this node may already have.

* Best-first search is the simplification of A* algorithm.

=> OR Graph is used

Page No.
Date

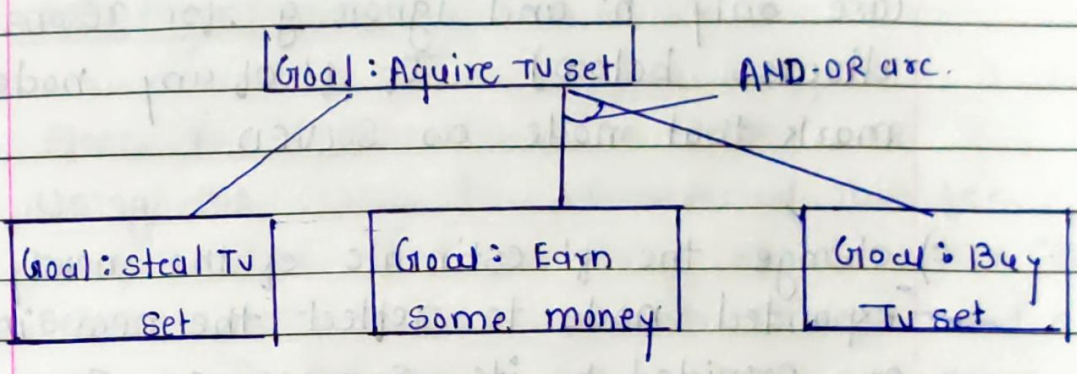
A* algorithm

1. Place the starting node s on open.
2. If open is empty, stop and return failure.
3. Remove from open the node n that has the smallest value of $f^*(n)$. If the node is a goal node, return success and stop. otherwise,
4. Expand n, generating all of its successor n' and place n on closed. For every successor n', if n' is not already on open or closed attach a back-pointer to n, compute $f^*(n')$ and place it on open
5. Each n' that is already on open or closed should be attached to back-pointers which reflect the lowest $g^*(n')$ path. If n' was on closed and its pointer was changed, remove it and placed it on open.
6. Return to step 2.

29/09/22

4] Problem Reduction

⇒ AND - OR Graph.



AND-OR Graph

4/10/22

Algorithm : ~~Production~~ Problem Reduction.

1. Initialize the graph to the starting node
2. Loop until the starting node is labeled SOLVED or until its cost goes above FUTILITY.

a) Traverse the graph, starting at the initial node and following the current best path, and accumulate the set of nodes that are on that path and have not yet been expanded or labeled as solved.

b) Pick one of these unexpanded nodes and expand it. If there are no successors, assign

FUTIIIY as the value of this node. Otherwise, add its successors to the graph and for each of them compute f' (use only h' and ignore g, for resons we discuss below). If f' of any node 0, mark that node as SOLVED.

c) change the f' estimate of the newly expanded node to reflect the new informati. on provided by its successors. Propagate this change backward through the graph. If any node contains a successor arc whose descendants are all solved, label the node itself as SOLVED. At each node that is visited while going up the graph, decide which of its successor arcs is the most promising and mark it as part of the current best path. This may cause the current best path to change. This propagation of revised cost estimates back up tree was not necessary in the best-first search algorithm because only unexpanded nodes were examined. But now expanded nodes must be re-examined so that the best current path can be ~~solved~~. selected. Thus it is important that their f' values.

be the best estimates available

4/10/22

★ AO* Algorithm

1. Place the start node s on open.
2. Using the search tree constructed thus far, compute the most promising solution tree To
3. Select a node n that is both on open and a part of To. Remove n from open and place it on closed.
4. If n is a terminal goal node, label n as solved. If the solution of n results in any of n's ancestors being solved, label all the ancestors as solved. If the start node s is solved, exit with success where To is the solution tree. Remove from open all nodes with a solved ancestor.
5. If n is not a solvable node (operators cannot be applied), label n as unsolvable. If the start node is labeled as unsolvable, exit with failure. If any of n's ancestors become unsolvable because n is, label them unsolvable as well. Remove from open all nodes with unsolvable ancestors

6. otherwise, expand node n generating all its successors. For each such successor node that contains more than one subproblem, generate their successors to give individual subproblems. Attach to each newly generated node a back pointer to its predecessor. Compute the cost estimate h^* for each newly generated node and place all such nodes that do not yet have descendants on open. Next, recompute the values h^* at n and each ancestor of n.
7. Return to step 2

6/10/22

Page No.
Date

5] Means - Ends Analysis

→ Means end analysis process centers around the detection of differences beth current state and goal state.

→ Once such a difference is isolated, an operator that can reduce the difference must be found.

→ If the operator cannot be applied to the current state, one has to set up a subproblem of getting to a state in which it can be applied.

→ The kind of backward chaining in which operators are selected and the subgoals are setup to establish the precondition of the operators is called the operator subgoaling

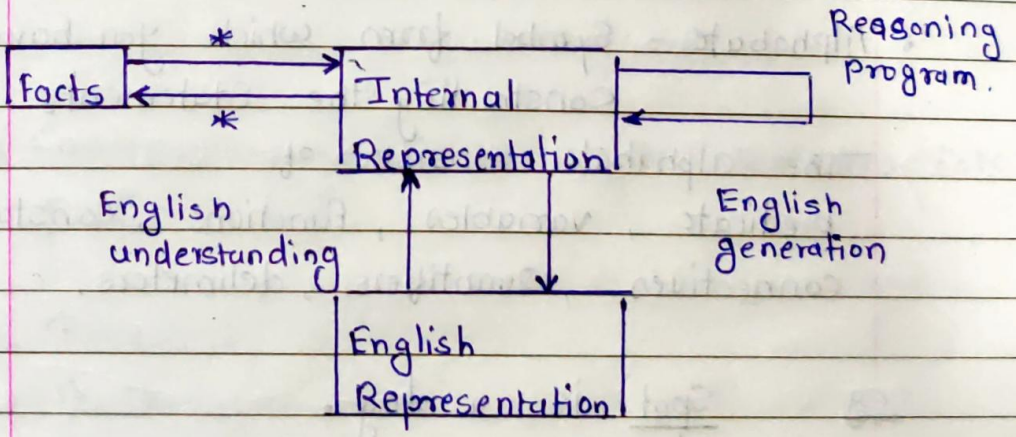
7/10/22

Knowledge Representation

★ Representations and Mappings

- Facts
  Truths in some relevant world.
- Representation of facts in some chosen formalism

⇒ 1) Knowledge level.
2) Symbol level
is a structuring these entity levels.



Mapping betn Facts and Representation

11/8/22

Page No.
Date

★ Propositional logic
• Predicate logic

FOPL (First order predicate logic)

1) Alphabate (symbol)
2) A formal language
3) A set of basic elements (axioms)
4) A set of inference Rule

• Alphabate - Symbol from which you have constructing the statements
The alphabate consists of predicate, variables, functions, constant, connectives, Quantifiers, delimeters,

eg Spot is a dog.
↓
Constant

• Predicate - Predicates are used to indicate the relation betn the domain elements.
They indicate that one element is related to another element in some specified way

Limitation - In proposition logic we do not use the quantifiers

Predicates along with the terms that identify the related elements are used to form atomic formulas which are the basic building blocks of predicate logic.

All man are mortal

$\forall x : [man(x) \rightarrow mortal(x)]$

man(x) ↓ predicate.

- Connective - $\wedge, \vee, \sim, \rightarrow$.
- Quatifiers - $\forall$ → Universal quatifier
  $\exists$ → existential quantifier

* Convert the following english sentences into predicate logic

1. Rover is black dog

=> 1) Rover is a dog
2) Rover is black

→ $dog(Rover) \wedge black(Rover)$

dog(Rover) ↓ atomic formula

2. John is a computer science student but not a pilot or football player

=) 1. John is a compter science student
2. John is not a pilot
3. John is not a football player

~~e.g student~~

wff → computer science (John) ∧ ~pilot (John) ∧
well formed formula. ~football player (John)

3. Leeann would be happy to find $1 bill or $5 bill.

⇒ 1. find one (leeann) ∨ find five (leeann) → happy (leeann)
2. happy (leeann, $1) ∨ happy (leeann, $5)

4. All elphants are gray

⇒ ∀x : elephant (x) → gray (x)

5. All plants require water.

⇒ ∀x : plant (x) → ~~into~~ require water (x)

6. You will gain ~~wait~~ weight unless you exercise.

⇒ 1. If you ~~will~~ do not exercise
2. then you will ~~get~~ gain ~~wait~~ weight

→ ∀x : ~exercise (x) → gain ~~wait~~ weight (x)

18/10/2

1. Marcus was a man

⇒ man (Marcus)

2. Marcus was a Pompein

⇒ pompeian (marcus)

3. All pompeian were Romans

⇒ ∀x : pompeian(x) → Romans (x)

4. Ceasar was a ruler

⇒ ruler (ceaser)

5. All Romans were either loyal to ceaser or hated him

⇒ ∀x : Roman (x) → [(loyalto (x, ceaser) ∨ hate (x, ceaser)) ∧ ~(loyalto(x, ceaser) ∧ hate (x, ceaser))]

6. Everyone is loyal to Someones

⇒ ∀x : ∃y : loyalto(x,y)

7. People only try to assassinate rulers they are not loyal to.

⇒ ∀x : ∀y : person (x) ∧ ruler (y) ∧ tryassasinate (x,y) → ~ loyalto (x,y)

8. Marcus tried to assassinate ceaser
⇒ tryassassinate (Marcus, Ceaser)

★ Resolution Algorithm: Convert to clause form

Write an algorithm for converting any wff into clause form

1. Eliminate → using the fact that $a \rightarrow b$ is equivalent to $\sim a \vee b$. ~~Perform~~

2. Reduce the scope of each ~ to a single term, using the fact that $\sim(\sim p) = p$, de Morgan's laws and the standard correspondance betn quantifiers

3 Standardize variable so that each quantifier binds unique variable.

4. Move all quantifiers to the left of the formula without changing their relative order. The formula is known as prenex normal form.

$\forall x : P(x) \vee \forall y : Q(y)$

$\forall x : \forall y : P(x) \vee Q(y)$

↓ Prefix ↓ Matrix

5. Eliminate existensial quantifiers. ~~A formula~~

$\exists(y) : \text{president}(y)$

$\Rightarrow \forall x : \exists y : \text{father of}(y, x)$

$\Rightarrow \forall x : \text{father-of}(S_2(x), x))$ ÷ Remove $\exists$

6. Drop the prefix.

7. Convert the matrix into conjunction of disjuncts.

$(a \wedge b) \vee c \equiv (a \vee c) \wedge (b \vee c)$

8 Create a seperate clause corresponding to each conjuct.

9. Standardize apart the variables in a set of clauses generated in step 8

★ The Basis of Resolution

⇒ Resolution produces proofs by refutation. In other words to prove a statement resolution attempts to show that the negation of statement produces a contradiction with the known statement

- The resolution procedure is a simple iterative process at each step called the parent clauses are compared or resolved producing a new clause that has been inferred from them.

winter ∨ summer
~winter ∨ cold
⇒ summer ∨ cold

★ Resolutional in propositional Logic

Algorithm

1. Convert all the propositions of F to clause form.

2. Negate P and convert the result to clause form. Add it to the set of clause obtained in step 1.

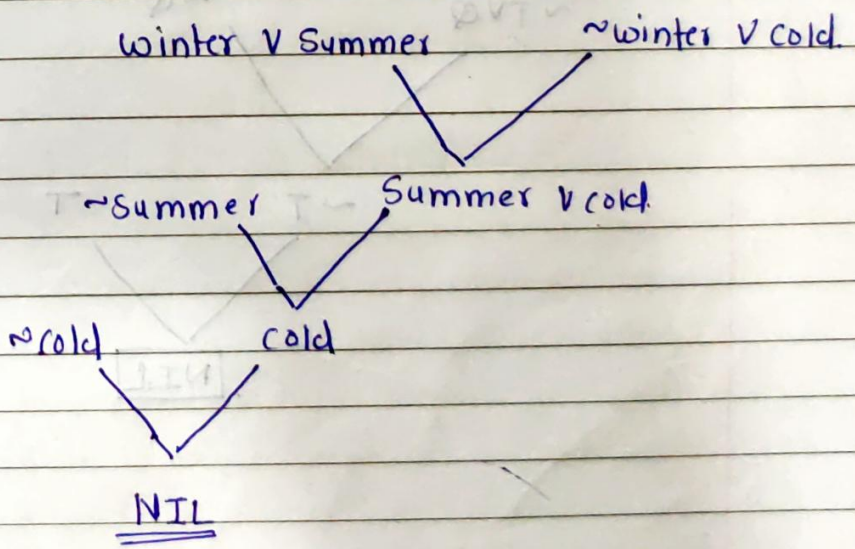
3. Repeat until either a contradiction is found or no progress can be made:

a) Select two clauses. Call these the parent clauses.

b) Resolve them together. The resulting clause called, resolvent, will be the disjuction of all the literals of both the parent clauses with the following exception:
If there are any pairs of literals L and ~L such that one of the parent clauses contains L and the other contains ~L, then select one such pair and eliminate both L and ~L from resolvent.

c) If the resolvent is the empty clause, then a contradiction has been found. If it is not, then add it to the set of clauses available to the procedure.

eg



$(P \wedge Q) \rightarrow R$
$\sim(P \wedge Q) \vee R$
$\sim P \vee \sim Q \vee R$

$S \vee T \rightarrow Q$
$\sim(S \vee T) \vee Q$
$\sim S \wedge \sim T \vee Q$
$(\sim S \vee Q) \wedge (\sim T \vee Q)$

$\sim S \vee Q$
$\sim T \vee Q$

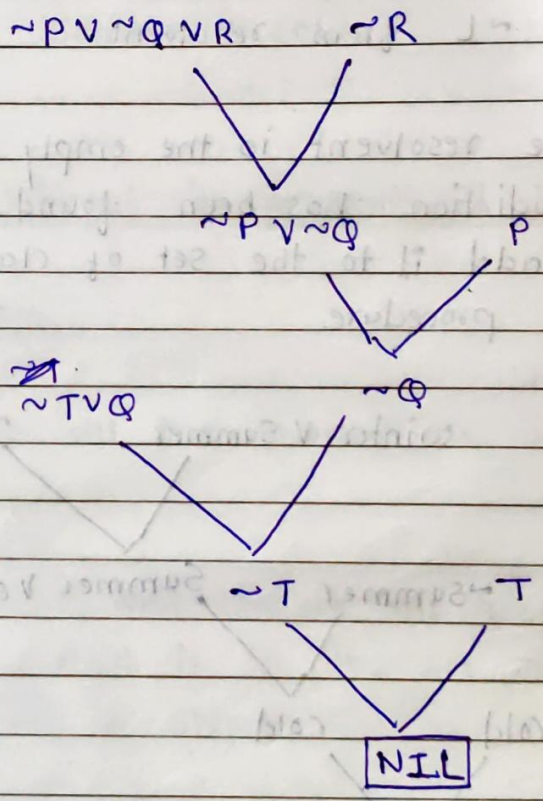
given axioms

converted to clause form

1) P — P
2) $(P \wedge Q) \rightarrow R$ — $\sim P \vee \sim Q \vee R$
3) $\sim(S \vee T) \rightarrow Q$ — $\sim S \vee Q$, $\sim T \vee Q$
4) T — T

Prove - R

We add $\sim R$ because you prove R & therefore we assume $\sim R$



tuom 18/10/22

We have to prove
Marcus was loyal to ceaser
→ loyalto (Marcus, ceaser

⇒ Axioms in clause form.

1. man (marcus)
2. pompeian (marcus)
3. ∀x : pompeian (x) → Romans (x)
⇒ ~pompeian (x) ∨ Romans (x)

4. ruler (ceaser)
5. ∀x : Roman(x) → [(loyalto (x, ceaser) ∨ hate (x, ceaser)) ∧ ~(loyalto (x, ceaser) ∧ hate (x, ceaser))]
OR ∀x : Roman (x) → loyalto(x, ceaser) ∨ hate(x, ceaser)

⇒ ~Roman(x) ∨ loyalto (x, ceaser) ∨ hate (x, ceaser)

6. ∀x : ∃y : loyalto (x, y)
⇒ ∀x : loyalto (x, f3(x))
loyalto (x3, f3(x3))

7. ∀x : ∀y : ~~person~~ man (x) ∧ ruler (y) ∧ tryassasinate (x, y) → ~loyalto (x, y)
⇒ ∀x : ∀y : ~[~~person~~ man (x) ∧ ruler (y) ∧ tryassasinate (x, y)] ∨ ~loyalto (x, y)

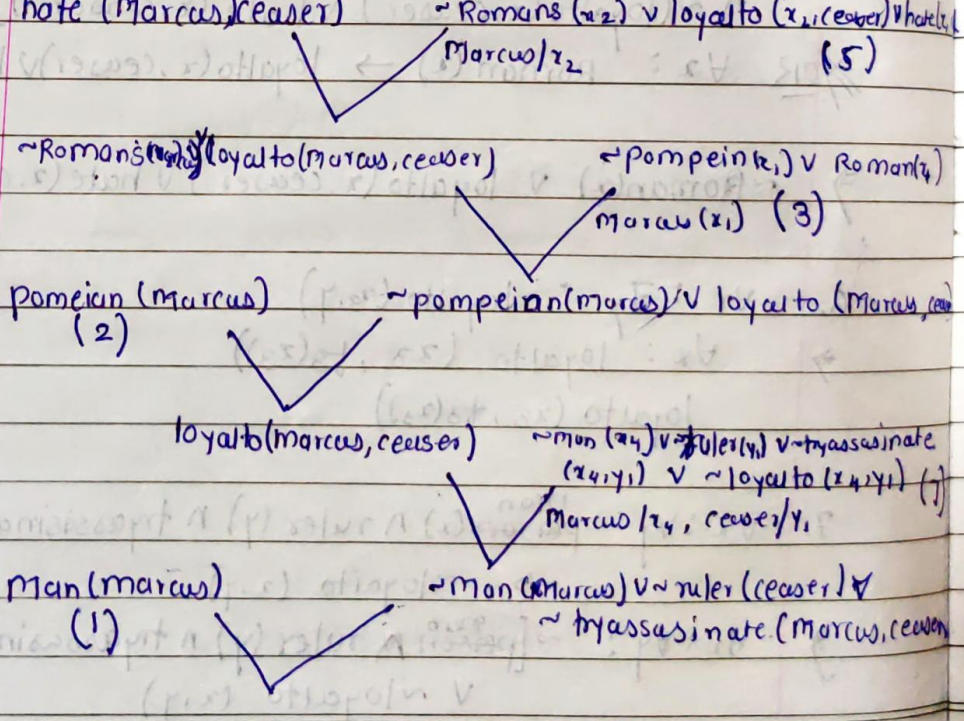
$\forall x : \forall y : \sim$ ~~person~~ man $(x_4) \vee \sim$ ~~loyalto~~ ruler $(y) \vee \sim$ tryassasinate $(x, y) \vee \sim$ loyalto(x, y)

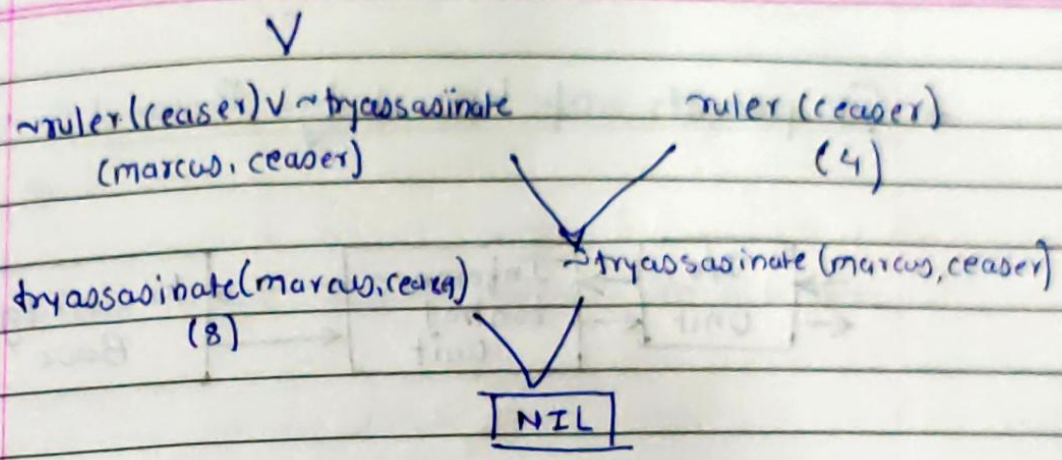
$\sim$ ~~person~~ man $(x_4) \vee \sim$ ~~loyalto~~ ruler $(y_1) \vee \sim$ tryassasinate $(x_4, y_1) \vee \sim$ loyalto (x_4, y_1)

8. tryassasinate (Marcus, Ceaser)

Resolution

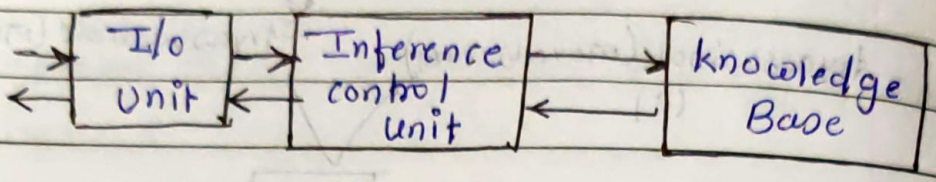
To prove :- hate (Marcus, Ceaser)
give negation of that i.e. $\Rightarrow \sim$hate(Marcus, Ceaser)





Hence proved, hate (marcus, ceaser).
∴ marcus hate ceaser.

★ Components of knowledge Based.



Q Why knowledge based is stored in knowledge base seperate from control and infering components ?

⇒ This makes it possible to add new knowledge or refine existing knowledg with recompiling the control and inferencing program.

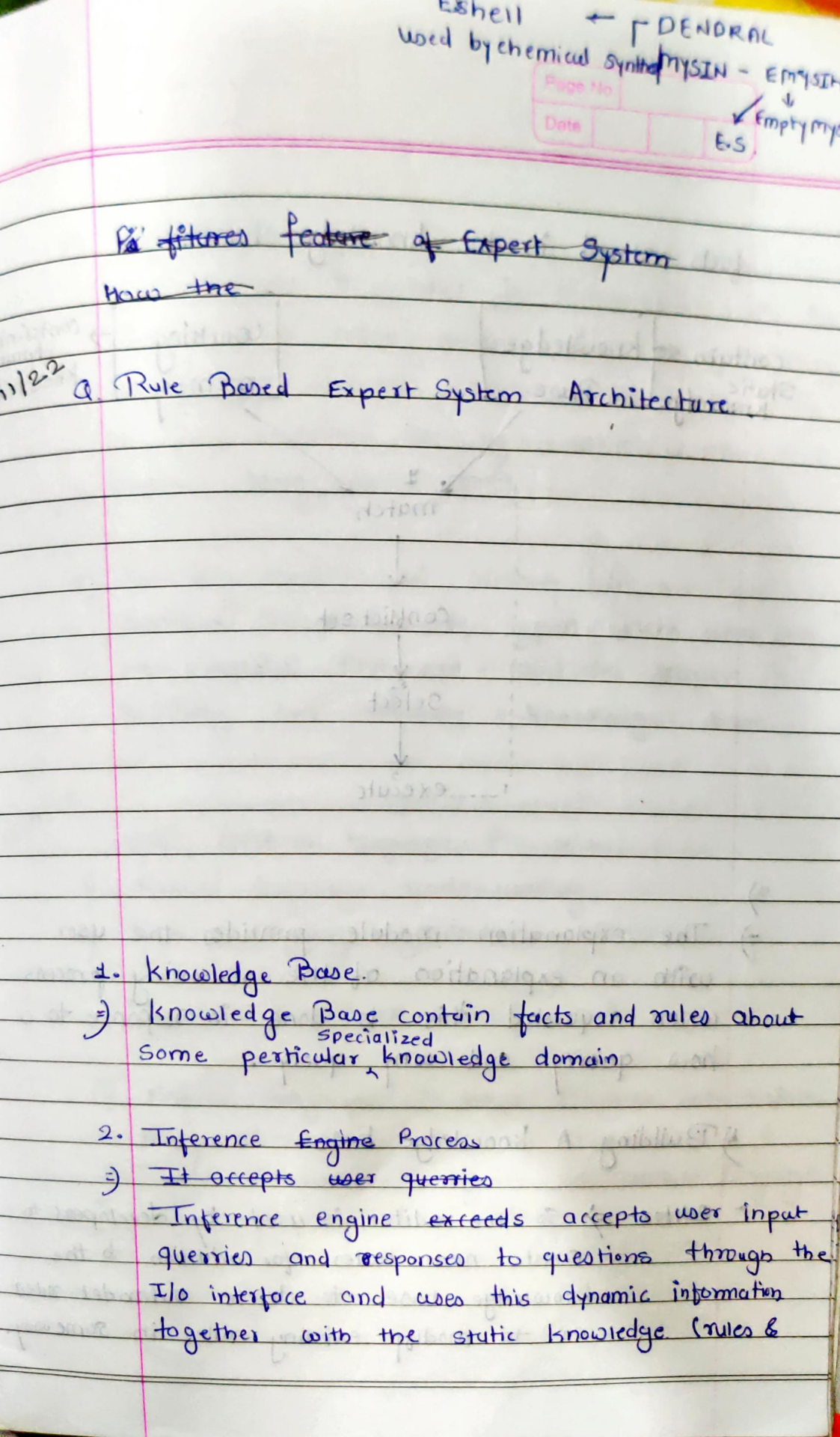
★ Expert System.

⇒ An expert system is a set of programs that manipulate encoded knowledge to solve problems in a specialized domain that requires human expertise

Eshell ← DENDRAL
used by chemical synthesis MYSIN - EMYSIN
↓
Empty MYS
E.S

~~Fix fitures feature of Expert System~~
~~How the~~

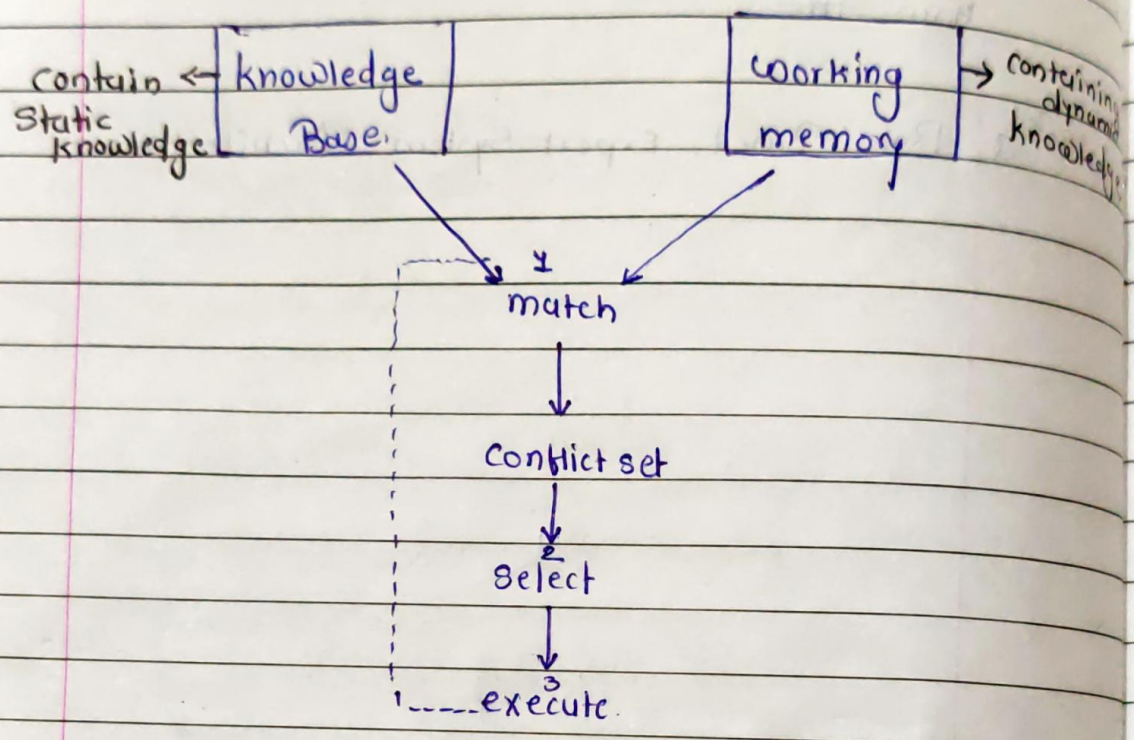
11/22 Q. Rule Based Expert System Architecture.



1. Knowledge Base.
=) Knowledge Base contain facts and rules about some particular specialized knowledge domain

2. Inference ~~Engine~~ Process
=) ~~It accepts user querries~~
Inference engine ~~exceeds~~ accepts user input querries and responses to questions through the I/O interface and uses this dynamic information together with the static knowledge (rules &

facts stored in the knowledge base.



3)

=) The explanation module provides the user with an explanation of the reasoning process when requested this is done in response to a how query and why query.

4) Building A knowledge base

- Editor => So the editor is used by developers to create new rules for addition to the knowledge base to delete outmodel rules or to modify existing rules in some way.

5) I/O Interface

=) It permits the user to communicate with the system in a more natural way by permitting by use of simple selection menues or the use of restricted language which is close to a natural language

6) Learning module and History file are not common components of expert system when they are provided they are used to assist in building and refining a knowledge base

15/11/22

NLP. (Natural language Processing)

1) Natural language understanding
2) Natural language generation

Reason - why NLP is difficult ?

1) Natural language is large because they contain many statement.
2) There is much ambiguity in natural language eg. can, fly, bear, orange

Levels of knowledge used in language understanding.

1] phonological
2] morphological
3] Syntantic
4] Semantic
5] Pragmatic
6] World knowledge

18/11/22

★ General Approaches to Natural language Understanding

1) Use of keyword and pattern matching
2) Combine Syntantic and Sematic directed analysis
3) Comparing and matching the input to real-world situations.

1) :- 1st approach is a simplest based on the use of sentence templetes which contain keywords or phrases such as I am ____ OR And I don't like ____.

- That are match against the input sentences. Each input Templet has associated with it one

or more output templets, one of which is used to produced or response to the given input

- Advantage -

1) Ungrammatical but meaningful sentences are accepted

- Disadvantage -

No actual knowledge structure are created. So, the program does not really understand.

3) ⇒ • 3rd approach is based on the use of the structures such as frames or scripts

- The approach realised more on a maping of the input to prescribed primitives which are used to build large knowledge structure..
- It depends on the use of the constraints imposed by a context and world knowledge to develop and understanding of the language input.
- Prestored discription and details for commonly occuring situation or event are recalled for use in understanding a new situation.
- The stored events are then used to filled in missing details about the ~~cor~~ current scenario.

- Advantage
  Much of the computation required for syntatical is bipased.

- Disadvantage
  Substential amount of specific as well as general world knowledge must be prestored.

2) ⇒ (common use)

- In the 2nd approach knowledge structures are constructed during a syntactical and Semantic analysis of the input sentences.
- Parsers are used to analyze individual sentences and to build structures that can b used directly or transformed into the require knowledge format

- Advantage
  Power and versatility its provide.

- Disadvantage
  Large amount of computation required and the need for still furthur processing to understand the contextual meaning of more than one sentence.

★ characteristics of Artificial Neural Network.

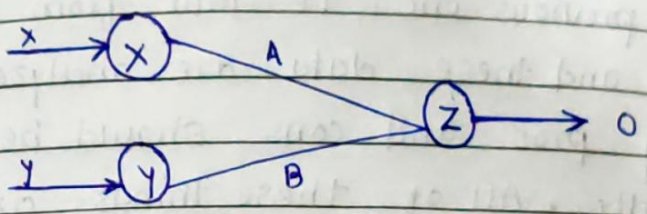
i) It is neurally implemented mathematical model
ii) It contains huge number of interconnected processing elements called neurons to do all operations.
iii) Information stored in the neurons are basically the weighted linkage of neurons
iv) The input signals arrive at the processing elements through connections and connecting weights.
v) It has the ability to learn, recall and generalize from the given data by suitable assignment and adjustment of weights.
vi) The collective behaviour of the neurons describe its computational power and no single neuron carries specific information

★ Simple Neuron Works

→ Working of simple Neuron works.

- Let there are two neurons X and Y which is transmitting signal to another neuron Z. then X and Y are input neurons for transmitting signals and Z is output neuron for receiving signal.

- The input neurons are connected to the output neuron, over a interconnection links (A and B) as shown in fig given below.



Architecture of a simple artificial neuron network

- For above Neuron Architecture, the net input has to be calculated in the way.

$$I = xA + yB$$

where x and y are the activations of the input neurons X and Y. The output z of the output neuron Z can be obtained by applying activations over the net input.

$$O = f(I)$$

output = function (net input calculated)

- The function to be applied over the net input is called activation function. There are various activation function possible for this.

Page No.
Date

★ Application of Neural Networks

1) Every new technology need assistance from the previous one. i.e data from previous ones and these data are analyzed so that every pros and cons should be studied correctly. All of these things are possible only through the help of neural network.
2) Neural network is suitable for the research on Animal behaviour, predator / prey relationships and population cycles.
3) It would be easier to do proper valuation of property, buildings, automobiles, machinery etc. with the help of neural network.
4) Neural Network can be used in betting on horse races, sporting events, and most importantly in stock market.
5) It can be used to predicate the correct judgement for any crime by using a large data of crime details as input and the resulting sentences as output.
6) By analyzing data and determining which of the data has any fault (files diverging from peers) called as Data mining cleaning and validation can be acheived through neural network.

7) Neural Networks can be used to predicate targets with the help of echo patterns we get from sonar, radar, seismic and magnetic instrument.
8) It can be used efficiently in employee hiring so that any company can hire the right employee depending upon the skills the employee has and what should be its productivity in future.
9) It has a large application in Medical research
10) It can be used to be Fraud Detection regarding credit cards, insurance or taxes by analyzing the past records.

★ Genetic Algorithm

- Genetic algorithm are adaptive heuristic search Algorithms that belong to the larger part of evolutionary algorithms. Genetic Algorithm are based on the ideas of neural selection and genetics.
- These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

- Genetic algorithm simulate the process of natural selection which means those species who can adapt to changes in their environment are able to survive and produce and go to next generation.
- In simple words, they simulate "survival of the fittest" among individual of consecutive generation for solving a problem. Each generation consist of a population of individuals represents a point in search space and possible solution.
- Each individual is represented as a string of character / integer / float / bits. This string is analogous to the chromosome.

Foundation of Genetic Algorithm

Genetic algo. are based on an analogy with genetic structure and behaviour of chromosomes of the population.
Following is the foundation of GAs based on this analogy.

i) Individual in population compete for resources and mate.

ii) Those individual who are successful (fittest) then mate to create more offspring than others.

iii) Genes from "fittest" parent create offspring which is better than either parent.

iv) Thus, each successive generation is more suited for their environment.

a) Search Space

→ • The population of individuals are maintained within search space. Each individual represents a solution in search space for given problem.

• Each individual is coded as finite length vector (analogous to chromosomes) of components. This variable components are analogous to Genes.

• Thus a chromosome (individual) is composed of several genes (variable components).

b) Fitness Score :

→ • A fitness score is given to each individual which shows the ability of an individual to "compete". The individual having optimal fitness score are sought.

- The GAs maintained the population of individuals (chromosomes / solutions) along with their fitness score. The individuals having better fitness scores are given more chance to reproduce than others. The individuals with better fitness scores are selected who mate and produce better offspring by combining chromosomes of parents
- The population size is static so the room has to be created for new arrivals so, some individuals die and get replaced by new arrivals eventually creating new generation when all the mating opportunity of the old population is exhausted. It is hoped that over successive generations better solution will arrive while least fit die.
- Each new generation has no average more "better genes" than the individual of previous generations
- Thus, each new generation have better "Partial solutions" than previous generation once the offspring produced having no significant difference from offspring produced by previous population, the population is converge The algo is said to be converged to a set of soln for the prob.

c) Operators of Genetic Algo.

→ Once the initial generation is created, the algo. evolves the generation using following operators:

1) Selection Operator:

⇒ The idea is to give preference to the individual with good fitness scores and allow them to pass their genes to successive generations

2) Crossover Operator:

⇒ This represent mating betn individuals. Two individuals are selected using selection operator and crossover sites are chossen randomly.