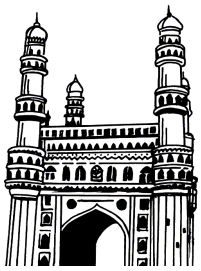
Rahul's ✔
Topper's Voice



M.C.A.

II Year III Sem

(Osmania University)

Latest 2024 Edition

ARTIFICIAL INTELLIGENCE

- ☞ **Study Manual**
- ☞ **Important Questions**
- ☞ **Solved Model Papers**

- by -

WELL EXPERIENCED LECTURER

Price ` 169-00

Rahul Publications™

Hyderabad. Cell : 9391018098, 9505799122

M.C.A.

II Year III Sem

(Osmania University)

ARTIFICIAL INTELLIGENCE

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price `. 169 -00

Sole Distributors : **Cell : 9391018098, 9505799122**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.

Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

ARTIFICIAL INTELLIGENCE

CONTENTS

STUDY MANUAL

SOLVED MODEL PAPERS

SYLLABUS

UNIT - I

Introduction, History, Intelligent Systems, Foundations of AI, Sub-areas of AI, Applications, Problem Solving. State-Space Search and Control Strategies: Introduction, General Problem Solving,Characteristics of Problem, Exhaustive Searches, Heuristic Search Techniques, Iterative-Deepening, A*,Constraint Satisfaction. Game Playing, Bounded Look-ahead Strategy and use of Evaluation Functions, Alpha-Beta Pruning

UNIT - II

Logic Concepts and Logic Programming: Introduction, Propositional Calculus, Propositional Logic, Natural Deduction System, Axiomatic System, Semantic Tableau System in Propositional Logic, Resolution Refutation in Propositional Logic, Predicate Logic, Logic Programming.

Knowledge Representation: Introduction, Approaches to Knowledge Representation, Knowledge Representation using Semantic Network, Knowledge Representation using Frames

UNIT - III

Expert System and Applications: Introduction, Phases in Building Expert Systems, Expert System Architecture, Expert Systems vs Traditional Systems, Truth Maintenance Systems, Application of Expert Systems, List of Shells and Tools. Uncertainty Measure-Probability Theory: Introduction, Probability Theory, Bayesian Belief Networks, Certainty Factor Theory, Dempster-Shafer Theory.

UNIT - IV

Machine-Learning Paradigms: Introduction, Machine Learning Systems, Supervised and Unsupervised Learning, Inductive Learning, Learning Decision Trees (Suggested Reading 2), Deductive Learning, Clustering, Support Vector Machines.

Artificial Neural Networks: Introduction, Artificial Neural Networks, Single-Layer Feed-Forward Networks, Multi-Layer Feed-Forward Networks, Radial-Basis Function Networks, Design Issues of Artificial Neural Networks, Recurrent Networks.

UNIT - V

Advanced Knowledge Representation Techniques: Case Grammars, Semantic Web.

Natural Language Processing: Introduction, Sentence Analysis Phases, Grammars and Parsers, Types of Parsers, Semantic Analysis, Universal Networking Knowledge.

Contents

UNIT - I

UNIT - II

Important Questions

UNIT - I

1. What is Artificial Intelligence? Write about various approaches used to define AI.

Ans :

Refer Unit-I, Page No. 1, Q.No. 1

---

2. What are intelligent systems? Write about its types.

Ans :

Refer Unit-I, Page No. 3, Q.No. 3

---

3. Explain about the sub areas of AI.

Ans :

Refer Unit-I, Page No. 5, Q.No. 5

---

4. What is problem in AI? Write the steps for problem solving in AI.

Ans :

Refer Unit-I, Page No. 7, Q.No. 7

---

5. Write about problem characteristics.

Ans :

Refer Unit-I, Page No. 11, Q.No. 12

---

6. What is Exhaustive search? Explain about it.

Ans :

Refer Unit-I, Page No. 13, Q.No. 14

---

7. Solve Kanpsack problem using Exhaustic search method.

Ans :

Refer Unit-I, Page No. 15, Q.No. 15

---

8. Explain A* Algorithm.

Ans :

Refer Unit-I, Page No. 20, Q.No. 22

---

9. Explain, How to solve Constraint Satisfaction Problems using Search.

Ans :

Refer Unit-I, Page No. 22, Q.No. 24

---

10. What is the importance of game playing in AI. Explain Min- Max Algorithm.

Ans :

Refer Unit-I, Page No. 23, Q.No. 25

UNIT - II

1. What is propositional calculus? Explain the fundamental components of propositional calculus.

Ans :

Refer Unit-II, Page No. 31, Q.No. 2

2. What is propositional logic? Explain the basic facts about propositional logic

Ans :

Refer Unit-II, Page No. 33, Q.No. 3

3. Explain about propositional logical connectives and logical equivalances.

Ans :

Refer Unit-II, Page No. 34, Q.No. 4

4. Define inference rules? Explain the types of inference rules.

Ans :

Refer Unit-II, Page No. 37, Q.No. 5

5. Write about natural deduction system.

Ans :

Refer Unit-II, Page No. 41, Q.No. 6

6. What is axiomatic system. Explain its rules.

Ans :

Refer Unit-II, Page No. 44, Q.No. 7

7. Explain about semantic tableau system in propositional logic.

Ans :

Refer Unit-II, Page No. 45, Q.No. 8

8. Explain about resolution refutation in propositional logic.

Ans :

Refer Unit-II, Page No. 47, Q.No. 9

9. Explain various techniques of knowledge representation.

Ans :

Refer Unit-II, Page No. 55, Q.No. 16

10. How to represent knowledge with semantic network? Explain

Ans :

Refer Unit-II, Page No. 58, Q.No. 18

UNIT - III

1. Explain various steps to develop an Expert system.

Ans :

Refer Unit-III, Page No. 63, Q.No. 2

2. Distinguish between traditional system and expert system.

Ans :

Refer Unit-III, Page No. 67, Q.No. 4

3. What is a Truth Maintenance System (TMS)? Expalin.

Ans :

Refer Unit-III, Page No. 67, Q.No. 5

4. What are the applications of Expert systems? Write.

Ans :

Refer Unit-III, Page No. 69, Q.No. 7

5. Give the list of various Expert system shells and tools.

Ans :

Refer Unit-III, Page No. 71, Q.No. 9

6. What is uncertainity? State its causes.

Ans :

Refer Unit-III, Page No. 72, Q.No. 10

7. Solve the following examples using Bayes' theorem.

Ans :

Refer Unit-III, Page No. 75, Q.No. 13

8. Explain about Bayesian Belief Network in artificial intelligence.

Ans :

Refer Unit-III, Page No. 76, Q.No. 14

9. Explin the uses of certainity factor in AI?

Ans :

Refer Unit-III, Page No. 81, Q.No. 16

10. Explain briefly about Dempster Shafer Theory

Ans :

Refer Unit-III, Page No. 82, Q.No. 17

UNIT - IV

1. Explain the main key components of Machine Learning Systems?

Ans :

Refer Unit-IV, Page No. 85, Q.No. 2

2. Explain the categories of supervised machine learning.

Ans :

Refer Unit-IV, Page No. 88, Q.No. 4

3. Explain the Advantages and Disadvantages and applications of Unsupervised Learning.

Ans :

Refer Unit-IV, Page No. 90, Q.No. 6

4. Explain briefly about learning decision trees in ML.

Ans :

Refer Unit-IV, Page No. 93, Q.No. 8

5. What is deductive learning ? Write about it.

Ans :

Refer Unit-IV, Page No. 97, Q.No. 10

6. What is clustering? Explain various types of clustering methods

Ans :

Refer Unit-IV, Page No. 98, Q.No. 11

7. Explain briefly about Artificial Neural Network?

Ans :

Refer Unit-IV, Page No. 14, Q.No. 103

8. Explain the Design Issues of Artificial Neural Networks.

Ans :

Refer Unit-IV, Page No. 117, Q.No. 24

9. Explain briefly about Recurrent Neural Network (RNN)?

Ans :

Refer Unit-IV, Page No. 118, Q.No. 25

10. Explain the application, advantages and disadvantages of Recurrent Neural Networks?

Ans :

Refer Unit-IV, Page No. 121, Q.No. 27

UNIT - V

1. Explain various cases in case grammars.

Ans :

Refer Unit-V, Page No. 125, Q.No. 1

2. Explain importance of Semantic web in AI?

Ans :

Refer Unit-V, Page No. 127, Q.No. 2

3. What is Natural Language Processing? State its advantages and disadvantages.

Ans :

Refer Unit-V, Page No. 129, Q.No. 3

4. Expalin the components and applications of NLP.

Ans :

Refer Unit-V, Page No. 129, Q.No. 4

5. What is Grammar in AI? Explain

Ans :

Refer Unit-V, Page No. 133, Q.No. 7

6. Explain the Chomsky classification of grammar.

Ans :

Refer Unit-V, Page No. 134, Q.No. 8

7. What is Parsing? Explain about it

Ans :

Refer Unit-V, Page No. 136, Q.No. 9

8. Explain various types of parsing in AI.

Ans :

Refer Unit-V, Page No. 137, Q.No. 10

9. Define meaning representation in AI. Explain the approaches in meaning representation.

Ans :

Refer Unit-V, Page No. 140, Q.No. 12

10. What is Universal Networking Language? Explain.

Ans :

Refer Unit-V, Page No. 141, Q.No. 14

UNIT I

Introduction, History, Intelligent Systems, Foundations of AI, Sub-areas of AI, Applications, Problem Solving. State-Space Search and Control Strategies: Introduction, General Problem Solving,Characteristics of Problem, Exhaustive Searches, Heuristic Search Techniques, Iterative-Deepening, A*,Constraint Satisfaction. Game Playing, Bounded Look-ahead Strategy and use of Evaluation Functions, Alpha-Beta Pruning

1.1 INTRODUCTION TO AI

Q1. What is Artificial Intelligence? Write about various approaches used to define AI.

Ans : **(Imp.)**

Artificial Intelligence is the branch of computer science concerned with making computers behave like humans.

The definitions of AI according to some text books are categorized into four approaches and are summarized in the table below :

Systems that think like humans "The exciting new effort to make computers think … machines with minds, in the full and literal sense."	**Systems that think rationally** "The study of mental faculties through the use of computer models."
Systems that act like humans The art of creating machines that perform functions that require intelligence when performed by people."	**Systems that act rationally** **"**Computational intelligence is the study of the design of intelligent agents."

Approaches

The four approaches in more detail are as follows :

1. Acting humanly : The Turing Test approach

- Test proposed by Alan Turing in 1950
- The computer is asked questions by a human interrogator.

The computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or not. Programming a computer to pass ,the computer need to possess the following capabilities :

Natural language processing to enable it to communicate successfully in English.

Knowledge representation to store what it knows or hears

Automated reasoning to use the stored information to answer questions and to draw new conclusions.

Machine learning to adapt to new circumstances and to detect and extrapolate patterns

To pass the complete Turing Test,the computer will need

- Computer vision to perceive the objects, and
- Robotics to manipulate objects and move about

2. Thinking humanly : The cognitive modelling approach

We need to get inside actual working of the human mind :

(a) through introspection – trying to capture our own thoughts as they go by;

(b) through psychological experiments

Allen Newell and Herbert Simon, who developed GPS, the "General Problem Solver" tried to trace the reasoning steps to traces of human subjects solving the same problems.

The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind

3. Thinking rationally : The "laws of thought approach"

The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking" ,that is irrefuatable reasoning processes. His syllogism provided patterns for argument structures that always yielded correct conclusions when given correct premises for example, "Socrates is a man; all men are mortal; therefore Socrates is mortal.".

These laws of thought were supposed to govern the operation of the mind; their study initiated a field called logic.

4. Acting rationally : The rational agent approach

An **agent** is something that acts. Computer agents are not mere programs ,but they are expected to have the following attributes also : (a) operating under autonomous control, (b) perceiving their environment, (c) persisting over a prolonged time period, (e) adapting to change.

A rational agent is one that acts so as to achieve the best outcome.

1.1.1 History

Q2. Explain about the evolution of AI.

Ans :

Following are some milestones in the history of AI which defines the journey from the AI generation to till date development.

Maturation of Artificial Intelligence (1943-1952)

- **Year 1943:** The first work which is now recognized as AI was done by Warren McCulloch and Walter pits in 1943. They proposed a model of artificial neurons.
- **Year 1949:** Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons. His rule is now called Hebbian learning.
- **Year 1950:** The Alan Turing who was an English mathematician and pioneered Machine learning in 1950. Alan Turing publishes "Computing Machinery and Intelligence" in which he proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a Turing test.

The birth of Artificial Intelligence (1952-1956)

- **Year 1955:** An Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as "Logic Theorist". This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
- **Year 1956:** The word "Artificial Intelligence" first adopted by American Computer scientist John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

At that time high-level computer languages such as FORTRAN, LISP, or COBOL were invented. And the enthusiasm for AI was very high at that time.

The golden years-Early enthusiasm (1956-1974)

- **Year 1966:** The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizenbaum created the first chatbot in 1966, which was named as ELIZA.
- **Year 1972:** The first intelligent humanoid robot was built in Japan which was named as WABOT-1.

The first AI winter (1974-1980)

- The duration between years 1974 to 1980 was the first AI winter duration. AI winter refers to the time period where computer scientist dealt with a severe shortage of funding from government for AI researches.

- During AI winters, an interest of publicity on artificial intelligence was decreased.

A boom of AI (1980-1987)

- **Year 1980:** After AI winter duration, AI came back with "Expert System". Expert systems were programmed that emulate the decision-making ability of a human expert.
- In the Year 1980, the first national conference of the American Association of Artificial Intelligence was held at Stanford University.

The second AI winter (1987-1993)

- The duration between the years 1987 to 1993 was the second AI Winter duration.
- Again Investors and government stopped in funding for AI research as due to high cost but not efficient result. The expert system such as XCON was very cost effective.

The emergence of intelligent agents (1993-2011)

- **Year 1997:** In the year 1997, IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
- **Year 2002:** for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
- **Year 2006:** AI came in the Business world till the year 2006. Companies like Facebook, Twitter, and Netflix also started using AI.

Deep learning, big data and artificial general intelligence (2011-present)

- **Year 2011:** In the year 2011, IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
- **Year 2012:** Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
- **Year 2014:** In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test."
- **Year 2018:** The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.
- Google has demonstrated an AI program "Duplex" which was a virtual assistant and which had taken hairdresser appointment on call, and lady on other side didn't notice that she was talking with the machine.

Now AI has developed to a remarkable level. The concept of Deep learning, big data, and data science are now trending like a boom. Nowadays companies like Google, Facebook, IBM, and Amazon are working with AI and creating amazing devices. The future of Artificial Intelligence is inspiring and will come with high intelligence.

1.1.2 Intelligent Systems

Q3. What are intelligent systems? Write about its types.

Ans : **(Imp.)**

Meaning

Intelligent systems in AI refer to systems that exhibit the ability to perform tasks typically associated with human intelligence. These systems use various techniques and approaches to simulate or replicate human-like cognitive functions such as learning, reasoning, problem-solving, perception, understanding natural language, and adapting to new situations. Intelligent systems aim to process information, make decisions, and generate responses in a manner that appears intelligent.

Types

Some common types of intelligent systems in AI include:

1. **Machine Learning Systems:** These systems use statistical techniques to enable computers to learn from data and improve their performance over time. They can be used for tasks like image and speech recognition, recommendation systems, and fraud detection.
2. **Natural Language Processing (NLP) Systems:** NLP systems allow computers to understand, interpret, and generate human language. They power applications such as language translation, sentiment analysis, chatbots, and virtual assistants.
3. **Expert Systems:** These are computer programs designed to emulate the decision-making abilities of a human expert in a specific domain. They use knowledge representation and reasoning techniques to provide advice or solutions to complex problems.

4. **Robotics and Autonomous Systems:** These systems combine AI with robotics to create machines that can perform tasks autonomously in real-world environments. Examples include self-driving cars, drones, and industrial robots.

5. **Cognitive Computing Systems:** These systems aim to simulate human thought processes and are designed to understand and respond to complex patterns and data. They often involve advanced techniques such as neural networks and deep learning.

6. **Reinforcement Learning Systems:** These systems learn through interaction with an environment by receiving feedback in the form of rewards or penalties. They are used in applications like game playing, control systems, and optimization.

7. **Perception Systems:** These systems enable computers to interpret and understand sensory data from the environment, such as visual information from cameras or audio input from microphones. Computer vision and speech recognition are examples of perception systems.

8. **Emotion Recognition Systems:** These systems attempt to detect and interpret human emotions based on facial expressions, voice tone, and other cues. They have applications in fields like marketing, customer service, and mental health.

Intelligent systems often incorporate a combination of techniques from various AI subfields, including machine learning, natural language processing, computer vision, and knowledge representation. The goal is to create systems that can perform complex tasks and adapt to new situations, ultimately contributing to more efficient and effective problem-solving and decision-making across various domains.

1.1.3 Foundations of AI

Q4. Write about the areas where AI can be used.

Ans :

1. Philosophy

- AI takes the following ideas from philosophy.
- Can formal rules be used to draw valid conclusion?

  Aristotle (312 – 322 B.C) was the first to formulate the precise set of laws governing the rational parts of the mind. He developed an informal system of syllogisms for proper reasoning, which in principle allowed one to generate conclusions mechanically given initial premises.
- How does the mind arises from a physical brain?

  It is one thing to say that mind operates, at least in part, according to logical rules and to build physical system that emulates some of those rules, but it's another thing to say that mind itself is such a physical system.
- Where does knowledge from?

  Given a physical mind that manipulate knowledge, the next problem is to establish the source of knowledge.
- How does knowledge lead to action?

  Philosophy tries to answer the connection between knowledge and action. This question is vital to AI because intelligence requires action as well as reasoning.

2. Mathematics

Philosophers staked out some of the fundamental ideals of AI, but the leap to formal science required a level of mathematical formalization in their fundamental areas:

- What are the formal rules to draw valid conclusions?
- What can be compute?
- How do we reason with uncertain information?

3. Economics

AI takes the following ideas from economics.

- How should we make decisions so as to maximize payoff (utility)?

  Decision theory, which combines probability theory. With utility theory, provides a formal and complete framework for decision made under uncertainty. This is suitable for large "economics" where each agent need pay no attention to the action of the other gent as individuals.
- How should we do this when others may not go along?

  For small economics, the situation is much more like a game. The action of one player can significantly affect the utility of another. Unlike decision theory game theory does not offer an

unambiguous prescription for selecting actions.

- How should we do this when the payoff may be far in the future?

For the most part economist did not address this question. This topic was pursed in the field of operation research (OR).

4. Neuroscience

Neuroscience is the study of the nervous system, particularly the brain. This study of how brain process information directly help in the development of AI.

5. Psychology

Psychology tries to answer the following question:

- How do humans and animals think and act? Modern science proves that computer models could be used to address the psychology of memory, language and logical thinking respectively. It is now common view among psychologist that "a cognitive theory should be like a computer program."

6. Computer engineering

- How can we build an efficient computer?

For AI to succeed, we need two things:

Intelligence and artifact.

The computer has been the artifact of the choice.

7. Control theory and cybernetics

- How can artifacts operate under their own control ?

Control theory and cybernetics deals with the self controlling machine. Modern control theory has its goal to design the system that maximize an objective function overtime. This roughly match the AI view: designing system that behave optimally.

8. Linguistics

Language is directly related to thought. Modern linguistics and AI, were born at about the same time and grew up together, intersecting in a hybrid field called computational linguistics or natural language processing.

1.1.4 Sub-Areas of AI

Q5. Explain about the sub areas of AI.

Ans : **(Imp.)**

Artificial Intelligence (AI) is a broad and interdisciplinary field that encompasses various sub-areas, each focusing on different aspects of replicating human intelligence or creating intelligent behaviour in machines. Here are some important sub-areas of AI:

1. **Machine Learning (ML):** ML is a subset of AI that involves the development of algorithms and techniques that allow computers to learn from data and improve their performance over time without being explicitly programmed. Sub-areas within machine learning include:
   - Supervised Learning
   - Unsupervised Learning
   - Semi-Supervised Learning
   - Reinforcement Learning
   - Deep Learning
2. **Natural Language Processing (NLP):** NLP focuses on enabling computers to understand, interpret, and generate human language. Sub-areas within NLP include:
   - Text Processing
   - Speech Recognition
   - Sentiment Analysis
   - Language Translation
   - Named Entity Recognition
3. **Computer Vision:** Computer vision involves teaching computers to interpret and understand visual information from the world, such as images and videos. Sub-areas within computer vision include:
   - Image Recognition
   - Object Detection
   - Image Generation
   - Facial Recognition
   - Scene Understanding
4. **Robotics:** Robotics integrates AI and engineering principles to create autonomous systems that can interact with the physical world. Sub-areas within robotics include:
   - Robot Control
   - Path Planning
   - Human-Robot Interaction
   - Manipulation and Grasping
5. **Expert Systems:** Expert systems emulate the decision-making abilities of human experts in

specific domains. They are designed to provide advice and solutions to complex problems. Sub-areas within expert systems include:

- Knowledge Representation
- Inference Engines
- Rule-Based Systems

6. **Knowledge Representation and Reasoning :** This area focuses on developing formal methods for representing knowledge and using logical reasoning to draw conclusions from that knowledge.
7. **Planning and Scheduling:** These sub-areas involve developing algorithms that enable computers to plan sequences of actions to achieve specific goals and allocate resources efficiently.
8. **Machine Perception:** This area deals with enabling machines to perceive and interpret sensory data from the environment, such as visual, auditory, or haptic information.
9. **Cognitive Computing:** Cognitive computing aims to simulate human thought processes, including learning and problem-solving, often incorporating advanced techniques like neural networks and deep learning.
10. **Emotion AI:** Emotion AI focuses on developing systems that can detect, understand, and respond to human emotions based on facial expressions, voice tone, and other cues.
11. **Game Playing and AI:** This sub-area involves creating AI agents that can play games at a high level of proficiency, often serving as benchmarks for AI research.
12. **Multi-Agent Systems:** Multi-agent systems study how multiple intelligent agents interact and collaborate in complex environments.
13. **Ethics and Fairness in AI:** This area addresses the ethical implications and potential biases in AI systems, as well as ways to ensure fairness and accountability.

These sub-areas often intersect and contribute to the advancement of AI as a whole. Researchers and practitioners in AI may specialize in one or more of these sub-areas to develop expertise and contribute to the field's progress.

1.1.5 Applications

Q6. Explain the applications of AI.

Ans : **(Imp.)**

AI has been dominant in various fields such as :

- **Gaming** : AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.
- **Natural Language Processing :** It is possible to interact with the computer that understands natural language spoken by humans.
- **Expert Systems :** There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.
- **Vision Systems** : These systems understand, interpret, and comprehend visual input on the computer. For example,
  - A spying aeroplane takes photographs, which are used to figure out spatial information or map of the areas.
  - Doctors use clinical expert system to diagnose the patient.
  - Police use computer software that can recognize the face of criminal with the stored portrait made by forensic artist.
- **Speech Recognition** : Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.
- **Handwriting Recognition** : The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.
- **Intelligent Robots** : Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such

as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

1.1.6 Problem Saving

Q7. What is problem in AI? Write the steps for problem solving in AI.

Ans : **(Imp.)**

The reflex agent of AI directly maps states into action. Whenever these agents fail to operate in an environment where the state of mapping is too large and not easily performed by the agent, then the stated problem dissolves and sent to a problem-solving domain which breaks the large stored problem into the smaller storage area and resolves one by one. The final integrated action will be the desired outcomes.

On the basis of the problem and their working domain, different types of problem-solving agent defined and use at an atomic level without any internal state visible with a problem-solving algorithm. The problem-solving agent performs precisely by defining problems and several solutions. So we can say that problem solving is a part of artificial intelligence that encompasses a number of techniques such as a tree, B-tree, heuristic algorithms to solve a problem.

We can also say that a problem-solving agent is a result-driven agent and always focuses on satisfying the goals.

There are basically three types of problem in artificial intelligence:

1. **Ignorable:** In which solution steps can be ignored.
2. **Recoverable:** In which solution steps can be undone.
3. **Irrecoverable:** Solution steps cannot be undo.

Steps problem-solving in AI: The problem of AI is directly associated with the nature of humans and their activities. So we need a number of finite steps to solve a problem which makes human easy works.

These are the following steps which require to solve a problem :

- **Problem definition :** Detailed specification of inputs and acceptable system solutions.
- **Problem analysis:** Analyse the problem thoroughly.
- **Knowledge Representation:** collect detailed information about the problem and define all possible techniques.
- **Problem-solving:** Selection of best techniques.

Components to formulate the associated problem:

- **Initial State:** This state requires an initial state for the problem which starts the AI agent towards a specified goal. In this state new methods also initialize problem domain solving by a specific class.
- **Action:** This stage of problem formulation works with function with a specific class taken from the initial state and all possible actions done in this stage.
- **Transition:** This stage of problem formulation integrates the actual action done by the previous action stage and collects the final stage to forward it to their next stage.
- **Goal test:** This stage determines that the specified goal achieved by the integrated transition model or not, whenever the goal achieves stop the action and forward into the next stage to determines the cost to achieve the goal.
- **Path costing:** This component of problem-solving numerical assigned what will be the cost to achieve the goal. It requires all hardware software and human working cost.

1.2 State - Space Search and Control Strategies

1.2.1 Introduction

Q8. What is state space search ? Explain the feqtures and steps in state space search.

Ans :

Meaning

State space search is a problem-solving technique used in Artificial Intelligence (AI) to find the solution path from the initial state to the goal state by exploring the various states. The state space search approach searches through all possible states of a problem to find a solution.

A state space is a way to mathematically represent a problem by defining all the possible states in which the problem can be. This is used in search algorithms to represent the initial state, goal state, and current state of the problem. Each state in the state space is represented using a set of variables.

The efficiency of the search algorithm greatly depends on the size of the state space, and it is important to choose an appropriate representation and search strategy to search the state space efficiently.

One of the most well-known state space search algorithms is the A algorithm. Other commonly used state space search algorithms include breadth-first search (BFS), depth-first search (DFS), hill climbing, simulated annealing, and genetic algorithms.

Features

State space search has several features that make it an effective problem-solving technique in Artificial Intelligence. These features include:

- **Exhaustiveness :** State space search explores all possible states of a problem to find a solution.
- **Completeness:** If a solution exists, state space search will find it.
- **Optimality:** Searching through a state space results in an optimal solution.
- **Uninformed and Informed Search:** State space search in artificial intelligence can be classified as uninformed if it provides additional information about the problem.

In contrast, informed search uses additional information, such as heuristics, to guide the search process.

Steps

The steps involved in state space search are as follows:

- To begin the search process, we set the current state to the initial state.
- We then check if the current state is the goal state. If it is, we terminate the algorithm and return the result.
- If the current state is not the goal state, we generate the set of possible successor states that can be reached from the current state.
- For each successor state, we check if it has already been visited. If it has, we skip it, else we add it to the queue of states to be visited.
- Next, we set the next state in the queue as the current state and check if it's the goal state. If it is, we return the result. If not, we repeat the previous step until we find the goal state or explore all the states.
- If all possible states have been explored and the goal state still needs to be found, we return with no solution.

1.2.2 General Problem Solving

Q9. Explain the steps needed to build a system to solve a particular problem.

Ans :

The steps that are required to build a system to solve a particular problem are:

1. Problem Definition that must include precise specifications of what the initial situation will be as well as what final situations constitute acceptable solutions to the problem.
2. Problem Analysis , this can have immense impact on the appropriateness of varies possible techniques for solving the problem.
3. Selection of the best technique(s) for solving the particular problem.

Define the Problem as State Space Search

Consider the problem of "Playing Chess" . to build a program that could play chess, we have to specify the starting position of the chess board, the rules that define legal moves. And the board position that represent a win. The goal of the winning the game, if possible, must be made explicit.

The starting position can be described by an 8 X 8 array square in which each element square (x,y),(x varying from 1to 8 & y varying from 1 to 8) describes the board position of an appropriate chess coin, the goal is any board position in which the opponent does not have a legal move and his or her "king" is under attack. The legal moves provide the way of getting from initial state of final state.

The legal moves can be described as a set of rules consisting of two parts: A left side that gives the current position and the right side that describes the

change to be made to the board position. An example is shown in the following figure.

Current Position

While pawn at square (5, 2), AND Square (5, 3) is empty, AND Square (5, 4). is empty.

Changing Board Position

Move pawn from Square (5, 2) to Square (5, 4).

The current position of a coin on the board is its STATE and the set of all possible STATES is STATE SPACE. One or more states where the problem terminates is FINAL STATE or GOAL STATE . The state space representation forms the basis of most of the AI methods. It allows for a formal definition of the problem as the need to convert some given situation into some desired situation using a set of permissible operations. It permits the problem to be solved with the help of known techniques and control strategies to move through the problem space until goal state is found.

Some of the problems that fall within the scope of AI and the kinds of techniques will be useful to solve these problems.

Q10. Explain about problem formulation.

Ans :

Problem Formulation:

- A problem formulation is about deciding what actions and states to consider, we will come to this point it shortly.
- We will describe our states as "in (CITYNAME)" where CITYNAME is the name of the city in which we are currently in.

Now suppose that our agent will consider actions of the from "Travel from city A to City B". and is standing in city 'A' and wants to travel to city 'E', which means that our current state is in(A) and we want to reach the state in(E).

There are 3 roads out of A, one toward B, one toward C and one toward D, none of these achieves the goal and will bring our agent to state in(E), given that our agent is not familiar with the geography of our alien map then it doesn't know which road is the best to take, so our agent will pick any road in random.

Now suppose that our agent is updated with the above map in its memory, the point of a map that our agent now knows what action bring it to what city, so our agent will start to study the map and consider a hypothetical journey through the map until it reaches E from A.

Once our agent has found the sequence of cities it should pass by to reach its goal it should start following this sequence.

The process of finding such sequence is called search, a search algorithm is like a black box which takes problem as input returns a solution, and once the solution is found the sequence of actions it recommends is carried out and this is what is called the execution phase.

We now have a simple (formulate, search, execute) design for our problem solving agent, so lets find out precisely how to formulate a problem.

Formulating problems

A problem can be defined formally by 4 components:

1. **Initial State**
   - it is the state from which our agents start solving the problem {e.i: in(A)}.
2. **State Description**
   - a description of the possible actions available to the agent, it is common to describe it by means of a successor function, given state x then SUCCESSOR-FN(x) returns a set of ordered pairs <action, successor> where action is a legal action from state x and successor is the state in which we can be by applying action.
   - The initial state and the successor function together defined what is called **state space** which is the set of all possible states reachable from the initial state {e.i: in(A), in(B), in(C), in(D), in(E)}.
3. **Goal Test**
   - we should be able to decide whether the current state is a goal state {e.i: is the current state is in(E)?}.
4. **Path cost**
   - a function that assigns a numeric value to each path, each step we take in solving the

problem should be somehow weighted, so If I travel from A to E our agent will pass by many cities, the cost to travel between two consecutive cities should have some cost measure, {e.i: Traveling from 'A' to 'B' costs 20 km or it can be typed as c(A, 20, B)}.

A solution to a problem is path from the initial state to a goal state, and solution quality is measured by the path cost, and the optimal solution has the lowest path cost among all possible solutions.

1.2.3 Characteristics of Problem

Q11. What are various types of problems ?

Ans :

There are different types of problem.

1. Single-state problem
2. Multiple-state problem
3. Contingency problem
4. Exploration problem

1. Single-state problem

Exact prediction is possible

State is known exactly after any sequence of actions.

Accessibility of the world all essential information can be obtained through sensors.

Consequences of actions are known to the agent.

Goal for each known initial state, there is a unique goal state that is guaranteed to be reachable via an action sequence. It is simplest case, but severely restricted.

Example :

Vacuum world

Limitations

- Can't deal with incomplete accessibility
- incomplete knowledge about consequences changes in the world
- indeterminism in the world, in action

2. Multiple-state problem

Semi-exact prediction is possible

- State is not known exactly, but limited to a set of possible states after each action.
- Accessibility of the world not all essential information can be obtained through sensors reasoning can be used to determine the set of possible states
- Consequences of actions are not always or completely known to the agent; actions or the environment might exhibit randomness
- Goal due to ignorance, there may be no fixed action sequence that leads to the goal less restricted, but more complex

Example :

Vacuum world, but the agent has no sensors

The action sequence right, suck, left, suck is guaranteed to reach the goal state from any initial state.

Limitations

- Can't deal with changes in the world during execution ("contingencies")

3. Contingency problem

exact prediction is impossible

- State unknown in advance, may depend on the outcome of actions and changes in the environment
- Accessibility of the world some essential information may be obtained through sensors only at execution time
- Consequences of action may not be known at planning time.
- Goal instead of single action sequences, there are trees of actions.
- Contingency branching point in the tree of actions.
- Agent design different from the previous two cases: the agent must act on incomplete plans. search and execution phases are interleaved.

Example :

Vacuum world, The effect of a suck action is random.

There is no action sequence that can be calculated at planning time and is guaranteed to reach the goal state.

- **Limitations**: Can't deal with situations in which the environment or effects of action are unknown

4. Exploration problem

Effects of actions are unknown

- State the set of possible states may be unknown
- Accessibility of the world some essential information may be obtained through sensors only at execution time
- Consequences of actions may not be known at planning time
- Goal can't be completely formulated in advance because states and consequences may not be known at planning time
- Discovery what states exist
- Experimentation what are the outcomes of actions.
- Learning remember and evaluate experiments.
- Agent design different from the previous cases: the agent must experiment.
- Search requires search in the real world, not in an abstract model realistic problems, very hard.

1.3 PROBLEM CHARACTERISTICS

Q12. Write about problem characteristics.

Ans :

Heuristic search is a very general method applicable to a large class of problem . It includes a variety of techniques. In order to choose an appropriate method, it is necessary to analyze the problem with respect to the following considerations.

1. Is the problem decomposable

A very large and composite problem can be easily solved if it can be broken into smaller problems and recursion could be used. Suppose we want to solve.

Ex:- + " x2 + 3x + sin2x cos 2x dx

This can be done by breaking it into three smaller problems and solving each by applying specific rules. Adding the results the complete solution is obtained.

2. Can solution steps be ignored or undone

Problem fall under three classes ignorable, recoverable and irrecoverable. This classification is with reference to the steps of the solution to a problem. Consider thermo proving. We may later find that it is of no help. We can still proceed further, since nothing is lost by this redundant step. This is an example of ignorable solutions steps.

Now consider the 8 puzzle problem tray and arranged in specified order. While moving from the start state towards goal state, we may make some stupid move and consider theorem proving. We may proceed by first proving lemma. But we may backtrack and undo the unwanted move. This only involves additional steps and the solution steps are recoverable.

Lastly consider the game of chess. If a wrong move is made, it can neither be ignored nor be recovered. The thing to do is to make the best use of current situation and proceed. This is an example of an irrecoverable solution steps.

i) Ignorable problems Ex: theorem proving

- In which solution steps can be ignored.

ii) Recoverable problems Ex: 8 puzzle

- In which solution steps can be undone

iii) Irrecoverable problems Ex: Chess

- In which solution steps can't be undone

A knowledge of these will help in determining the control structure.

3 Is the Universal Predictable

Problems can be classified into those with certain outcome (eight puzzle and water jug problems) and those with uncertain outcome (playing cards) in certain – outcome problems, planning could be done to generate a sequence of operators that guarantees to a lead to a solution. Planning helps to avoid unwanted solution steps. For uncertain out come problems, planning can at best generate a sequence of operators that has a good probability of leading to a solution. The uncertain outcome problems do not guarantee a solution and it is often very expensive since the number of solution and it is often very expensive since the number of solution paths to be explored increases exponentially with the number of points at which the outcome can not be predicted. Thus one of the hardest types of problems to solve is the irrecoverable, uncertain – outcome problems (Ex: Playing cards).

4. Is good solution absolute or relative (Is the solution a state or a path ?)

There are two categories of problems. In one, like the water jug and 8 puzzle problems, we are satisfied

with the solution, unmindful of the solution path taken, whereas in the other category not just any solution is acceptable. We want the best, like that of traveling sales man problem, where it is the shortest path. In any – path problems, by heuristic methods we obtain a solution and we do not explore alternatives. For the best-path problems all possible paths are explored using an exhaustive search until the best path is obtained.

5. The knowledge base consistent

In some problems the knowledge base is consistent and in some it is not. For example consider the case when a Boolean expression is evaluated. The knowledge base now contains theorems and laws of Boolean Algebra which are always true. On the contrary consider a knowledge base that contains facts about production and cost. These keep varying with time. Hence many reasoning schemes that work well in consistent domains are not appropriate in inconsistent domains.

Ex : Boolean expression evaluation.

6. What is the role of Knowledge

Though one could have unlimited computing power, the size of the knowledge base available for solving the problem does matter in arriving at a good solution. Take for example the game of playing chess, just the rues for determining legal moves and some simple control mechanism is sufficient to arrive at a solution. But additional knowledge about good strategy and tactics could help to constrain the search and speed up the execution of the program. The solution would then be realistic.

Consider the case of predicting the political trend. This would require an enormous amount of knowledge even to be able to recognize a solution , leave alone the best.

Ex : **1.** Playing chess

2. News paper understanding

7. Does the task requires interaction with the person

The problems can again be categorized under two heads.

i) Solitary in which the computer will be given a problem description and will produce an answer, with no intermediate communication and with he demand for an explanation of the reasoning process. Simple theorem proving falls under this category . given the basic rules and laws, the theorem could be proved, if one exists.

Ex: theorem proving (give basic rules & laws to computer)

ii) Conversational, in which there will be intermediate communication between a person and the computer, wither to provide additional assistance to the computer or to provide additional informed information to the user, or both problems such as medical diagnosis fall under this category, where people will be unwilling to accept the verdict of the program, if they can not follow its reasoning.

Ex: Problems such as medical diagnosis.

8. Problem Classification

Actual problems are examined from the point of view, the task here is examine an input and decide which of a set of known classes.

Ex: Problems such as medical diagnosis, engineering design.

Q13. Explain how to analyse and represent the problem.

Ans :

This problem can be abstracted to the mathematical problem of finding a path from a start node to a goal node in a directed graph. Many other problems can also be mapped to this abstraction, so it is worthwhile to consider this level of abstraction. Most of this chapter explores various algorithms for finding such paths.

This notion of search is computation inside the agent. It is different from searching in the world, when it may have to act in the world, for example, an agent searching for its keys, lifting up cushions, and so on. It is also different from searching the web, which involves searching for information. Searching in this chapter means searching in an internal representation for a path to a goal.

The idea of search is straightforward: the agent constructs a set of potential partial solutions to a problem that can be checked to see if they truly are solutions or if they could lead to solutions. Search proceeds by repeatedly selecting a partial solution, stopping if it is a path to a goal, and otherwise extending it by one more arc in all possible ways.

Search underlies much of artificial intelligence. When an agent is given a problem, it is usually given only a description that lets it recognize a solution, not an algorithm to solve it. It has to search for a solution.

The existence of NP-complete problems, with efficient means to recognize answers but no efficient methods for finding them, indicates that searching is, in many cases, a necessary part of solving problems.

It is often believed that humans are able to use intuition to jump to solutions to difficult problems. However, humans do not tend to solve general problems; instead they solve specific instances about which they may know much more than the underlying search space. Problems in which little structure exists or in which the structure cannot be related to the physical world are very difficult for humans to solve. The existence of public key encryption codes, where the search space is clear and the test for a solution is given - for which humans nevertheless have no hope of solving and computers cannot solve in a realistic time frame - demonstrates the difficulty of search.

The difficulty of search and the fact that humans are able to solve some search problems efficiently suggests that computer agents should exploit knowledge about special cases to guide them to a solution. This extra knowledge beyond the search space is **heuristic knowledge**. This chapter considers one kind of heuristic knowledge in the form of an estimate of the cost from a node to a goal.

1.3.1 Exhaustive Searches

Q14. What is Exhaustive search? Explain about it.

Ans : **(Imp.)**

Exhaustive search is a simple but potentially resource-intensive method used in artificial intelligence (AI) to solve problems by systematically exploring all possible solutions or states in a search space.

Exhaustive Search is a brute-force algorithm that systematically enumerates all possible solutions to a problem and checks each one to see if it is a valid solution. This algorithm is typically used for problems that have a small and well-defined search space, where it is feasible to check all possible solutions.

In other words, it involves examining every possible option to find the optimal solution. While exhaustive search guarantees finding the best solution (if one exists), it can be impractical for large or complex search spaces due to its computational demands.

Here's how exhaustive search works:

1. **Problem Definition:** Clearly define the problem, including the goal, initial state, and constraints.
2. **Generate Options:** Enumerate or generate all possible options or solutions within the problem's search space. This can involve generating permutations, combinations, sequences, or other variations depending on the nature of the problem.
3. **Evaluate Options:** Evaluate each generated option using an objective function or some criteria to determine its quality or suitability. This evaluation can involve calculations, comparisons, or other measurements.
4. **Select Best Solution:** Identify the best solution based on the evaluation. This could mean selecting the option with the highest score, the lowest cost, or satisfying specific criteria.
5. **Complexity Considerations:** Exhaustive search becomes impractical when the search space is large or infinite, as evaluating all possible options may require an unreasonable amount of time, memory, or computational resources.

Exhaustive search is most suitable for small and well-defined problems where the search space is manageable. It is commonly used in cases where the number of possible solutions is relatively small and when optimality is critical. However, as the problem size increases, the number of possibilities grows exponentially, leading to a combinatorial explosion that makes exhaustive search infeasible.

Examples:

Input: Items[] = {1, 2, 3, 4, 5, 6};

List of sets = {1, 2, 3}, {4, 5}, {5, 6}, {1, 4}

Output: Maximum number of sets that can be packed: 3

Input: Items[] = {1, 2};

List of sets = {1}, {4, }, {5}, {1}

Output: Maximum number of sets that can be packed: 1

Approach : Loop through all the sets and check if the current item is in the current set If the item is in the set, increment the number of sets that can be packed and Update the maximum number of sets that can be packed.

Here is a step-by-step description of how the algorithm works in this code:

- The program defines a max Packed Sets() function that takes a set of items and a list of sets as input.

- The function initializes the maximum number of sets that can be packed to 0.
- The function loops through all the sets in the list of sets. For each set, it initializes the number of sets that can be packed to 0.
- The function then loops through all the items in the set of items. For each item, it checks if the item is in the current set. If the item is in the set, the number of sets that can be packed is incremented and the item is removed from the set of items so that it is not counted again.
- The function then updates the maximum number of sets that can be packed by taking the maximum of the current maximum and the number of sets that can be packed for the current set.
- The function repeats steps 3-5 for all the sets in the list of sets.
- Once all the sets have been processed, the function returns the maximum number of sets that can be packed as the result.
- The main() function of the program creates a set of items and a list of sets and then calls the maxPackedSets() function to find the maximum number of sets that can be packed into the given set of items.
- The result of the max PackedSets() function is printed to the console, indicating the maximum number of sets that can be packed.

Characteristics of Exhaustive Search Algorithm:

- The Exhaustive Search algorithm is a simple and straightforward approach to solving problems.
- However, it can be slow and computationally expensive, especially for problems with a large search space.
- It is also not guaranteed to find the optimal solution to a problem, as it only checks solutions that are explicitly enumerated by the algorithm.
- Despite these limitations, Exhaustive Search can be a useful technique for solving certain types of problems.

Let's explore an examples of an exhaustive search in AI:

Example: Traveling Salesman Problem (TSP)

The Traveling Salesman Problem is a classic optimization problem where a salesman needs to visit a set of cities exactly once and return to the starting city while minimizing the total distance traveled. Let's consider a simplified version of the problem with four cities: A, B, C, and D.

1. Problem Definition

Cities: A, B, C, D

Distances between cities (in arbitrary units):

A to B: 10

A to C: 15

A to D: 20

B to C: 25

B to D: 30

C to D: 35

2. Generate Options: Enumerate all possible routes that the salesman can take, considering that the starting and ending city must be the same. In this case, there are 3! (3 factorial) possible routes, as follows:

A -> B -> C -> D -> A

A -> B -> D -> C -> A

A -> C -> B -> D -> A

A -> C -> D -> B -> A

A -> D -> B -> C -> A

A -> D -> C -> B -> A

3. Evaluate Options: Calculate the total distance for each route by summing the distances between consecutive cities. For example:

A -> B -> C -> D -> A: 10 + 25 + 35 + 20 = 90

A -> B -> D -> C -> A: 10 + 30 + 15 + 20 = 75

...and so on for all routes.

4. Select Best Solution: Choose the route with the shortest total distance as the optimal solution:

A -> B -> D -> C -> A: 10 + 30 + 15 + 20 = 75 (shortest distance)

In this simplified example, an exhaustive search was feasible because there were only six possible routes to consider. However, as the number of cities increases, the number of possible routes grows rapidly, making an

exhaustive search impractical. For instance, with 10 cities, there are 9! (362,880) possible routes to evaluate, and the problem quickly becomes computationally intractable.

To handle larger instances of the Traveling Salesman Problem and similar optimization problems, more efficient algorithms like dynamic programming, branch and bound, or heuristic-based methods are often employed to find near-optimal solutions without having to examine every possible combination exhaustively.

Q15. Solve Kanpsack problem using Exhaustic search method.

Ans : **(Imp.)**

The Knapsack Problem is a classic optimization problem in computer science and mathematics. It involves selecting a subset of items from a given set, each with a specific weight and value, in order to maximize the total value while staying within a given weight capacity. The problem is often used to illustrate concepts of optimization and dynamic programming. Let's go through an example of the Knapsack Problem:

Example: Knapsack Problem

Suppose you are a thief planning to rob a jewelry store. You have a knapsack with a maximum weight capacity of 10 kilograms, and you want to maximize the total value of the items you steal. You are presented with the following items:

Item	Weight (kg)	Value ($)
Gold	3	50
Silver	4	40
Diamond	5	100
Watch	1	10
Ruby	2	60

You need to decide which items to steal to maximize the total value while staying within the weight capacity of your knapsack.

Solution:

To solve the Knapsack Problem, we can use dynamic programming to build a table that helps us make decisions about which items to include in the knapsack. Here's the process:

1. **Create a Table:** Create a table where each cell (i, w) represents the maximum value that can be obtained using the first i items, given a knapsack with a weight capacity of w.
2. **Fill in the Table:** Start from the first item and iteratively fill in the table based on the following recurrence relation:

cssCopy code

```
Table[i][w] = max(Table[i-1][w], value[i] + Table[i-1][w - weight[i]])
```

- i represents the current item being considered.
- w represents the current weight capacity being considered.
- Table[i-1][w] represents the maximum value obtained without considering the current item.
- value[i] represents the value of the current item.
- weight[i] represents the weight of the current item.

3. **Construct the Solution:** Once the table is filled, backtrack through the table to determine which items were included in the optimal solution.

Table:

Item	Weight (kg)	Value ($)	Weight Capacity (w)	0	1	2	3	4	5	6	7	8	9	10
				0	0	0	0	0	0	0	0	0	0	0
Gold	3	50		0	0	0	50	50	50	50	50	50	50	50
Silver	4	40		0	0	0	50	50	50	90	90	90	90	90
Diamond	5	100		0	0	0	50	50	100	100	150	150	150	150
Watch	1	10		0	10	10	50	60	60	110	110	160	160	160
Ruby	2	60		0	10	60	60	70	110	110	160	170	220	220

Optimal Solution:

By examining the table, we can see that the optimal solution is to select the items "Silver," "Diamond," and "Ruby" to maximize the total value of $220 while staying within the weight capacity of 10 kilograms.

This example demonstrates how the Knapsack Problem can be solved using dynamic programming, making it possible to find the optimal combination of items to include in the knapsack while considering weight constraints and maximizing value.

1.3.2 Heuristic Search Techniques

Q16. What is a Heuristic Search ? Explain the techniques of Heuristic Search.

Ans :

Meaning

A Heuristic is a technique to solve a problem faster than classic methods, or to find an approximate solution when classic methods cannot. This is a kind of a shortcut as we often trade one of optimality, completeness, accuracy, or precision for speed.

A Heuristic (or a heuristic function) takes a look at search algorithms. At each branching step, it evaluates the available information and makes a decision on which branch to follow.

Heuristic Search Techniques

Here are some commonly used heuristic search techniques in AI:

1. **Greedy Best-First Search:** Greedy best-first search selects the most promising node based on a heuristic evaluation function that estimates the cost from the current node to the goal. It prioritizes nodes that appear to lead toward the goal, but it can get stuck in local optima and may not find the optimal solution.
2. ***A Search :**** A* (pronounced "A star") is an informed search algorithm that combines the advantages of both breadth-first and greedy best-first search. It evaluates nodes based on both the heuristic estimate of the cost to reach the goal and the cost incurred so far to reach the current node. A* is guaranteed to find an optimal solution if certain conditions are met.
3. ***IDA (Iterative Deepening A) :**** IDA* is a memory-efficient variation of A* that uses iterative deepening to explore the search space. It iteratively increases the threshold for the heuristic function until a solution is found. IDA* is useful when memory limitations prevent the use of A*.
4. **Hill Climbing:** Hill climbing is a local search algorithm that starts at an initial solution and iteratively makes small improvements by moving to neighboring states that have better heuristic values. However, hill climbing can get stuck in local optima and may not explore the entire search space.

5. **Simulated Annealing:** Simulated annealing is a probabilistic optimization technique inspired by the annealing process in metallurgy. It allows the algorithm to escape local optima by accepting worse solutions with decreasing probability over time. The probability of accepting a worse solution depends on the current temperature parameter.
6. **Genetic Algorithms:** Genetic algorithms are inspired by natural selection and evolution. They maintain a population of potential solutions and use selection, crossover, and mutation operations to evolve and improve the solutions over successive generations.
7. **Particle Swarm Optimization (PSO):** PSO is inspired by the social behavior of birds or fish. Particles (potential solutions) move through the search space, adjusting their positions based on their own best performance and the best performance of neighboring particles.
8. **Ant Colony Optimization (ACO):** ACO is inspired by the foraging behavior of ants. It uses artificial ants to explore a search space and deposit pheromones on paths. The probability of selecting a path is influenced by the pheromone levels.
9. **Beam Search:** Beam search is a variation of breadth-first search that keeps a fixed number (beam width) of the most promising nodes at each level. It prunes unpromising branches and explores only a subset of the search space, which can be helpful in reducing memory and time requirements.

Q17. Explain about generate and test algorithm.

Ans :

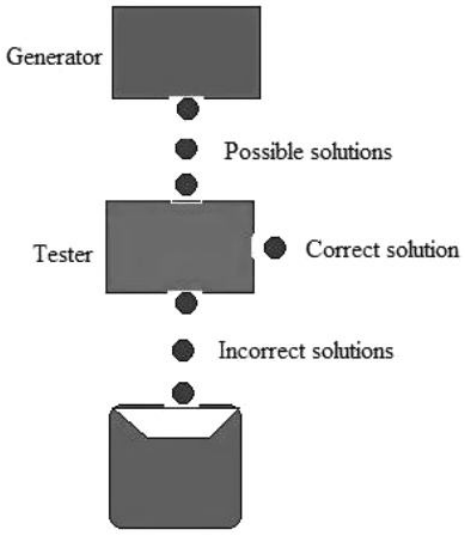
Heuristic search is an AI search technique that employs heuristic for its moves. Heuristic is a rule of thumb that probably leads to a solution. Heuristics play a major role in search strategies because of exponential nature of the most problems. Heuristics help to reduce the number of alternatives from an exponential number to a polynomial number. In Artificial Intelligence, heuristic search has a general meaning, and a more specialized technical meaning. In a general sense, the term heuristic is used for any advice that is often effective, but is not guaranteed to work in every case. Within the heuristic search architecture, however, the term heuristic usually refers to the special case of a heuristic evaluation function.

Generate and Test search algorithm

Generate-and-test search algorithm is a very simple algorithm that guarantees to find a solution if done systematically and there exists a solution.

Algorithm: Generate-And-Test

1. Generate a possible solution.
2. Test to see if this is the expected solution.
3. If the solution has been found quit else go to step 1.



Potential solutions that need to be generated vary depending on the kinds of problems. For some problems the possible solutions may be particular points in the problem space and for some problems, paths from the start state.

Generate-and-test, like depth-first search, requires that complete solutions be generated for testing. In its most systematic form, it is only an exhaustive search of the problem space. Solutions can also be generated randomly but solution is not guaranteed. This approach is what is known as British Museum algorithm: finding an object in the British Museum by wandering randomly.

Q18. Explain about Hill climbing algorithm. State its advantages and disadvantages.

Ans :

Meaning

Hill climbing search algorithm is simply a loop that continuously moves in the direction of increasing value. It stops when it reaches a "peak" where no neighbour has higher value. This algorithm is considered

to be one of the simplest procedures for implementing heuristic search. The hill climbing comes from that idea if you are trying to find the top of the hill and you go up direction from where ever you are. This heuristic combines the advantages of both depth first and breadth first searches into a single method.

The name hill climbing is derived from simulating the situation of a person climbing the hill. The person will try to move forward in the direction of at the top of the hill. His movement stops when it reaches at the peak of hill and no peak has higher value of heuristic function than this. Hill climbing uses knowledge about the local terrain, providing a very useful and effective heuristic for eliminating much of the unproductive search space. It is a branch by a local evaluation function.

The hill climbing is a variant of generate and test in which direction the search should proceed. At each point in the search path, a successor node that appears to reach for exploration.

Algorithm:

Step 1: Evaluate the starting state. If it is a goal state then stop and return success.

Step 2: Else, continue with the starting state as considering it as a current state.

Step 3: Continue step-4 until a solution is found i.e. until there are no new states left to be applied in the current state.

Step 4:

a. Select a state that has not been yet applied to the current state and apply it to produce a new state.

b. Procedure to evaluate a new state.

- **i.** If the current state is a goal state, then stop and return success.
- **ii.** If it is better than the current state, then make it current state and proceed further.
- **iii.** If it is not better than the current state, then continue in the loop until a solution is found.

Step 5: Exit.

Advantages

- Hill climbing technique is useful in job shop scheduling, automatic programming, designing, and vehicle routing and portfolio management.
- It is also helpful to solve pure optimization problems where the objective is to find the best according to the objective function.
- It requires much less conditions than other search techniques.

Disadvantages

- The question that remains on hill climbing search is whether this hill is the highest hill possible.
- Unfortunately without further extensive exploration, this question cannot be answered.
- This technique works but as it uses local information that's why it can be fooled.
- The algorithm doesn't maintain a search tree, so the current node data structure need only record the state and its objective function value.
- It assumes that local improvement will lead to global improvement.

Q19. Explain Best- Frist Search algorithm.

Ans :

Best first search is an instance of graph search algorithm in which a node is selected for expansion based on evaluation function f (n). Traditionally, the node which is the lowest evaluation is selected for the explanation because the evaluation measures distance to the goal. Best first search can be implemented within general search frame work via a priority queue, a data structure that will maintain the fringe in ascending order off values. This search algorithm serves as combination of depth first and breadth first search algorithm. Best first search algorithm is often referred greedy algorithm this is because they quickly attack the most desirable path as soon as its heuristic weight becomes the most desirable.

Concept

Step 1:

Traverse the root node

Step 2:

Traverse any neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue.

Step 3:

Traverse any neighbour of neighbour of the root node, that is maintaining a least distance from the root node and insert them in ascending order into the queue

Step 4:

This process will continue until we are getting the goal node.

Algorithm

Step 1:

Place the starting node or root node into the queue.

Step 2:

If the queue is empty, then stop and return failure.

Step 3:

If the first element of the queue is our goal node, then stop and return success.

Step 4:

Else, remove the first element from the queue. Expand it and compute the estimated goal distance for each child. Place the children in the queue in ascending order to the goal distance.

Step 5:

Go to step-3

Step 6:

Exit

1.3.3 Iterativer-Deepening

Q20. Explain Iterative Deepening Depth-First Search algorithm with an example

Ans :

Iterative Deepening Depth-First Search (IDDFS) is a search algorithm that performs depth-first search (DFS) in a tree or graph with an increasing depth limit until the goal state is found. The working algorithm of IDDFS can be described as follows:

1. Initialize the depth limit to 0.
2. Repeat the following steps for increasing values of the depth limit until the goal state is found or there are no more nodes to explore: a. Perform a depth-first search on the tree or graph with the current depth limit. b. If the goal state is found, return the solution. c. If there are no more nodes to explore, return failure.
3. If the goal state is not found after exploring all nodes up to a certain depth limit, increase the depth limit and repeat steps 2a-2c.

The advantage of IDDFS over a traditional DFS is that it gradually increases the depth limit of the search, rather than using a fixed depth limit. This means that IDDFS can find the shortest path to the goal state in a tree or graph with an unknown depth.

The time complexity of IDDFS is O(b ^ d), where b is the branching factor of the search tree or graph, and d is the depth of the goal state. However, the space complexity of IDDFS is only O(d) since it only keeps track of the current path being explored.

Pseudo-code for IDDFS

1. IDDFS(T):
2. for d = 0 to infinity:
3. if (DLS(T, d)):
4. return 1
5. else
6. return 0

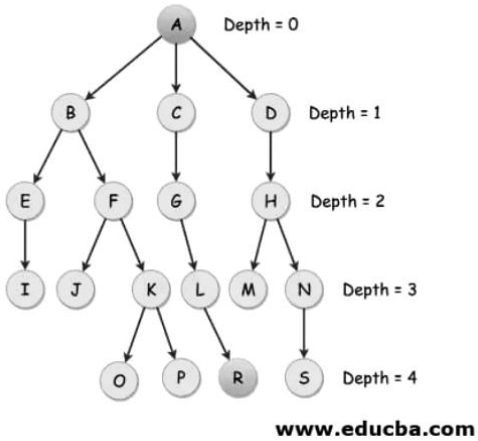
In order to implement the iterative deepening search we have to mark differences among:

1. Breakdown as the depth limit bound was attained.
2. A breakdown where depth bound was not attained.

While in the case once we try the search method multiple times by increasing the depth limit each time and in the second case even if we keep on searching multiple times since no solution exists then it means simply the waste of time. Thus we come to the conclusion that in the first case failure is found to be failing unnaturally, and in the second case, the failure is failing naturally.

Example

Let us take an example to understand this



Here in the given tree, the starting node is A and the depth initialized to 0. The goal node is R where we have to find the depth and the path to reach it. The depth from the figure is 4. In this example, we consider the tree as a finite tree, while we can consider the same procedure for the infinite tree as well. We knew that in the algorithm of IDDFS we first do DFS till a specified

depth and then increase the depth at each loop. This special step forms the part of DLS or Depth Limited Search. Thus the following traversal shows the IDDFS search.

The tree can be visited as: A B E F C G D H

DEPTH = {0, 1, 2, 3, 4}

DEPTH LIMITS	IDDFS
0	A
1	A B C D
2	A B E F C G D H
3	A B E I F J K C G L D H M N
4	A B E I F J K O P C G L R D H M N S

This gives us a glimpse of the IDDFS search pattern.

Q21. State the advantages and disadvantages.

Ans :

Advantages

- IDDFS gives us the hope to find the solution if it exists in the tree.
- When the solutions are found at the lower depths say n, then the algorithm proves to be efficient and in time.
- The great advantage of IDDFS is found in-game tree searching where the IDDFS search operation tries to improve the depth definition, heuristics, and scores of searching nodes so as to enable efficiency in the search algorithm.
- Another major advantage of the IDDFS algorithm is its quick responsiveness. The early results indications are a plus point in this algorithm. This followed up with multiple refinements after the individual iteration is completed.
- Though the work is done here is more yet the performance of IDDFS is better than single BFS and DFS operating exclusively.
- Space and time complexities are expressed as: O(d) and here d is defined as goal depth.
- Let us consider the run time of IDDFS. Let say b > l where b is branching factor and l is the depth limit. Then next we search the goal node under the bound k. On the depth k, we say there may be b^k nodes that are only generated once. Similarly, the nodes at the depth limit k-1 is twice and thrice for k-2 depth. Thus the node generated at depth l is k times.

Disadvantages

- The time taken is exponential to reach the goal node.
- The main problem with IDDFS is the time and wasted calculations that take place at each depth.
- The situation is not as bad as we may think of especially when the branching factor is found to be high.
- The IDDFS might fail when the BFS fails. When we are to find multiple answers from the IDDFS, it gives back the success nodes and its path once even if it needs to be found again after multiple iterations. To stop the depth bound is not increased further.

1.3.4 A*

Q22. Explain A* Algorithm.

Ans :

THE A* ALGORITHM:-

1. Start with OPEN containing the initial node. Its g = 0 and f ' = h '

   Set CLOSED to empty list.

2. Repeat

   If OPEN is empty , stop and return failure

   Else pick the BESTNODE on OPEN with lowest f ' value and place it on CLOSED

   If BESTNODE is goal state return success and stop

   Else

   Generate the successors of BESTNODE.

For each SUCCESSOR do the following:

1. Set SUCCESSOR to point back to BESTNODE. (back links will help to recover the path)
2. compute g(SUCCESSOR) = g(BESTNODE) cost of getting from BESTNODE to SUCCESSOR.
3. If SUCCESSOR is the same as any node on OPEN, call that node OLS and add OLD to BESTNODE 's successors. Check g(OLD) and

g(SUCCESSOR). It g(SUCCESSOR) is cheaper then reset OLD 's parent link to point to BESTNODE. Update g(OLD) and f '(OLD).

4. If SUCCESSOR was not on OPEN , see if it is on CLOSED . if so call the node CLOSED OLD , and better as earlier and set the parent link and g and f ' values appropriately.

5. If SUCCESSOR was not already on earlier OPEN or CLOSED, then put it on OPEN and add it to the list of BESTNODE 's successors.

   Compute f '(SUCCESSOR) = g (SUCCESSOR) + h ' (SUCCESSOR)

Best first searches will always find good paths to a goal after exploring the entire state space. All that is required is that a good measure of goal distance be used.

1.3.5 Constraint Satisfaction

Q23. What is constraint satisfaction problem? Explain it with an example.

(OR)

Explain CSP with the example of SEND + MORE = MONEY.

Ans : **(Imp.)**

A constraint satisfaction problem (CSP) consists of

- a set of variables,
- a domain for each variable, and
- a set of constraints.

The aim is to choose a value for each variable so that the resulting possible world satisfies the constraints; we want a model of the constraints.

A finite CSP has a finite set of variables and a finite domain for each variable. Many of the methods considered in this chapter only work for finite CSPs, although some are designed for infinite, even continuous, domains.

The multidimensional aspect of these problems, where each variable can be seen as a separate dimension, makes them difficult to solve but also provides structure that can be exploited.

Given a CSP, there are a number of tasks that can be performed:

- Determine whether or not there is a model.
- Find a model.
- Find all of the models or enumerate the models.
- Count the number of models.
- Find the best model, given a measure of how good models are;
- Determine whether some statement holds in all models.

Constraint Satisfaction Problem1.

Many problems in AI can be considered as problems of constraint satisfaction, in which the goal state satisfies a given set of constraint. constraint satisfaction problems can be solved by using any of the search strategies. The general form of the constraint satisfaction procedure is as follows:

Until a complete solution is found or until all paths have led to lead ends, do

1. select an unexpanded node of the search graph.

2. Apply the constraint inference rules to the selected node to generate all possible new constraints.

3. If the set of constraints contains a contradiction, then report that this path is a dead end.

4. If the set of constraints describes a complete solution then report success.

5. If neither a constraint nor a complete solution has been found then apply the rules to generate new partial solutions. Insert these partial solutions into the search graph.

Example: consider the crypt arithmetic problems.

```
  SEND
+ MORE
--------
 MONEY
--------
```

Assign decimal digit to each of the letters in such a way that the answer to the problem is correct to the same letter occurs more than once, it must be assign the same digit each time no two different letters may be assigned the same digit. Consider the crypt arithmetic problem.

```
  SEND
+ MORE
--------
 MONEY
--------
```

CONSTRAINTS :

1. no two digit can be assigned to same letter.

2. only single digit number can be assign to a letter.

3. no two letters can be assigned same digit.

4. Assumption can be made at various levels such that they do not contradict each other.

5. The problem can be decomposed into secured constraints. A constraint satisfaction approach may be used.

6. Any of search techniques may be used.

7. Backtracking may be performed as applicable us applied search techniques.

8. Rule of arithmetic may be followed.

Initial state of problem.

D=?

E=?

Y=?

N=?

R=?

O=?

S=?

M=?

C1=?

C2=?

C1 ,C 2, C3 stands for the carry variables respectively.

Goal State: the digits to the letters must be assigned in such a manner so that the sum is satisfied.

Solution Process :

We are following the depth-first method to solve the problem.

1. initial guess m=1 because the sum of two single digits can generate at most a carry '1'.

2. When n=1 o=0 or 1 because the largest single digit number added to m=1 can generate the sum of either 0 or 1 depend on the carry received from the carry sum. By this we conclude that o=0 because m is already 1 hence we cannot assign same digit another letter(rule no.)

3. We have m=1 and o=0 to get o=0 we have s=8 or 9, again depending on the carry received from the earlier sum.

The same process can be repeated further. The problem has to be composed into various constraints.

SOLUTION :

```
   9  5  6  7
+  1  0  8  5
---------------
1  0  6  5  2
```

VALUES :

S=9

E=5

N=6

D=7

M=1

O=0

R=8

Y=2

Q24. Explain, How to solve Constraint Satisfaction Problems using Search.

Ans :

Generate-and-test algorithms assign values to all variables before checking the constraints. Because individual constraints only involve a subset of the variables, some constraints can be tested before all of the variables have been assigned values. If a partial assignment is inconsistent with a constraint, any complete assignment that extends the partial assignment will also be inconsistent.

Example: In the delivery scheduling problem the assignments *A=1* and *B=1* are inconsistent with the constraint *A'"B* regardless of the values of the other variables. If the variables *A* and *B* are assigned values first, this inconsistency can be discovered before any values are assigned to *C*, *D*, or *E*, thus saving a large amount of work.

An alternative to generate-and-test algorithms is to construct a search space from which the search strategies of the previous chapter can be used. The search problem can be defined as follows:

- The nodes are assignments of values to some subset of the variables.
- The neighbors of a node *N* are obtained by selecting a variable *V* that is not assigned in

node N and by having a neighbor for each assignment of a value to V that does not violate any constraint.

Suppose that node N represents the assignment $X_1=v_1,\ldots,X_k=v_k$. To find the neighbors of N, select a variable Y that is not in the set $\{X_1,\ldots,X_k\}$. For each value y_i"$dom(Y)$, such that $X_1=v_1,\ldots, X_k=v_k, Y=y_i$ is consistent with the constraints, $X_1=v_1,\ldots,X_k=v_k, Y=y_i$ is a neighbor of N.

- The start node is the empty assignment that does not assign a value to any variables.
- A goal node is a node that assigns a value to every variable. Note that this only exists if the assignment is consistent with the constraints.

In this case, it is not the path that is of interest, but the goal nodes.

Example : Suppose you have a CSP with the variables A, B, and C, each with domain $\{1,2,3,4\}$. Suppose the constraints are $A<B$ and $B<C$. A possible search tree is shown in figure.

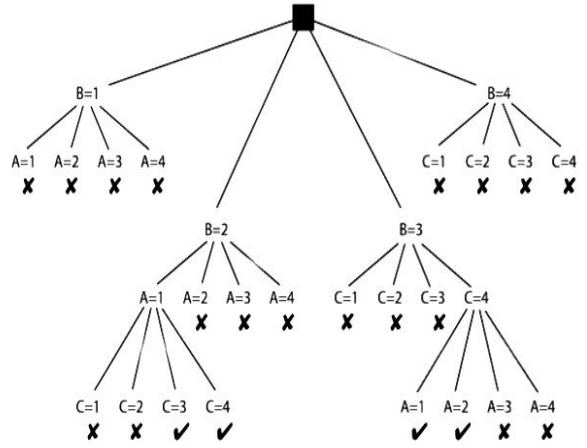


Figure 4.1: Search tree for the CSP of Example 4.13

In this figure, a node corresponds to all of the assignments from the root to that node. The potential nodes that are pruned because they violate constraints are labeled with ". The leftmost ' ' corresponds to the assignment $A=1$, $B=1$. This violates the $A<B$ constraint, and so it is pruned.

This CSP has four solutions. The leftmost one is $A=1$, $B=2$, $C=3$. The size of the search tree, and thus the efficiency of the algorithm, depends on which variable is selected at each time. A static ordering, such as always splitting on A then B then C, is less efficient than the dynamic ordering used here. The set of answers is the same regardless of the variable ordering.

In the preceding example, there would be $4^3=64$ assignments tested in a generate-and-test algorithm. For the search method, there are 22 assignments generated.

Searching with a depth-first search, typically called **backtracking**, can be much more efficient than generate and test. Generate and test is equivalent to not checking constraints until reaching the leaves. Checking constraints higher in the tree can prune large subtrees that do not have to be searched.

1.3.6 Game Playing

Q25. What is the importance of game playing in AI. Explain Min- Max Algorithm.

Ans : **(Imp.)**

Game Playing is an important domain of artificial intelligence. Games don't require much knowledge; the only knowledge we need to provide is the rules, legal moves and the conditions of winning or losing the game. Both players try to win the game. So, both of them try to make the best move possible at each turn. Searching techniques like BFS(Breadth First Search) are not accurate for this as the branching factor is very high, so searching will take a lot of time. So, we need another search procedures that improve –

- **Generate procedure** so that only good moves are generated.
- **Test procedure** so that the best move can be explored first.

Game playing is a popular application of artificial intelligence that involves the development of computer programs to play games, such as chess, checkers, or

Go. The goal of game playing in artificial intelligence is to develop algorithms that can learn how to play games and make decisions that will lead to winning outcomes.

1. One of the earliest examples of successful game playing AI is the chess program Deep Blue, developed by IBM, which defeated the world champion Garry Kasparov in 1997. Since then, AI has been applied to a wide range of games, including two-player games, multiplayer games, and video games.

There are two main approaches to game playing in AI, rule-based systems and machine learning-based systems.

1. **Rule-based systems** use a set of fixed rules to play the game.
2. **Machine learning-based systems** use algorithms to learn from experience and make decisions based on that experience.

In recent years, machine learning-based systems have become increasingly popular, as they are able to learn from experience and improve over time, making them well-suited for complex games such as Go. For example, AlphaGo, developed by DeepMind, was the first machine learning-based system to defeat a world champion in the game of Go.

Game playing in AI is an active area of research and has many practical applications, including game development, education, and military training. By simulating game playing scenarios, AI algorithms can be used to develop more effective decision-making systems for real-world applications.

The most common search technique in game playing is **Minimax search procedure**. It is depth-first depth-limited search procedure. It is used for games like chess and tic-tac-toe.

Minimax algorithm uses two functions –

- **MOVEGEN :** It generates all the possible moves that can be generated from the current position.
- **STATICEVALUATION :** It returns a value depending upon the goodness from the viewpoint of two-player

This algorithm is a two player game, so we call the first player as PLAYER1 and second player as PLAYER2. The value of each node is backed-up from its children. For PLAYER1 the backed-up value is the maximum value of its children and for PLAYER2 the backed-up value is the minimum value of its children. It provides most promising move to PLAYER1, assuming that the PLAYER2 has make the best move. It is a recursive algorithm, as same procedure occurs at each level.

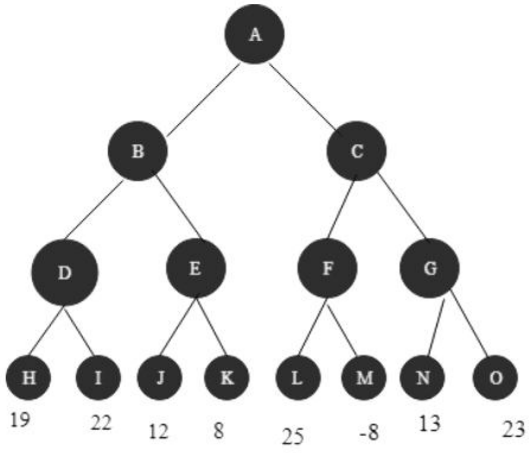


Fig.: 1. Before backing-up of values

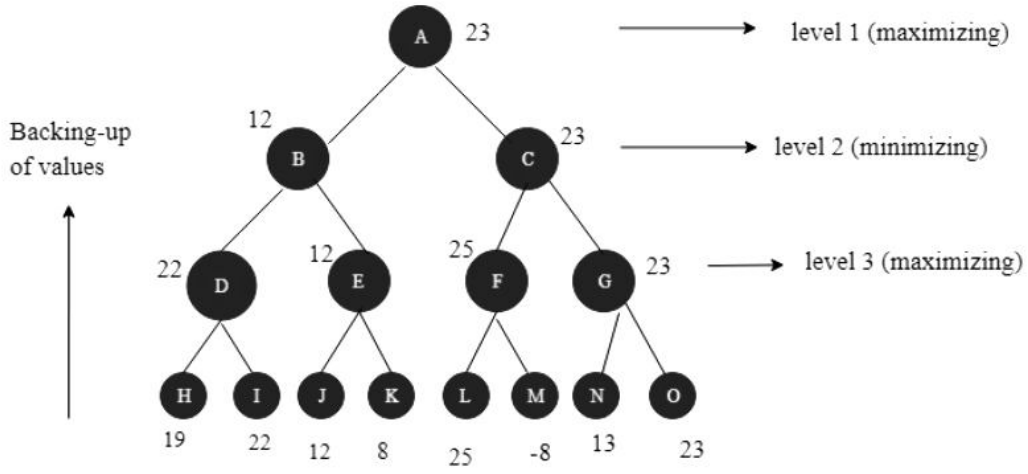


Fig.: 2. After backing-up of values We assume that PLAYER1 will start the game.

4 levels are generated. The value to nodes H, I, J, K, L, M, N, O is provided by STATICEVALUATION function. Level 3 is maximizing level, so all nodes of level 3 will take maximum values of their children. Level 2 is minimizing level, so all its nodes will take minimum values of their children. This process continues. The value of A is 23. That means A should choose C move to win.

Reference : Artificial Intelligence by Rich and Knight

Advantages of Game Playing in Artificial Intelligence:

1. **Advancement of AI:** Game playing has been a driving force behind the development of artificial intelligence and has led to the creation of new algorithms and techniques that can be applied to other areas of AI.
2. **Education and training:** Game playing can be used to teach AI techniques and algorithms to students and professionals, as well as to provide training for military and emergency response personnel.
3. **Research:** Game playing is an active area of research in AI and provides an opportunity to study and develop new techniques for decision-making and problem-solving.
4. **Real-world applications:** The techniques and algorithms developed for game playing can be applied to real-world applications, such as robotics, autonomous systems, and decision support systems.

Disadvantages of Game Playing in Artificial Intelligence:

1. **Limited scope:** The techniques and algorithms developed for game playing may not be well-suited for other types of applications and may need to be adapted or modified for different domains.
2. **Computational cost:** Game playing can be computationally expensive, especially for complex games such as chess or Go, and may require powerful computers to achieve real-time performance.

1.3.7 Bounded Look-Ahead Strategy and Use of Evaluation Functions

Q26. Explain Bounded look –ahead strategy with an example.

Ans : **(Imp.)**

The bounded look-ahead strategy and the use of evaluation functions are techniques commonly employed in artificial intelligence (AI) for decision-making in various domains, including game playing. These techniques help AI agents make informed choices by considering future possibilities and using a heuristic evaluation function to assess the desirability of different states. Let's explore these concepts with an example:

Example: Tic-Tac-Toe Game

Tic-Tac-Toe is a two-player game played on a 3x3 grid. Players take turns placing their symbols (X or O) in empty cells, with the goal of getting three of their symbols in a row, column, or diagonal. The first player to achieve this wins the game.

Bounded Look-Ahead Strategy

The bounded look-ahead strategy involves exploring a limited number of future moves or states to make a decision. It is used to avoid the computational complexity of examining the entire game tree. In this example, let's consider a bounded look-ahead strategy where the AI agent looks ahead two moves.

Use of Evaluation Function

An evaluation function assigns a value to a game state based on certain criteria. In Tic-Tac-Toe, the evaluation function can be a simple heuristic that measures the "goodness" of a position for the AI player. For example, the function could assign higher values to positions that are closer to winning (having two symbols in a row) and lower values to positions that are closer to losing (having two opponent symbols in a row). The AI aims to maximize its own score and minimize the opponent's score.

AI Agent's Decision Process

1. **Initial State:** The AI agent receives the current state of the Tic-Tac-Toe board and evaluates its available moves.
2. **Bounded Look-Ahead:** The AI agent explores the possible moves it can make and the opponent's potential responses for the next two moves (one move each).
3. **Evaluation Function:** For each move, the AI agent uses the evaluation function to assess the desirability of the resulting game states.
4. **Selection:** The AI agent selects the move that leads to the highest evaluated outcome or, alternatively, prevents the opponent from winning.
5. **Game Play:** The AI agent makes its move, updates the board state, and communicates its choice to the opponent (human or AI).

Example Move:

Suppose the AI agent is playing as X and has the following Tic-Tac-Toe board:

```
| X | O
_____
  |  |
_____
O| X |
```

The AI agent evaluates its available moves based on its bounded look-ahead strategy and evaluation function. It calculates the resulting positions after each move and their associated values based on the evaluation function.

Outcome:

Using its bounded look-ahead strategy and evaluation function, the AI agent may choose to place an X in the empty cell at the center of the board. This move prevents the opponent from having two symbols in a diagonal and creates the potential for a winning diagonal for the AI player in the next move.

Throughout the game, the AI agent continues to apply its bounded look-ahead strategy and evaluation function to make decisions that lead to a favorable outcome. This example illustrates how these techniques allow the AI agent to consider limited future possibilities and use heuristics to make informed choices in a game-playing scenario.

1.3.8 Alpha-Beta Pruning

Q27. Explain about alpha beta pruning technique.

Ans : **(Imp.)**

- Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm.
- As we have seen in the minimax search algorithm that the number of game states it has to examine are exponential in depth of the tree. Since we cannot eliminate the exponent, but we can cut it to half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameter Alpha and beta for future expansion, so it is called alpha-beta pruning. It is also called as Alpha-Beta Algorithm.
- Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prune the tree leaves but also entire sub-tree.

The two-parameter can be defined as:

a. Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -".

b. Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +".

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making algorithm slow. Hence by pruning these nodes, it makes the algorithm fast.

The main condition which required for alpha-beta pruning is:

$\alpha >= \beta$

Pseudo-code for Alpha-beta Pruning:

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
    if depth ==0 or node is a terminal node then
    return static evaluation of node

    if MaximizingPlayer then          // for Maximizer Player
  maxEva= -infinity
  for each child of node do
  eva= minimax(child, depth-1, alpha, beta, False)
 maxEva= max(maxEva, eva)
 alpha= max(alpha, maxEva)
  if beta<=alpha
break
return maxEva

else                                  // for Minimizer player
   minEva= +infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, true)
   minEva= min(minEva, eva)
   beta= min(beta, eva)
    if beta<=alpha
 break
return minEva
```

Q28. Explain about the working of alpha beta pruning with example

Ans :

Working of Alpha-Beta Pruning:

Let's take an example of two-player search tree to understand the working of Alpha-beta pruning.

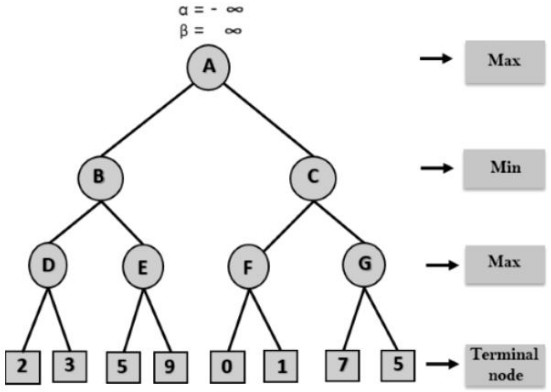
Let's take an example of two-player search tree to understand the working of Alpha-beta pruning

Step 1:

At the first step the, Max player will start first move from node A where $\alpha = -\infty$ and $\beta = +\infty$, these value of alpha and beta passed down to node B where again $\alpha = -\infty$ and $\beta = +\infty$ and Node B passes the same value to its child D.

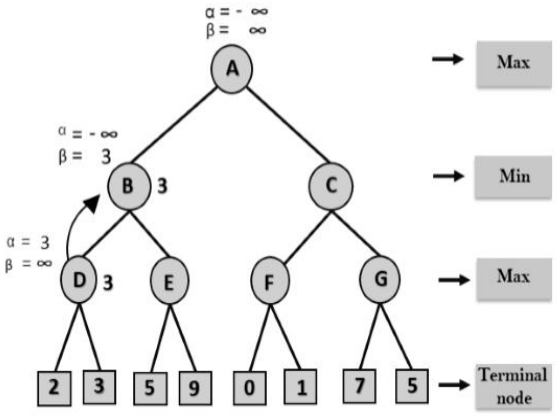
Step 2:

At Node D, the value of á will be calculated as its turn for Max. The value of α is compared with firstly 2 and then 3, and the max (2, 3) = 3 will be the value of α at node D and node value will also 3.



Step 3:

Now algorithm backtrack to node B, where the value of â will change as this is a turn of Min, Now $\beta = +\infty$, will compare with the available subsequent nodes value, i.e. min (∞ , 3) = 3, hence at node B now $\alpha = -\infty$, and α= 3.



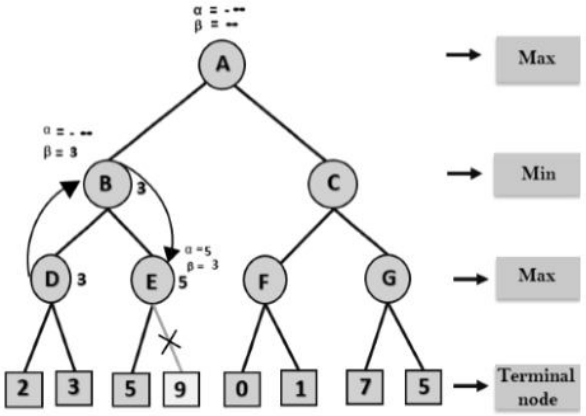
In the next step, algorithm traverse the next successor of Node B which is node E, and the values $\alpha = -\infty$, and $\beta = 3$ will also be passed.

Step 4:

At node E, Max will take its turn, and the value of alpha will change. The current value of alpha will be compared with 5, so max (-∞ , 5) = 5, hence at node E α= 5 and β= 3, where $\alpha >= \beta$, so the right successor of E will be pruned, and algorithm will not traverse it, and the value at node E will be 5.

Step 5:

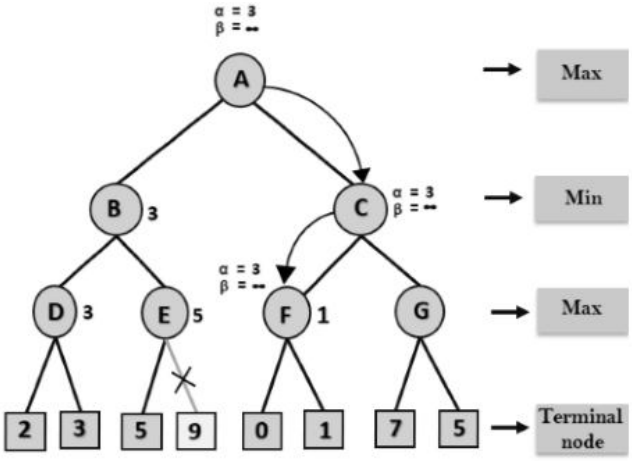
At next step, algorithm again backtrack the tree, from node B to node A. At node A, the value of alpha will be changed the maximum available value is 3 as max (-", 3)= 3, and â= +", these two values now passes to right successor of A which is Node C.



At node $\alpha = 3$ and $\beta = +\infty$, and the same values will be passed on to node F.

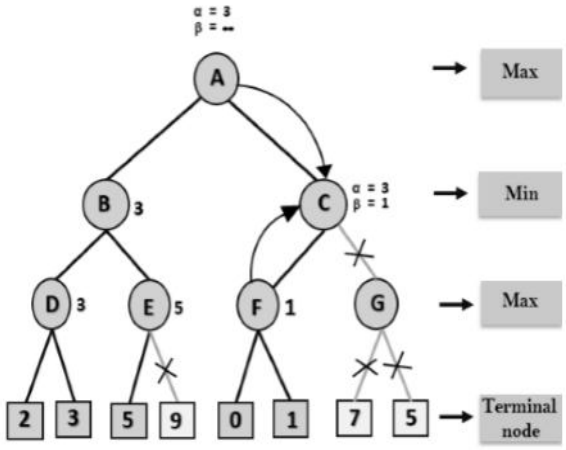
Step 6:

At node F, again the value of á will be compared with left child which is 0, and max(3,0)= 3, and then compared with right child which is 1, and max(3,1)= 3 still α remains 3, but the node value of F will become 1.



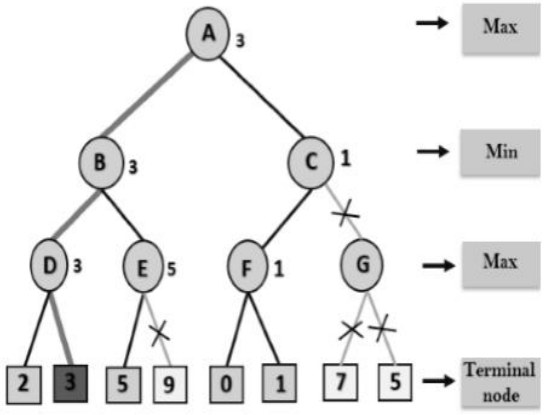
Step 7:

Node F returns the node value 1 to node C, at C α= 3 and β = + ∞, here the value of beta will be changed, it will compare with 1 so min (∞, 1) = 1. Now at C, ∞ =3 and b= 1, and again it satisfies the condition $\alpha >= \beta$, so the next child of C which is G will be pruned, and the algorithm will not compute the entire sub-tree G.



Step 8:

C now returns the value of 1 to A here the best value for A is max (3, 1) = 3. Following is the final game tree which is the showing the nodes which are computed and nodes which has never computed. Hence the optimal value for the maximizer is 3 for this example.



UNIT II

Logic Concepts and Logic Programming: Introduction, Propositional Calculus, Propositional Logic, Natural Deduction System, Axiomatic System, Semantic Tableau System in Propositional Logic, Resolution Refutation in Propositional Logic, Predicate Logic, Logic Programming.

Knowledge Representation: Introduction, Approaches to Knowledge Representation, Knowledge Representation using Semantic Network, Knowledge Representation using Frames

2.1 Logic Concepts And Logic Programming

2.1.1 Introduction

Q1. Explain briefly about logic programming?

Ans :

Logic Programming is a programming paradigm in which the problems are expressed as facts and rules by program statements but within a system of formal logic. Just like other programming paradigms like object oriented, functional, declarative, and procedural, etc., it is also a particular way to approach programming.

1. Inductive logic programming is worried about summing up negative and positive models with regards to foundation information: ML of logic programs.
2. Late work here, joining logic programming, probability and learning, has offered to ascend to the new field of probabilistic inductive logic programming and statistical learning.
3. Abductive logic programming is an undeniable level information portrayal structure that can be utilized to tackle issues definitively dependent on abductive thinking.
4. It expands ordinary logic programming by permitting a few predicates to be deficiently characterized, announced as inducible predicates.
5. Programmable Logic Array is a gate followed by a programmable and fixed architecture logic gadget with programmable or gates.

2.1.2 Propositional Calculus

Q2. What is propositional calculus? Explain the fundamental components of propositional calculus.

Ans : **(Imp.)**

Meaning

Propositional calculus, also known as propositional logic or sentential logic, is a branch of mathematical logic that deals with the study of propositions, which are statements or assertions that can be either true or false. In propositional calculus, we analyze the relationships between propositions using logical operators and rules.

These operators allow us to combine, modify, and reason about propositions systematically.

Explain the fundamental components of propositional calculus.

Let's explore the fundamental concepts of propositional calculus with examples:

1. Propositions

i) Propositions are basic statements that can be evaluated as either true or false.

Examples of propositions

(a) P: "The sky is blue." (This proposition can be either true or false)

(b) Q: "2 + 2 = 5." (This proposition is false)

ii) R: "It is raining." (This proposition can change its truth value over time)

2. Logical Connectives (Operators)

Propositional calculus introduces several logical connectives that allow us to form complex propositions from simpler ones. The main logical connectives are:

(a) Conjunction (AND, $\wedge$)

The conjunction of two propositions is true if and only if both propositions are true.

Example: If P is "The sun is shining" and Q is "It is a warm day," then P $\wedge$ Q is "The sun is shining, and it is a warm day."

(b) Disjunction (OR, $\vee$)

The disjunction of two propositions is true if at least one of the propositions is true.

Example: If P is "It is Monday" and Q is "It is a holiday," then P $\vee$ Q is "It is Monday or it is a holiday."

(c) Negation (NOT, $\neg$)

The negation of a proposition reverses its truth value.

Example: If P is "The store is open," then $\neg$P is "The store is not open."

(d) Implication ($\rightarrow$)

The implication of two propositions, P $\rightarrow$ Q, is true unless P is true, and Q is false.

Example: If P is "It is raining" and Q is "I will bring an umbrella," then P $\rightarrow$ Q is "If it is raining, then I will bring an umbrella."

(e) Biconditional ($\leftrightarrow$):

The biconditional of two propositions, P $\leftrightarrow$ Q, is true if both propositions have the same truth value.

Example: If P is "The traffic light is green" and Q is "I can cross the road," then P $\leftrightarrow$ Q is "I can cross the road if and only if the traffic light is green."

3. Truth Tables

i) Truth tables are a way to systematically analyze the truth values of compound propositions based on the truth values of their constituent propositions.

ii) Here's a truth table for the logical AND operator (conjunction):

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

4. This truth table shows all possible combinations of truth values for P and Q and the resulting truth value of P ∧ Q.

5. Propositional Logic Rules

i) In propositional calculus, we have rules and theorems that govern how propositions and logical connectives can be manipulated.

ii) Some common rules include the commutative, associative, and distributive laws.

Example of the distributive law:

(P ∧ Q) ∨ R is equivalent to (P ∨ R) ∧ (Q ∨ R).

6. Inference and Deduction

- Propositional calculus is used for making deductions and drawing conclusions based on given premises.
- For example, if we know that "If it is raining (P), then I will bring an umbrella (Q)," and we observe that "It is raining (P)" is true, we can deduce that "I will bring an umbrella (Q)" must also be true, using the implication operator.

Propositional calculus forms the foundation of more advanced branches of logic, such as predicate logic and first-order logic, which deal with more complex statements and quantifiers like "for all" (∀) and "there exists" (∃).

These systems are essential in various fields, including mathematics, computer science, and artificial intelligence, for modeling and reasoning about the world.

2.1.3 Propositional Logic

Q3. What is propositional logic? Explain the basic facts about propositional logic.

Ans : **(Imp.)**

Meaning

Propositional logic (PL) is the simplest form of logic where all the statements are made by propositions. A proposition is a declarative statement which is either true or false.

It is a technique of knowledge representation in logical and mathematical form.

Example

a) It is Sunday.

b) The Sun rises from West (False proposition)

c) 3+3= 7(False proposition)

d) 5 is a prime number.

Following are some basic facts about propositional logic

i) Propositional logic is also called Boolean logic as it works on 0 and 1.

ii) In propositional logic, we use symbolic variables to represent the logic, and we can use any symbol for a representing a proposition, such A, B, C, P, Q, R, etc.

iii) Propositions can be either true or false, but it cannot be both.

iv) Propositional logic consists of an object, relations or function, and logical connectives.

v) These connectives are also called logical operators.

vi) The propositions and connectives are the basic elements of the propositional logic.

vii) Connectives can be said as a logical operator which connects two sentences.

viii) A proposition formula which is always true is called tautology, and it is also called a valid sentence.

ix) A proposition formula which is always false is called Contradiction.

x) A proposition formula which has both true and false values is called

xi) Statements which are questions, commands, or opinions are not propositions such as "Where is Rohini", "How are you", "What is your name", are not propositions.

Syntax of Propositional Logic

The syntax of propositional logic defines the allowable sentences for the knowledge representation. There are two types of Propositions:

(a) Atomic Propositions

(b) Compound propositions

(a) Atomic Proposition

Atomic propositions are the simple propositions. It consists of a single proposition symbol. These are the sentences which must be either true or false.

Example

i) 2+ 2 is 4, it is an atomic proposition as it is a true fact.

ii) "The Sun is cold" is also a proposition as it is a false fact.

(b) Compound Proposition

Compound propositions are constructed by combining simpler or atomic propositions, using parenthesis and logical connectives.

Example

a) It is raining today, and street is wet.

b) Ankit is a doctor, and his clinic is in Mumbai.

Q4. Explain about propositional logical connectives and logical equivalances.

Ans :

Meaning

Logical connectives are used to connect two simpler propositions or representing a sentence logically. We can create compound propositions with the help of logical connectives.

There are mainly five connectives, which are given as follows:

1. Negation

A sentence such as ¬ P is called negation of P. A literal can be either Positive literal or negative literal.

2. Conjunction

A sentence which has $\wedge$ connective such as, P $\wedge$ Q is called a conjunction.

Example: Rohan is intelligent and hardworking. It can be written as,

P = Rohan is intelligent,

Q = Rohan is hardworking. $\rightarrow$ P $\wedge$ Q.

3. Disjunction

A sentence which has $\vee$ connective, such as P $\vee$ Q. is called disjunction, where P and Q are the propositions.

Example: "Ritika is a doctor or Engineer",

Here P = Ritika is Doctor.

Q = Ritika is Doctor, so we can write it as P $\vee$ Q.

4. Implication

A sentence such as P $\rightarrow$ Q, is called an implication. Implications are also known as if-then rules. It can be represented as

If it is raining, then the street is wet.

Let

P= It is raining, and Q= Street is wet, so it is represented as P $\rightarrow$ Q

5. Biconditional

A sentence such as P $\Leftrightarrow$ Q is a Biconditional sentence, example If I am breathing, then I am alive

P= I am breathing, Q= I am alive, it can be represented as P $\Leftrightarrow$ Q.

Following is the summarized table for Propositional Logic Connectives:

Connective symbols	Word	Technical term	Example
$\wedge$	AND	Conjunction	A $\wedge$ B
$\vee$	OR	Disjunction	A $\vee$ B
$\rightarrow$	Implies	Implication	A $\rightarrow$ B
$\Leftrightarrow$	If and onlyif	Biconditional	A $\Leftrightarrow$ B
$\neg$ or ~	Not	Negation	$\neg$ Aor $\neg$ B

Truth Table

In propositional logic, we need to know the truth values of propositions in all possible scenarios. We can combine all the possible combination with logical connectives, and the representation of these combinations in a tabular format is called Truth table. Following are the truth table for all logical connectives:

For Negation

P	$\neg$ P
True	False
False	True

For Conjunction

P	Q	P $\wedge$ Q
True	True	True
True	False	False
False	True	False
False	False	False

For disjunction

P	Q	P $\vee$ Q
True	True	True
False	True	True
True	False	True
False	False	False

For Implication

P	Q	P $\rightarrow$ Q
True	True	True
True	False	False
False	True	True
False	Flase	True

For Biconditional

P	Q	$P \Leftrightarrow Q$
True	True	True
True	False	False
False	True	False
False	False	True

Truth table with three Propositions

We can build a proposition composing three propositions P, Q, and R. This truth table is made-up of 8n Tuples as we have taken three proposition symbols.

P	Q	R	$\neg R$	$P \vee Q$	$P \vee Q \rightarrow \neg R$
True	True	True	False	True	False
True	True	False	True	True	True
True	False	True	False	True	False
True	False	False	True	True	True
False	True	True	False	True	False
False	True	False	True	True	True
False	False	True	False	False	True
False	False	False	True	False	True

Logical Equivalence

Logical equivalence is one of the features of propositional logic. Two propositions are said to be logically equivalent if and only if the columns in the truth table are identical to each other.

Let's take two propositions A and B, so for logical equivalence, we can write it as $A \Leftrightarrow B$. In below truth table we can see that column for $\neg A \vee B$ and $A \rightarrow B$, are identical hence A is Equivalent to B

A	B	$\neg A$	$\neg A \vee B$	$A \rightarrow B$
T	T	F	T	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Properties of Operators

Commutativity

$P \wedge Q = Q \wedge P$, or

$P \vee Q = Q \vee P$

Associativity

$(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$

$(P \vee Q) \vee R = P \vee (Q \vee R)$

Identity element

$P \wedge True = P$

$P \vee True = True$

Distributive

$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$

$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$

DE Morgan's Law

$\neg (P \wedge Q) = (\neg P) \vee (\neg Q)$

$\neg (P \vee Q) = (\neg P) \wedge (\neg Q)$.

Double-negation Elimination

$\neg (\neg P) = P$.

Limitations of Propositional Logic

We cannot represent relations like ALL, some, or none with propositional logic.

Example:

(a) All the girls are intelligent

(b) Some apples are sweet

Propositional logic has limited expressive power.

In propositional logic, we cannot describe statements in terms of their properties or logical relationships.

Q5. Define inference rules? Explain the types of inference rules.

Ans : **(Imp.)**

Meaning

Inference rules are the templates for generating valid arguments. Inference rules are applied to derive proofs in artificial intelligence, and the proof is a sequence of the conclusion that leads to the desired goal.

In inference rules, the implication among all the connectives plays an important role. Following are some terminologies related to inference rules:

Implication

It is one of the logical connectives which can be represented as $P \rightarrow Q$. It is a Boolean expression.

Converse

The converse of implication, which means the right-hand side proposition goes to the left-hand side and vice-versa. It can be written as $Q \rightarrow P$.

Contrapositive

The negation of converse is termed as contrapositive, and it can be represented as $\neg Q \rightarrow \neg P$.

Inverse

The negation of implication is called inverse. It can be represented as $\neg P \rightarrow \neg Q$.

From the above term some of the compound statements are equivalent to each other, which we can prove using truth table:

P	Q	$P \rightarrow Q$	$Q \rightarrow P$	$\neg Q \rightarrow \neg P$	$\neg P \rightarrow \neg Q$
T	T	T	T	T	T
T	F	F	T	F	T
F	T	T	F	T	F
F	F	T	T	T	T

Hence from the above truth table, we can prove that $P \rightarrow Q$ is equivalent to $\neg Q \rightarrow \neg P$, and $Q \rightarrow P$ is equivalent to $\neg P \rightarrow \neg Q$.

Types

1. Modus Ponens

The Modus Ponens rule is one of the most important rules of inference, and it states that if P and $P \rightarrow Q$ is true, then we can infer that Q will be true. It can be represented as:

Notation for Modus ponens : $\frac{P \rightarrow Q, P}{\therefore Q}$

Example:

Statement-1: "If I am sleepy then I go to bed" = = > $P \rightarrow Q$

Statement-2: "I am sleepy" $\Rightarrow P$

Conclusion: "I go to bed." $\Rightarrow Q$.

Hence, we can say that, if $P \rightarrow Q$ is true and P is true then Q will be true.

Proof by Truth table

P	Q	$P \rightarrow Q$
0	0	0
0	1	1
1	0	0
1	1	1

2. Modus Tollens

The Modus Tollens rule state that if P'! Q is true and $\neg Q$ is true, then $\neg P$ will also true. It can be represented as:

Notation for modus tollens : $\frac{P \rightarrow Q, \sim Q}{\sim P}$

Statement-1: "If I am sleepy then I go to bed" $\Rightarrow P \rightarrow Q$

Statement-2: "I do not go to the bed." $\Rightarrow \sim Q$

Statement-3: Which infers that "**I am not sleepy**" $\Rightarrow \sim P$

Proof by Truth table:

P	Q	~P	~Q	P → Q
0	0	1	1	1
0	1	1	0	1
1	0	0	1	0
1	1	0	0	1

3. Hypothetical Syllogism

The Hypothetical Syllogism rule state that if P'!R is true whenever P'!Q is true, and Q'!R is true. It can be represented as the following notation:

Example

Statement-1: If you have my home key then you can unlock my home. $P \rightarrow Q$

Statement-2: If you can unlock my home then you can take my money. $Q \rightarrow R$

Conclusion: If you have my home key then you can take my money. $P \rightarrow R$

Proof by truth table:

P	Q	R	P → Q	Q → R	P → R
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	1	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	0	0
1	1	1	1	1	1

4. Disjunctive Syllogism

The Disjunctive syllogism rule state that if $P \vee Q$ is true, and $\neg P$ is true, then Q will be true. It can be represented as:

Notation of disjunctive syllogism: $\frac{P \vee Q, \neg P}{Q}$

Example

Statement-1: Today is Sunday or Monday. $\Rightarrow P \vee Q$

Statement-2: Today is not Sunday. $\Rightarrow \neg P$

Conclusion: Today is Monday. $\Rightarrow Q$

Proof by truth-table:

P	Q	$\neg$ P	P $\vee$ Q
0	0	1	0
0	1	1	1
1	0	0	1
1	1	0	1

5. Addition

The Addition rule is one the common inference rule, and it states that If P is true, then $P \vee Q$ will be true.

Notation of addition : $\frac{P}{P \vee Q}$

Example:

Statement: I have a vanilla ice-cream. $\Rightarrow P$

Statement-2: I have Chocolate ice-cream.

Conclusion: I have vanilla or chocolate ice-cream. $\Rightarrow (P \vee Q)$

Proof by Truth-Table

P	Q	P $\vee$ Q
0	0	0
1	0	1
0	1	1
1	1	1

6. Simplification

The simplification rule state that if $P \wedge Q$ is true, then Q or P will also be true. It can be represented as:

Notation of simplification rule : $\frac{P \wedge Q}{Q}$ or $\frac{P \wedge Q}{P}$

Proof by Truth-Table:

P	Q	P $\wedge$ Q
0	0	0
1	0	0
0	1	0
1	1	1

7. Resolution

The Resolution rule state that if $P \vee Q$ and $\neg P \wedge R$ is true, then $Q \vee R$ will also be true. It can be represented as

Notation of resolution $\frac{P \vee Q, \neg P \wedge R}{Q \vee R}$

Proof by Truth-Table

P	P	Q	R	$P \vee Q$	$\neg P \wedge R$	$Q \vee R$
0	1	0	0	0	0	0
0	1	0	1	0	0	1
0	1	1	0	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	0	1

2.1.4 Natural Deduction System

Q6. Write about natural deduction system.

Ans :

Natural deduction is a formal inference system which is said naturally to mirror the way in which humans reason. A natural deduction system consists of rules of inference eliminating and introducing each of the connectives and quantifiers of the predicate calculus. The name natural deductive system is given because it mimics the pattern of natural reasoning. It has about 10 deductive inference rules.

Conventions

E for Elimination.

P, Pk , $(1 \leq k \leq n)$ are atoms.

αk, $(1 \leq k \leq n)$ and β are formulae.

Natural Deduction Rules

Rule 1: I-$\wedge$ (Introducing $\wedge$)

I- $\wedge$: If P1, P2, …, Pn then P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn

Interpretation

If we have hypothesized or proved P1, P2, … and Pn , then their conjunction P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn is also proved or derived.

Rule 2: E- $\wedge$ (Eliminating $\wedge$)

E- $\wedge$: If P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn then Pi (1 $\leq$ i $\leq$ n)

Interpretation

If we have proved P1 $\wedge$ P2 $\wedge$ … $\wedge$ Pn , then any Pi is also proved or derived. This rule shows that Ë can be eliminated to yield one of its conjuncts.

Rule 3: I-$\vee$ (Introducing $\vee$)

I-$\vee$: If Pi (1 $\leq$ i $\leq$ n) then P1 $\vee$ P2 $\vee$ … $\vee$ Pn

Interpretation

If any Pi (1 $\leq$ i $\leq$ n) is proved, then P1 $\vee$ … $\vee$ Pn is also proved.

Rule 4: E-$\vee$ (Eliminating $\vee$)

E-$\vee$: If P1 $\vee$ … $\vee$ Pn, P1 $\rightarrow$ P, … , Pn $\rightarrow$ P then P

Interpretation

If P1 V … V Pn, P1 $\rightarrow$ P, … , and Pn $\rightarrow$ P are proved, then P is proved.

Rule 5: I- $\rightarrow$ (Introducing $\rightarrow$)

I-$\rightarrow$ à : If from α1, …, αn infer β is proved then α1 $\wedge$ … $\wedge$ αn $\rightarrow$ β is proved

Interpretation:

If given α1, α2, …andan to be proved and from these we deduce β then α1 $\wedge$ α2 $\wedge$ … $\wedge$ αn $\rightarrow$ β is also proved.

Rule 6: E-$\rightarrow$ (Eliminating $\rightarrow$) - *Modus Ponen*

E-$\rightarrow$: If P1 $\rightarrow$ P, P1 then P

Rule 7: I- $\leftrightarrow$ (Introducing $\leftrightarrow$)

I-$\leftrightarrow$ If P1 $\rightarrow$ P2, P2 $\rightarrow$ P1 then P1 $\rightarrow$ P2

Rule 8: E-$\leftrightarrow$ (Elimination $\leftrightarrow$)

E-$\leftrightarrow$: If P1 $\leftrightarrow$ P2 then P1 $\rightarrow$ P2 , P2 $\rightarrow$ P1

Rule 9: I- ~ (Introducing ~)

I- ~ : If from P infer P1 $\wedge$ ~ P1 is proved then ~P is proved

Rule 10: E- ~ (Eliminating ~)

E- ~ : If from ~ P infer P1 $\wedge$ ~ P1 is proved then P is proved

i) If a formula is derived / proved from a set of premises / hypotheses {α1, …, αn},

" then one can write it as from α1, …, αn infer β.

ii) In natural deductive system,

" a theorem to be proved should have a form from α1, …, αn infer β.

iii) Theorem infer β means that

" there are no premises and β is true under all interpretations i.e., β is a tautology or valid.

iv) If we assume that $\alpha \rightarrow \beta$ is a premise, then we conclude that â is proved ifa is given i.e.,

" if 'from α infer β' is a theorem then $\alpha \rightarrow \beta$ is concluded.

The converse of this is also true.

Deduction Theorem:

To prove a formula ?1 ? ?2 ?… ? ?n ? ?, it is sufficient to prove a theorem **from** ?**1,** ?**2, …,**?**n infer** ?.

Example1: Prove that P $\wedge$ (Q $\vee$ R) follows from P $\wedge$ Q

Solution:

This problem is restated in natural deductive system as "from P $\wedge$ Q infer P $\wedge$ (Q $\vee$ R)". The formal proof is given as follows:

{Theorem} from P $\wedge$ Q infer P $\wedge$ (Q $\vee$ R)

{premise}	P $\wedge$ Q	(1)
{E-$\wedge$, (1)}	P	(2)
{E-$\wedge$, (1)}	Q	(3)
{I-V , (3) }	Q $\vee$ R	(4)
{I-$\wedge$, (2, 4)}	P $\wedge$ (Q $\vee$ R)	Conclusion

Example2: Prove the following theorem:

infer((Q $\rightarrow$ P) $\wedge$ (Q $\rightarrow$ R)) $\rightarrow$ (Q $\rightarrow$ (P $\wedge$ R))

Solution:

In order to prove infer ((Q $\rightarrow$ P) $\wedge$ (Q $\rightarrow$ R)) $\rightarrow$ (Q $\rightarrow$ (P $\wedge$ R)), prove a theorem from {Q à P, Q $\rightarrow$ R} infer Q $\rightarrow$ (P $\wedge$ R).

Further, to prove Q $\rightarrow$ (P $\wedge$ R), prove a sub theorem from Q infer P $\wedge$ R

{Theorem} from Q $\rightarrow$ P, Q $\rightarrow$ R infer

{premise 1}	Q $\rightarrow$ P	(1)
{premise 2}	Q $\rightarrow$ R	(2)
{sub theorem}	from Q infer P $\wedge$ R	(3)
{premise}	Q	(3.1)
{E-$\rightarrow$, (1, 3.1) }	P	(3.2)
{E-$\rightarrow$, (2, 3.1) }	R	(3.3)
{I- $\wedge$, (3.2,3.3) }	P $\wedge$ R	(3.4)
{I-$\rightarrow$, (3)}	Q $\rightarrow$ (P $\wedge$ R)	Conclusion

2.1.5 Axiomatic System

Q7. What is axiomatic system. Explain its rules.

Ans : **(Imp.)**

It is based on the set of only three axioms and one rule of deduction.

It is minimal in structure but as powerful as the truth table and natural deduction approaches.

The proofs of the theorems are often difficult and require a guess in selection of appropriate axiom(s) and rules.

These methods basically require forward chaining strategy where we start with the given hypotheses and prove the goal.

Examples: Establish the following:

Axiom1 (A1): $\alpha \rightarrow (\beta \rightarrow \alpha)$

Axiom2 (A2): $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma)$

Axiom3 (A3): $(\sim\alpha \rightarrow \sim\beta) \rightarrow (\beta \rightarrow \alpha)$

Modus Ponen (MP) defined as follows:

Hypotheses : $\alpha \rightarrow \beta$ and α consequent: β

Examples: Establish the following:

1. {Q}(P → Q) i.e., P → Q is a deductive consequenc of {Q}

{Hypothesis}	Q	(1)
{Axiom A1}	Q → (P → Q)	(2)
{MP, (1, 2)}	P → Q	proved

2. {P → Q, Q → R}. {P → R) i.e., P → R is deductive consequence of {P → Q, Q → R}.

{Hypothesis}	P → Q	(1)
{Hypothesis}	Q → R	(2)
{Axiom A1}	(Q → R) → (P → (Q → R))	(3)
{MP, (2, 3)}	P → (Q → R)	(4)
{Axiom A2}	(P → (Q → R)) → ((P → Q) → (P → R))	(5)
{MP, (4, 5)}	(P → Q) → (P → R)	(6)
{MP, (1, 6)}	P → R	Proved

Deduction Theorems in Axiomatic System

Deduction Theorem:

If Σ is a set of hypothesies and α and β are well-formed formulae, then $\{\Sigma \cup \alpha\} \mid\!- \beta$ implies $\Sigma \mid\!- (\alpha \rightarrow \beta)$

Converse of deduction theorem

Given $\Sigma \mid\!- (\alpha \rightarrow \beta)$,

We can prove $\{\Sigma \cup \alpha\} \mid\!- \beta$

Useful Tips

1. Given α, we can easily prove $\alpha \rightarrow \beta$ for any well-formed formulae a and β.

2. If $\alpha \rightarrow \beta$ is to be proved, then include a in the set of hypotheses Σ and derive β from the set $\{\Sigma u \alpha\}$. Then using deduction theorem, we conclude $\alpha \rightarrow \beta$

Example:

Prove $\sim P \rightarrow (P \rightarrow Q)$ using deduction theorem.

Proof: Prove $\{\sim P\} \vdash (P \rightarrow Q)$ and

$\vdash \sim P \rightarrow (P \rightarrow Q)$ follows from deduction theorem.

2.1.6 Semantic Tableau System in Propositional Logic

Q8. Explain about semantic tableau system in propositional logic.

Ans : **(Imp.)**

The construction of a semantic tableau proceeds as follows: express the premises and negation of the conclusion of an argument in PC using only negation ($\sim$) and disjunction ($\vee$) as propositional connectives. Eliminate every occurrence of two negation signs in a sequence (e.g., $\sim \sim \sim \sim \sim$ a becomes $\sim$a)

Now construct a tree diagram branching downward such that each disjunction is replaced by two branches, one for the left disjunct and one for the right. The original disjunction is true if either branch is true.

Assume a and β be any two formulae

These are the rules for the traditional PL connectives:

Rule 1

A tableau for a formula $(\alpha \wedge \beta)$ is constructed by adding both α and β to the same path (branch). This can be represented as follows:

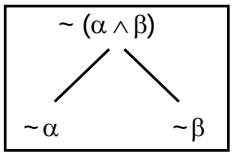
$\alpha \wedge \beta$

α

β

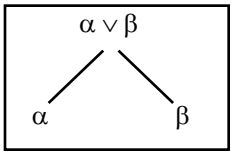
Rule 2:

A tableau for a formula $\sim(\alpha \wedge \beta)$ is constructed by adding two alternative paths one containg $\sim \alpha$ and other containing $\sim \beta$.



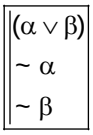
Rule 3

A tableau for a formula $(\alpha \vee \beta)$ is constructed by adding two new paths one containing α and other containing β.



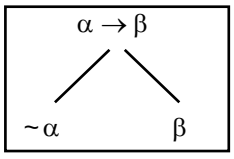
Rule 4

A tableau for a formula $\sim(\alpha \vee \beta)$ is constructed by adding both $\sim \alpha$ and $\sim \beta$ to the same path. This can be expressed as follows:

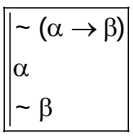


Rule 5

Rule 6

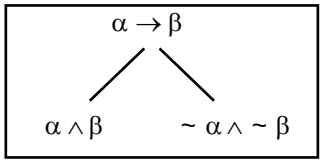


Rule 7



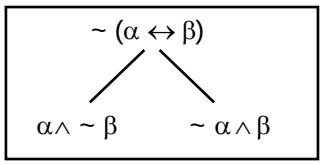
Rule 8

$\alpha \leftrightarrow \beta \cong (\alpha \wedge \beta) \vee (\sim \alpha \wedge \sim \beta)$



Rule 9

$\sim (\alpha \leftrightarrow \beta) \cong (\alpha \wedge \sim \beta) \vee (\sim \alpha \wedge \beta)$



Consistency and Inconsistency

If an atom P and ~ P appear on a same path of a semantic tableau,

i) then inconsistency is indicated and such path is said to be contradictory or closed (finished) path.

ii) Even if one path remains non contradictory or unclosed (open), then the formula ? at the root of a tableau is consistent.

Contradictory tableau (or finished tableau)

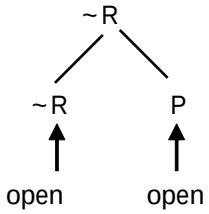
i) It defined to be a tableau in which all the paths are contradictory or closed (finished).

ii) If a tableau for a formula a at the root is a contradictory tableau**,**

iii) then a formula a is said to be inconsistent.

Show that α: $(Q \wedge \sim R) \wedge (R \rightarrow P)$ is consistent and find its model.

{Tableau root}	$(Q \wedge \sim R) \wedge (R \rightarrow P)$	(1)
{Apply rule 1 to 1}	$(Q \wedge \sim R)$	(2)
	$(R \rightarrow P)$	(3)
{Apply rule 1 to 2}	Q	
{Apply rule 6 to 3}		



{Q = T, R = F} amd{P = T, Q = T, R = F} are models of α.

Show that α:

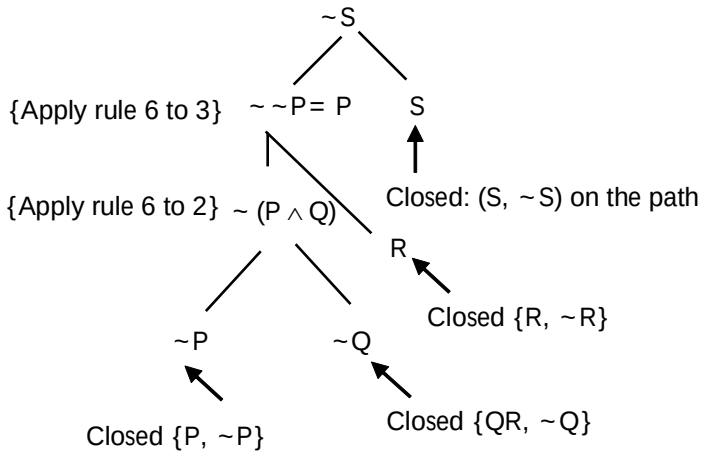
$(P \wedge Q \rightarrow R) \wedge (\sim P \rightarrow S) \wedge Q \wedge \sim R \wedge \sim S$ is inconsistent using tableaux method.

{Applyrule 1 to 1} $P \wedge Q \rightarrow R$ (1)

$\sim P \rightarrow S$ (2)

Q (3)

$\sim R$



α is inconsistent as we get contradictory tableau.

2.1.7 Resolution Refutation In Propositional Logic

Q9. Explain about resolution refutation in propositional logic.

Ans : **(Imp.)**

Resolution is a theorem proving technique that proceeds by building refutation proofs, i.e., proofs by contradictions. Resolution is used, if there are various statements are given, and we need to prove a conclusion of those statements. Unification is a key concept in proofs by resolutions. Resolution is a single inference rule which can efficiently operate on the conjunctive normal form or clausal form.

Clause

Disjunction of literals (an atomic sentence) is called a **clause**. It is also known as a unit clause.

Conjunctive Normal Form

A sentence represented as a conjunction of clauses is said to be conjunctive normal form or CNF.

The resolution inference rule:

The resolution rule for first-order logic is simply a lifted version of the propositional rule. Resolution can resolve two clauses if they contain complementary literals, which are assumed to be standardized apart so that they share no variables.

$$\frac{1_1 V \ldots V 1_k,\ m1 V \ldots . V m_n}{Subst(\theta, 1_1 V \ldots V 1_{i-1} V 1_{i+1} V \ldots . V\ l_k V m_1 V \ldots V m_{j-1} V m_{j+1} V \ldots V m_n)}$$

Where l_i and m_j are complementary literals.

This rule is also called the binary resolution rule because it only resolves exactly two literals.

Example

We can resolve two clauses which are given below:

[Animal (g(x) $\vee$ Loves (f(x), x)] and [$\neg$ Loves(a, b) $\vee \neg$ Kills(a, b)]

Where two complimentary literals are: Loves (f(x), x) and $\neg$ Loves (a, b)

These literals can be unified with unifier θ= [a/f(x), and b/x] , and it will generate a resolvent clause:

[Animal (g(x) $\vee \neg$ Kills(f(x), x)].

Steps for Resolution

1. Conversion of facts into first-order logic.

2. Convert FOL statements into CNF

3. Negate the statement which needs to prove (proof by contradiction)

4. Draw resolution graph (unification).

To better understand all the above steps, we will take an example in which we will apply resolution.

Example

(a) John likes all kind of food.

(b) Apple and vegetable are food

(c) Anything anyone eats and not killed is food.

(d) Anil eats peanuts and still alive

(e) Harry eats everything that Anil eats.

Prove by resolution that:

(f) John likes peanuts.

Step-1: Conversion of Facts into FOL

In the first step we will convert all the given statements into its first order logic.

(a) $\forall x : food(x) \rightarrow likes(John, x)$

(b) $food(Apple) \wedge food(vegetables)$

(c) $\forall x \forall y : eats(x, y) \wedge \neg killed(x) \rightarrow food(y)$

(d) eats (Anil, Peanuts) $\wedge$ alive(Anil)

(e) $\forall x$: eats (Anil, x) $\rightarrow$ eats(Harry, x)

(f) $\forall x : \neg$ killed(x) $\rightarrow$ alive(x)

(g) $\forall x$: alive(x) $\rightarrow$ killed(x) added predicates

(h) likes(John, Peanuts)

Step-2: Conversion of FOL into CNF

In First order logic resolution, it is required to convert the FOL into CNF as CNF form makes easier for resolution proofs.

Distribute conjunction $\wedge$ over disjunction $\neg$

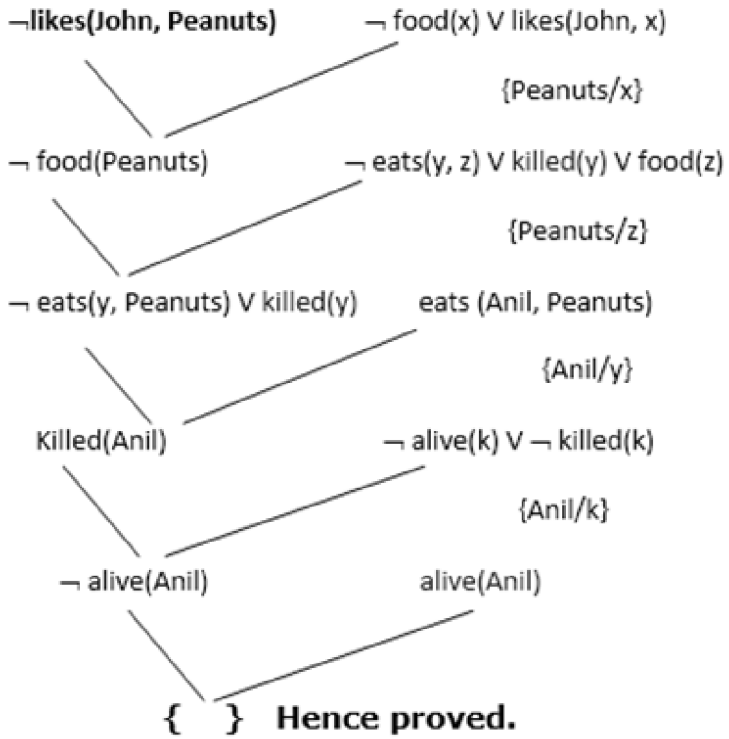
This step will not make any change in this problem.

Step-3: Negate the statement to be proved

In this statement, we will apply negation to the conclusion statements, which will be written as ¬likes(John, Peanuts)

Step-4: Draw Resolution graph:

Now in this step, we will solve the problem by resolution tree using substitution. For the above problem, it will be given as follows:



Hence the negation of the conclusion has been proved as a complete contradiction with the given set of statements.

Explanation of Resolution Graph

1. In the first step of resolution graph, ¬likes(John, Peanuts), and likes(John, x) get resolved(canceled) by substitution of {Peanuts/x}, and we are left with ¬ food(Peanuts)

2. In the second step of the resolution graph, ¬ food (Peanuts), and food(z) get resolved (canceled) by substitution of { Peanuts/z}, and we are left with ¬ eats (y, Peanuts) V killed(y).

3. In the third step of the resolution graph, ¬ eats (y, Peanuts) and eats (Anil, Peanuts) get resolved by substitution {Anil/y}, and we are left with Killed (Anil).

4. In the fourth step of the resolution graph, Killed(Anil) and ¬ killed(k) get resolve by substitution {Anil/k}, and we are left with ¬ alive(Anil) .

5. In the last step of the resolution graph ¬ alive(Anil) and alive(Anil) get resolved.

2.1.8 Predicate Logic

Q10. Define predicate logic. Explain components and characteristics of predicate logic.

Ans :

Meaning

Predicate Logic or First-Order Logic (FOL) is used to represent complex expressions in easier forms using predicates, variables, and quantifiers. The real-world facts can be simply represented as local propositions written as well-formed formulas in propositional logic.

Components of Predicate Logic

The three components of Predicate logic are:

1. Predicates

The symbols which are used to represent the properties or relationships between real-world objects are called Predicates. They play an important role in First-Order Logic for knowledge representation.

For example, predicates like isLocatedIn(Delhi, India) are used to represent the information.

2. Variables

The symbols used to represent the objects or entities are called Variables. They are used to quantify the objects by providing them with a symbol and passing it to predicates.

For example, in the above example is Located In(Delhi, India), the variables are Delhi and India.

3. Quantifiers

The symbols in logical statements that are used to represent the scope of variables are called Quantifiers. They are used to represent the relationships between objects and properties.

The two main quantifiers are Universal Quantifier (“) and Existential Quantifier (“).

Characteristics

The following are the characteristics of Predicate Logic in AI:

1. The Predicate Logic makes the representation of complex relationships simpler by representing them using the symbols like Predicates and Variables.

2. Predicate Logic is used to represent many complex relationships. It can represent negation, conjunction, disjunction, and many more types of statements.

3. Predicate Logic consists of well-defined rules and proper syntax to represent the relationships via valid logical expressions.

4. Predicate Logic is used widely in the field of Artificial Intelligence because of its capability to represent facts and relationships in a structured manner.

Before moving towards the example of Predicate Logic, let’s first understand the Connectives and Quantifiers in Predicate Logic.

Q11. Write about logical connectives and Quantifiers in predicate logic.

Ans :

Meaning

Logical Connectives are symbols that are used to represent more than one statement by combining them to form a complex logical statement. These are used to understand the relationship between the propositions.

The following are the most used logical connectives:

1. Negation (¬)

Negation is used to negate a statement, which means it reverses the truth value of the expression by changing true to false and vice versa.

For example, if an expression p is true, the negation is represented by ¬p.

2. Conjunction ($\wedge$)

The conjunction is used to combine two statements, and its value is true when both the expression are true. It is basically the representation of logical AND.

For example, if both p and q are true, the value of p $\wedge$ q is also true.

3. Disjunction ($\vee$)

The disjunction is also used to connect two statements, but its value is true when either of the expression is true. It is basically the representation of logical OR.

For example, if either p or q is true, the value of $p \vee q$ is also true.

4. Implication ($\rightarrow$)

The implication is also used to connect two statements, but in a way, if one statement, i.e., antecedent, is true, then the other statement, consequent, must have to be true.

For example, $p \rightarrow q$ is false only if p is true and q is false.

5. Biconditional ($\leftrightarrow$)

The Biconditional is the logical representation of iff(if and only if) and returns the expression as true if both have the same truth value.

For example, $p \leftrightarrow q$ is true if both p and q are either true or false.

The following is the truth table of the above-discussed Logical Connectives.

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
True	True	True	True	True	True
True	False	False	True	False	False
False	True	False	True	True	False
False	False	False	False	True	True

Quantifiers in Predicate Logic

Quantifiers, as we read above, are the symbols in logical statements that are used to represent the scope of variables. They are used to represent the relationships between objects and properties.

The two main quantifiers are:

1. Universal Quantifier ($\forall$)

The symbol $\forall$ is used to represent the Universal Quantifier, which basically means "For All". It signifies the expression is true for every object or entity.

For example, "All Boys Like Football" can be written as $\forall x : Boys(x) \rightarrow Like(x, Football)$.

2. Existential Quantifier ($\exists$)

The symbol $\exists$ is used to represent the Existential Quantifier, which basically means "There exists". It signifies the expression is true for at least one object or entity.

For example, "Some Boys Like Football" can be written as $\exists x : Boys(x) \wedge Like(x, Football)$.

Example of Predicate Logic

Let's see a simple example to understand Predicate Logic in a much better way. Suppose we are given the following statement.

Statement

All white birds are beautiful.

Let's define the predicates and write the expression for this statement.

Predicates

i) IsWhite(x): Represents the property that x is white.

ii) IsBeautiful(x): Represents the property that x is beautiful.

The following will be the expression for the same.

Expression

$\forall$ x (IsWhite(x) $\rightarrow$ IsBeautiful(x))

Explanation

The statement $\forall$ x (IsWhite(x) $\rightarrow$ Is Beautiful (x)) can be read as "For all x, if x is white, then x is beautiful". The universal quantifier ($\forall$) indicates that the statement applies to all objects (birds) in the domain. The implication ($\rightarrow$) connects the properties "IsWhite(x)" and "IsBeautiful(x)," stating that if an object x is white, then it is also beautiful.

2.1.9 Logic Programming

Q12. What is logic programming ? Explain about it.

Ans :

Logic Programming is a programming paradigm in which the problems are expressed as facts and rules by program statements but within a system of formal logic. Just like other programming paradigms like object oriented, functional, declarative, and procedural, etc., it is also a particular way to approach programming.

Inductive logic programming is worried about summing up negative and positive models with regards to foundation information: ML of logic programs.

Abductive logic programming is an undeniable level information portrayal structure that can be utilized to tackle issues definitively dependent on abductive thinking.

It expands ordinary logic programming by permitting a few predicates to be deficiently characterized, announced as inducible predicates.

Programmable Logic Array is a gate followed by a programmable and fixed architecture logic gadget with programmable or gates.

Logic Programming uses facts and rules for solving the problem. That is why they are called the building blocks of Logic Programming. A goal needs to be specified for every program in logic programming. To understand how a problem can be solved in logic programming, we need to know about the building blocks " Facts and Rules "

Facts

Actually, every logic program needs facts to work with so that it can achieve the given goal. Facts basically are true statements about the program and data. For example, Delhi is the capital of India.

Rules

Actually, rules are the constraints which allow us to make conclusions about the problem domain. Rules basically written as logical clauses to express various facts. For example, if we are building any game then all the rules must be defined.

Rules are very important to solve any problem in Logic Programming. Rules are basically logical conclusion which can express the facts. Following is the syntax of rule

A: B1, B2,...,B_n.

Here, A is the head and B1, B2, ...Bn is the body.

For example

ancestor(X,Y) :- father(X,Y).

ancestor(X,Z) :- father(X,Y), ancestor(Y,Z).

This can be read as, for every X and Y, if X is the father of Y and Y is an ancestor of Z, X is the ancestor of Z. For every X and Y, X is the ancestor of Z, if X is the father of Y and Y is an ancestor of Z.

Benefits of using logic programming in AI applications?

Logic programming is a powerful tool for AI applications. It allows for the concise representation of knowledge and the efficient execution of inference. Logic programming has been used in a wide range of AI applications, including natural language processing, knowledge representation and reasoning, planning, and machine learning.

Logic programming has several advantages over other AI paradigms. First, logic programs are declarative, meaning that they specify what is to be done, rather than how it is to be done. This makes them easier to understand and maintain than procedural programs. Second, logic programs can be executed efficiently by computers. Third, logic programs can be easily extended and modified.

Fourth, logic programming is a well-understood paradigm with a rich theoretical foundation. This foundation can be used to develop new AI applications and to understand and improve existing ones. Finally, logic programming is well suited for use in distributed systems, such as the World Wide Web.

2.2 KNOWLEDGE REPRESENTATION

2.2.1 Introduction

Q13. What is Knowledge Representation? Explain the different types of knowledge.

Ans : **(Imp.)**

Meaning

Knowledge Representation in AI describes the representation of knowledge. Basically, it is a study of how the beliefs, intentions, and judgments of an intelligent agent can be expressed suitably for automated reasoning. One of the primary purposes of Knowledge Representation includes modeling intelligent behavior for an agent.

Knowledge Representation and Reasoning (KR, KRR) represents information from the real world for a computer to understand and then utilize this knowledge to solve complex real-life problems like communicating with human beings in natural language. Knowledge representation in AI is not just about storing data in a database, it allows a machine to learn from that knowledge and behave intelligently like a human being.

The different kinds of knowledge that need to be represented in AI include:

i) Objects

ii) Events

iii) Performance

iv) Facts

v) Meta-Knowledge

vi) Knowledge-base

Now that you know about Knowledge representation in AI, let's move on and know about the different types of Knowledge.

Types of Knowledge

There are 5 types of Knowledge such as:

1. Declarative Knowledge

It includes concepts, facts, and objects and expressed in a declarative sentence.

2. Structural Knowledge

It is a basic problem-solving knowledge that describes the relationship between concepts and objects.

3. Procedural Knowledge

This is responsible for knowing how to do something and includes rules, strategies, procedures, etc.

4. Meta Knowledge

Meta Knowledge defines knowledge about other types of Knowledge.

5. Heuristic Knowledge

This represents some expert knowledge in the field or subject.

These are the important types of Knowledge Representation in AI. Now, let's have a look at the cycle of knowledge representation and how it works.

Q14. Cycle of Knowledge Representation in AI.

Ans :

Artificial Intelligent Systems usually consist of various components to display their intelligent behavior. Some of these components include:

i) Perception

ii) Learning

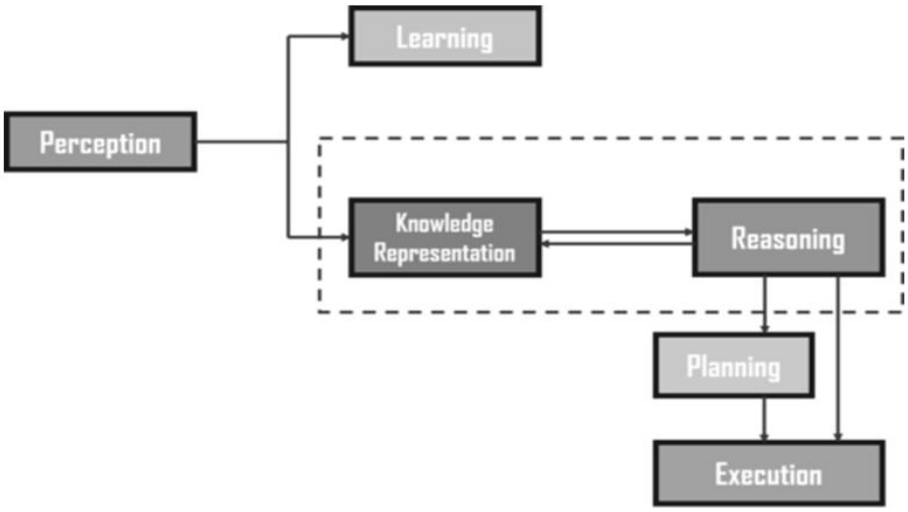
iii) Knowledge Representation & Reasoning

iv) Planning

v) Execution

Here is an example to show the different components of the system and how it works:

Example



The above diagram shows the interaction of an AI system with the real world and the components involved in showing intelligence.

1. The Perception component retrieves data or information from the environment. with the help of this component, you can retrieve data from the environment, find out the source of noises and check if the AI was damaged by anything. Also, it defines how to respond when any sense has been detected.
2. Then, there is the Learning Component that learns from the captured data by the perception component. The goal is to build computers that can be taught instead of programming them. Learning focuses on the process of self-improvement. In order to learn new things, the system requires knowledge acquisition, inference, acquisition of heuristics, faster searches, etc.
3. The main component in the cycle is Knowledge Representation and Reasoning which shows the human-like intelligence in the machines. Knowledge representation is all about understanding intelligence. Instead of trying to understand or build brains from the bottom up, its goal is to understand and build intelligent behavior from the top-down and focus on what an agent needs to know in order to behave intelligently. Also, it defines how automated reasoning procedures can make this knowledge available as needed.
4. The Planning and Execution components depend on the analysis of knowledge representation and reasoning. Here, planning includes giving an initial state, finding their preconditions and effects, and a sequence of actions to achieve a state in which a particular goal holds. Now once the planning is completed, the final stage is the execution of the entire process.

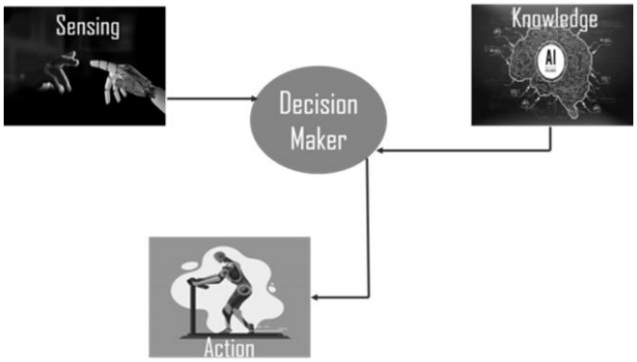
So, these are the different components of the cycle of Knowledge Representation in AI. Now, let's understand the relationship between knowledge and intelligence.

Q15. What is the Relation between Knowledge & Intelligence?

Ans :

In the real world, knowledge plays a vital role in intelligence as well as creating artificial intelligence. It demonstrates the intelligent behavior in AI agents or systems. It is possible for an agent or system to act accurately on some input only when it has the knowledge or experience about the input.

Let's take an example to understand the relationship:



In this example, there is one decision-maker whose actions are justified by sensing the environment and using knowledge. But, if we remove the knowledge part here, it will not be able to display any intelligent behavior.

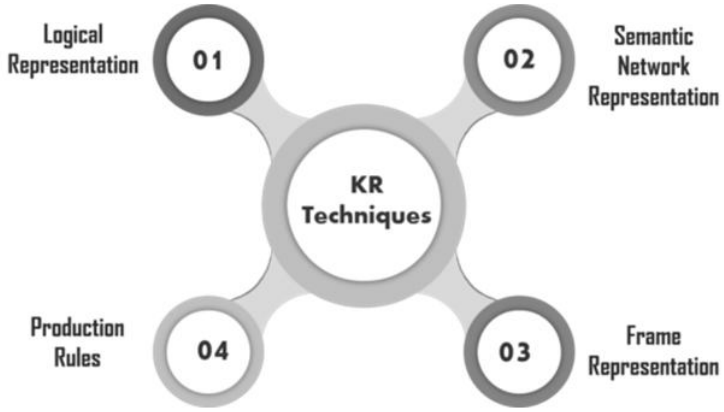
Now that you know the relationship between knowledge and intelligence, let's move on to the techniques of Knowledge Representation in AI.

Q16. Explain various techniques of knowledge representation.

Ans : **(Imp.)**

Techniques of Knowledge Representation in AI

There are four techniques of representing knowledge such as:



Now, let's discuss these techniques in detail.

1. Logical Representation

Logical representation is a language with some definite rules which deal with propositions and has no ambiguity in representation. It represents a conclusion based on various conditions and lays down some important communication rules. Also, it consists of precisely defined syntax and semantics which supports the sound inference. Each sentence can be translated into logics using syntax and semantics.

S.No.	Syntax	S.No.	Semantics
1.	It decides how we can construct legal sentences in logic.	1.	Semantics are the rules by which we can interpret the sentence in the logic.
2.	It determines which symbol we can use in knowledge representation.	2.	It assigns a meaning to each sentence.
3.	Also, how to write those symbols.		

Advantages

1. Logical representation helps to perform logical reasoning.
2. This representation is the basis for the programming languages.

Disadvantages

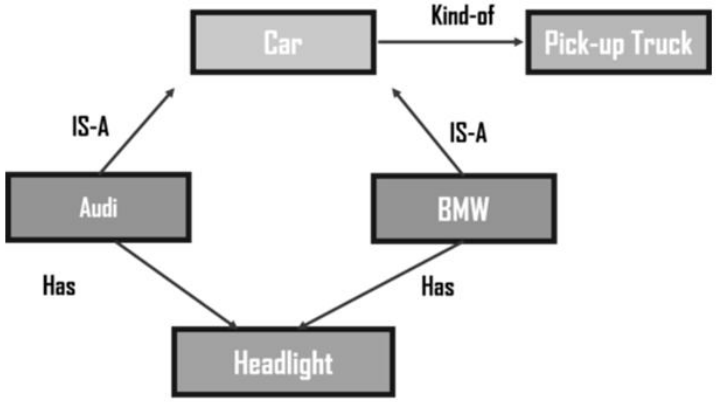
1. Logical representations have some restrictions and are challenging to work with.
2. This technique may not be very natural, and inference may not be very efficient.

2. Semantic Network Representation

Semantic networks work as an alternative of predicate logic for knowledge representation. In Semantic networks, you can represent your knowledge in the form of graphical networks. This network consists of nodes representing objects and arcs which describe the relationship between those objects. Also, it categorizes the object in different forms and links those objects.

This representation consist of two types of relations:

1. IS-A relation (Inheritance)
2. Kind-of-relation



Advantages

1. Semantic networks are a natural representation of knowledge.
2. Also, it conveys meaning in a transparent manner.
3. These networks are simple and easy to understand.

Disadvantages

1. Semantic networks take more computational time at runtime.
2. Also, these are inadequate as they do not have any equivalent quantifiers.
3. These networks are not intelligent and depend on the creator of the system..

3. Frame Representation

A frame is a record like structure that consists of a collection of attributes and values to describe an entity in the world. These are the AI data structure that divides knowledge into substructures by representing stereotypes situations. Basically, it consists of a collection of slots and slot values of any type and size. Slots have names and values which are called facets.

Advantages

1. It makes the programming easier by grouping the related data.
2. Frame representation is easy to understand and visualize.
3. It is very easy to add slots for new attributes and relations.
4. Also, it is easy to include default data and search for missing values.

Disadvantages

1. In frame system inference, the mechanism cannot be easily processed.
2. The inference mechanism cannot be smoothly proceeded by frame representation.
3. It has a very generalized approach.

Production Rules

In production rules, agent checks for the condition and if the condition exists then production rule fires and corresponding action is carried out. The condition part of the rule determines which rule may be applied to a problem. Whereas, the action part carries out the associated problem-solving steps. This complete process is called a recognize-act cycle.

The production rules system consists of three main parts:

1. The set of production rules
2. Working Memory
3. The recognize-act-cycle

Advantages

1. The production rules are expressed in natural language.
2. The production rules are highly modular and can be easily removed or modified.

Disadvantages

1. It does not exhibit any learning capabilities and does not store the result of the problem for future uses.
2. During the execution of the program, many rules may be active. Thus, rule-based production systems are inefficient.

So, these were the important techniques for Knowledge Representation in AI. Now, let's have a look at the requirements for these representations.

Representation Requirements

A good knowledge representation system must have properties such as:

1. **Representational Accuracy**

   It should represent all kinds of required knowledge.

2. **Inferential Adequacy**

   It should be able to manipulate the representational structures to produce new knowledge corresponding to the existing structure.

3. **Inferential Efficiency**

   The ability to direct the inferential knowledge mechanism into the most productive directions by storing appropriate guides.

4. **Acquisitional Efficiency**

   The ability to acquire new knowledge easily using automatic methods.

Now, let's have a look at some of the approaches to Knowledge Representation in AI along with different examples.

2.2.2 Approaches to knowledge Representation

Q17. Write about various Approaches to Knowledge Representation in AI.

Ans : **(Imp.)**

There are different approaches to knowledge representation such as:

1. **Simple Relational Knowledge**

It is the simplest way of storing facts which uses the relational method. Here, all the facts about a set of the object are set out systematically in columns. Also, this approach of knowledge representation is famous in database systems where the relationship between different entities is represented. Thus, there is little opportunity for inference.

Example:

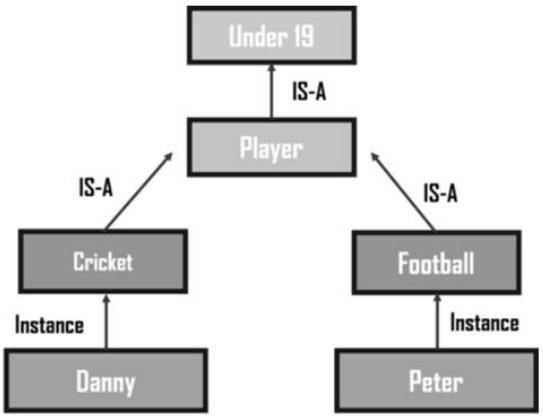
Name	Age	Emp ID
John	25	100071
Amanda	23	100056
Sam	27	100042

This is an example of representing simple relational knowledge.

2. Inheritable Knowledge

In the inheritable knowledge approach, all data must be stored into a hierarchy of classes and should be arranged in a generalized form or a hierarchal manner. Also, this approach contains inheritable knowledge which shows a relation between instance and class, and it is called instance relation. In this approach, objects and values are represented in Boxed nodes.

Example



3. Inferential Knowledge

The inferential knowledge approach represents knowledge in the form of formal logic. Thus, it can be used to derive more facts. Also, it guarantees correctness.

Example

Statement 1: John is a cricketer.

Statement 2: All cricketers are athletes.

Then it can be represented as;

Cricketer(John)

$\forall$ x = Cricketer (x) $\rightarrow$ Athelete (x)s

These were some of the approaches to knowledge representation in AI along with examples. With this, we have come to the end of our article. I hope you understood what is Knowledge Representation in AI and its different types.

2.2.3 knowledge Representation using Semantic Network

Q18. How to represent knowledge with semantic network? Explain

Ans :

A semantic network is a graphical representation of knowledge, where nodes represent concepts or objects, and links represent relationships between them. The syntax of a semantic network consists of nodes and links, and the semantics involve defining the meaning of each node and link.

One of the main advantages of semantic networks is that they can be easily visualized, making them more intuitive to understand than logical representations. Additionally, they can categorize objects and link them together.

However, there are some drawbacks associated with this representation method. For instance, semantic networks can be computationally expensive at runtime, as traversing the entire network tree may be necessary to answer certain questions. Furthermore, modeling the vastness of human-like memory is not practical. Semantic networks also lack quantifier equivalents such as "for all" or "for some", and do not have standard definitions for link names. Additionally, they are not inherently intelligent and depend on the creator of the system.

Example

The following are a few statements that must be represented with nodes and arcs:

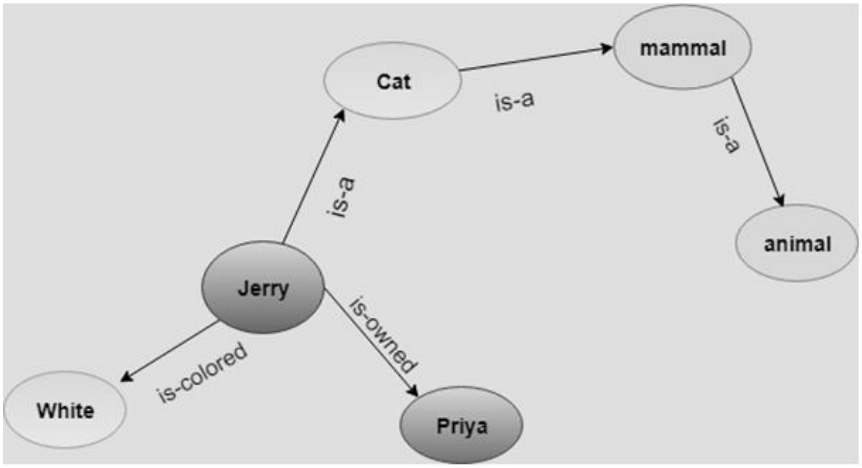
i) Jerry is a cat.

ii) Jerry is a mammal

iii) Jerry is owned by Priya.

iv) Jerry is brown-colored.

v) All Mammals are animals.



In the above diagram, we have represented the different type of knowledge in the form of nodes and arcs. Each object is connected with another object by some relation.

Drawbacks in Semantic Representation

1. Semantic networks take more computational time at runtime as we need to traverse the complete network tree to answer some questions. It might be possible in the worst case scenario that after traversing the entire tree, we find that the solution does not exist in this network.

2. Semantic networks try to model human-like memory (Which has 1015 neurons and links) to store the information, but in practice, it is not possible to build such a vast semantic network.

3. These types of representations are inadequate as they do not have any equivalent quantifier, e.g., for all, for some, none, etc.

4. Semantic networks do not have any standard definition for the link names.

5. These networks are not intelligent and depend on the creator of the system.

Advantages of Semantic Network

1. Semantic networks are a natural representation of knowledge.

2. Semantic networks convey meaning in a transparent manner.

3. These networks are simple and easily understandable.

Example

Every human, animal and bird is living thing who breathe and eat.

$\forall$ X [human(X) $\rightarrow$ living(X)]

$\forall$ X [animal(X) $\rightarrow$ living(X)]

$\forall$ X [bird(X) $\rightarrow$ living(X)]

All birds are animal and can fly.

$\forall$ X [bird(X) $\wedge$ canfly(X)]

Every man and woman are humans who have two legs.

$\forall$ X [man(X) $\wedge$ haslegs(X)]

$\forall$ X [woman(X) $\wedge$ haslegs(X)]

$\forall$ X [human(X) $\wedge$ has(X, legs)]

Cat is an animal and has a fur. animal(cat) Ù has(cat, fur)

All animals have skin and can move.

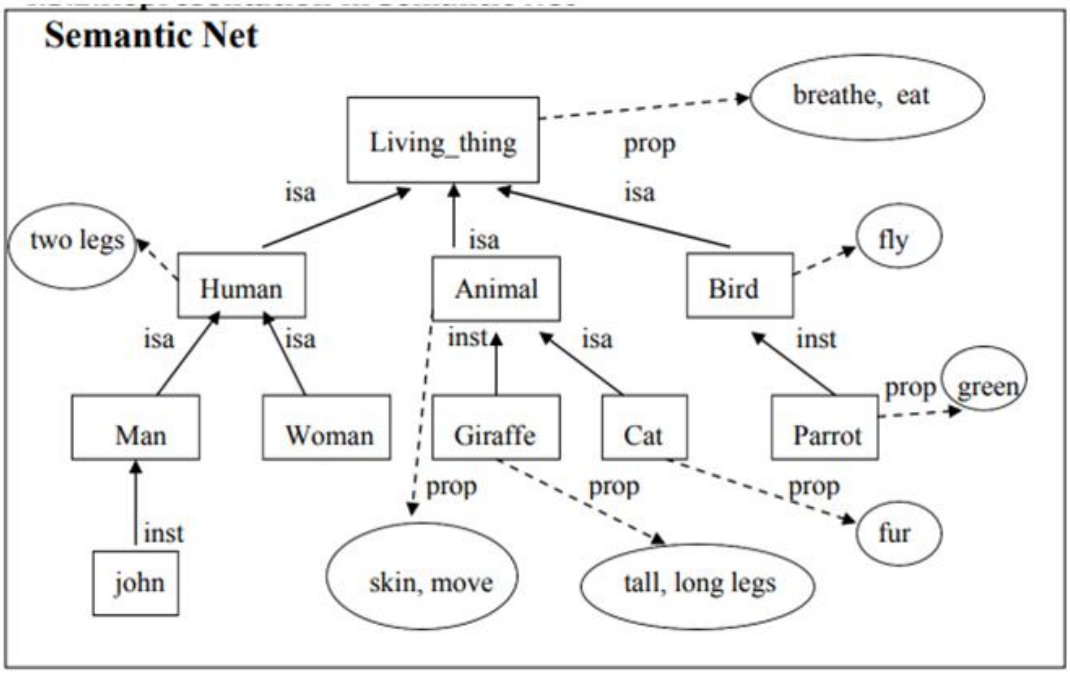
$\forall$ X [animal(X) $\rightarrow$ has(X, skin) $\wedge$ canmove(X)]

Giraffe is an animal who is tall and has long legs.

animal(giraffe) Ù has(giraffe, long_legs) $\wedge$ is(giraffe, tall)

Parrot is a bird and is green in color.

bird(parrot) $\wedge$ has(parrot, green_colour)



2.2.4 Knowledge Representation Using Frames

Q19. How knowledge can be represented using Frames? Explain

Ans :

Frames are more structured form of packaging knowledge,

– used for representing objects, concepts etc.

1. Frames are organized into hierarchies or network of frames.

2. Lower level frames can inherit information from upper level frames in network.

3. Nodes are connected using links viz.,

i) ako / subc (links two class frames, one of which is subclass of other e.g., science_faculty class is ako of faculty class),

ii) is_a / inst(connects a particular instance of a class frame e.g., Renukais_ascience_faculty)

iii) a_part_of (connects two class frames one of which is contained in other e.g., faculty class is_part_of department class).

Property link of semantic net is replaced by SLOT Fields

1. A frame may have any number of slots needed for describing object. e.g.,

– faculty frame may have name, age, address, qualification etc as slot names.

2. Each frame includes two basic elements : slots and facets.

Each slot may contain one or more facets (called fillers) which may take many forms such as:

i) value (value of the slot),

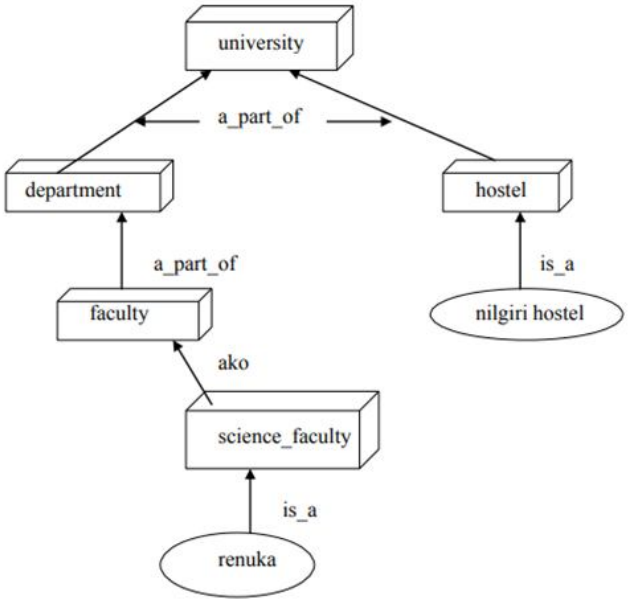
ii) default (default value of the slot),

iii) range (indicates the range of integer or enumerated values, a slot can have),

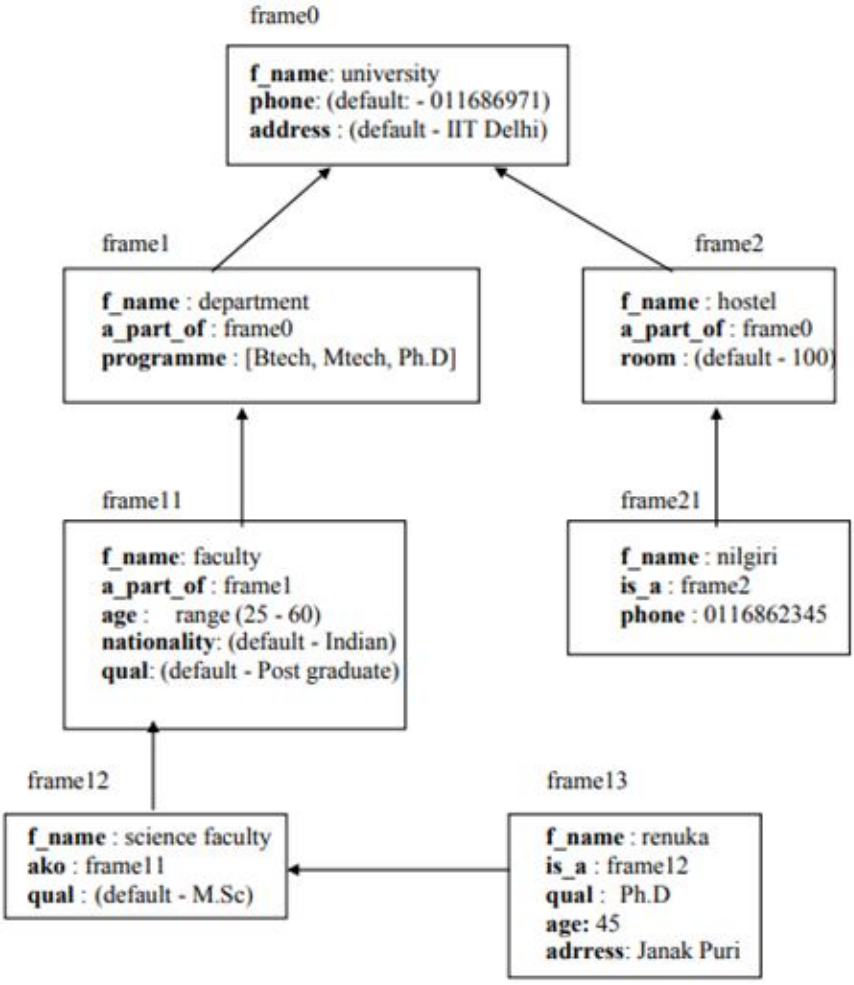
iv) demons (procedural attachments such as if_needed, if_deleted, if_added etc.) and

v) other (may contain rules, other frames, semantic net or any type of other information).

Frame Network – Example



Detailed Representation of Frame Network



Description of Frames

1. Each frame represents either a class or an instance.
2. Class frame represents a general concept whereas instance frame represents a specific occurrence of the class instance.
3. Class frame generally have default values which can be redefined at lower levels.
4. If class frame has actual value facet then decedent frames can not modify that value.
5. Value remains unchanged for subclasses and instances.

Inheritance in Frames

1. Suppose we want to know nationality or phone of an instance-frame frame13 of renuka.
2. Theseinformations are not given in this frame.
3. Search will start from frame13 in upward direction till we get our answer or have reached root frame.
4. The frames can be easily represented in prolog by choosing predicate name as frame with two arguments.
5. First argument is the name of the frame and second argument is a list of slot - facet pair.

UNIT III

Expert System and Applications: Introduction, Phases in Building Expert Systems, Expert System Architecture, Expert Systems vs Traditional Systems, Truth Maintenance Systems, Application of Expert Systems, List of Shells and Tools. Uncertainty Measure-Probability Theory: Introduction, Probability Theory, Bayesian Belief Networks, Certainty Factor Theory, Dempster-Shafer Theory.

3.1 EXPERT SYSTEM AND APPLICATIONS

3.1.1 Introduction

Q1. Explain briefly about Expert System?

Ans :

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries.

The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base.

The system helps in decision making for compsex problems using both facts and heuristics like a human expert. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as medicine, science, etc.

The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box.

3.1.2 Phases In Building Expert Systems

Q2. Explain various steps to develop an Expert system.

Ans : **(Imp.)**

Steps to Develop an Expert System

An expert system (ES) was developed and refined over several years. Expert System Life Cycle has five stages:

Step1: Identification: Determining the characteristics of the problem.

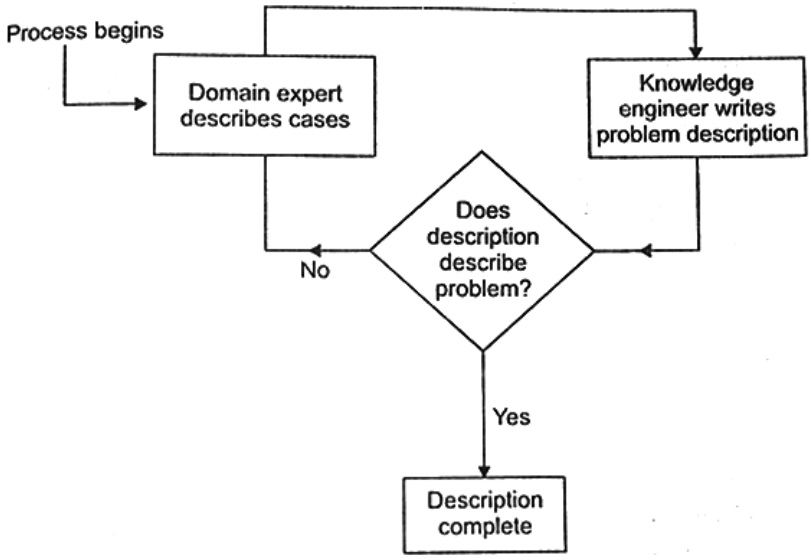


Fig.: Iterative process of identifying the problem in expert system

Step2: Conceptualization: Finding the concept to produce the solution.

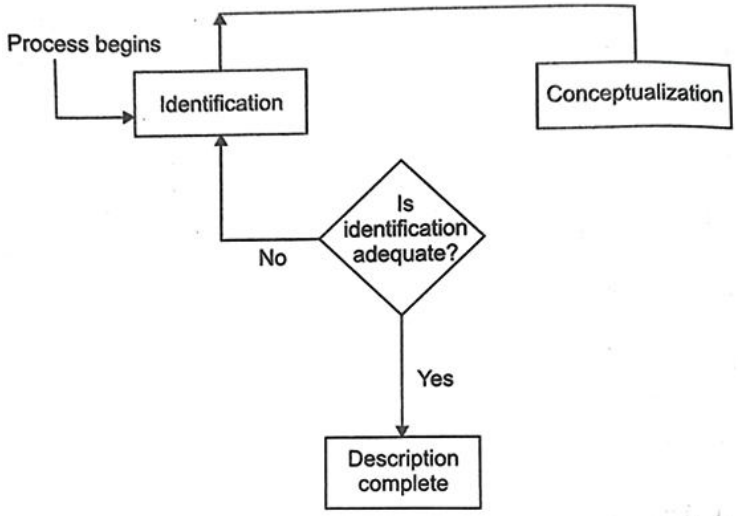


Fig.: Conceptual phase of expert system development life cycle

Step3: Formalization: Designing structures to organize the knowledge.

Step4: Implementation: Formulating rules which embody the knowledge.

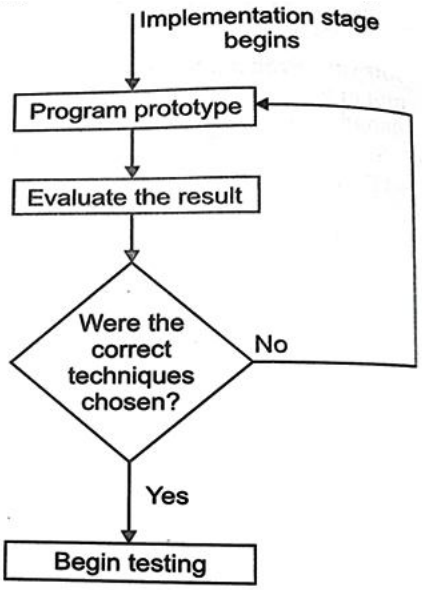


Fig.: Implementation phase of expert system development life cycle

Step5: Testing: Validating the rules.

The chances of a prototype expert system executing flawlessly are very less initially. A knowledge engineer doesn't expect the testing process to verify that the system has been constructed entirely correctly. Testing provides an opportunity to identify the weaknesses in the structure and the implementation of the system. Testing includes are:

i) The system implements correctly or incorrectly.

ii) Rules implement correctly or not.

iii) The System uses for testing for both simple and complex problems by domain experts to uncover more defects.

iv) An Expert System is finally tested to be successful only when it is operated at the level of a human expert.

v) The testing process is NOT complete until it indicates that the solutions suggested by the expert system are consistently valid.

vi) Expert systems are typically interactive, they work in question-and-answer form. This interaction between users and the Experts system continues until the system can conclude.

3.1.3 Expert System Architecture

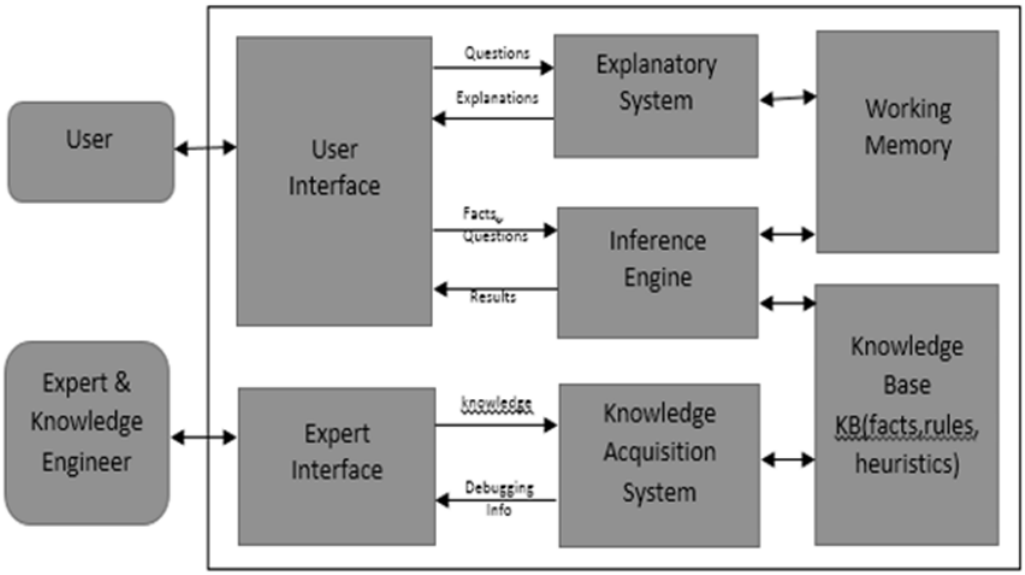
Q3. Explain the expert of architecture system.

Ans :

In artificial intelligence, an expert system is a computer system that emulates the decision-making ability of a human expert.[1] Expert systems are designed to solve complex problems by reasoning about knowledge, represented mainly as if–then rules rather than through conventional procedural code. Expert systems were among the first truly successful forms of artificial intelligence (AI) software.

An expert system is divided into two subsystems: the inference engine and the knowledge base. The knowledge base represents facts and rules. The inference engine applies the rules to the known facts to deduce new facts. Inference engines can also include explanation and debugging abilities.

Architecture of Expert System



The architecture of an Expert System (ES) consists of the following major components:

1. Knowledge Base (KB)

Repository of special heuristics or rules that direct the use of knowledge, facts (productions). It contains the knowledge necessary for understanding, formulating, & problem solving.

2. Working Memory(Blackboard): if forward chaining used

It describes the current problem & record intermediate results

Records Intermediate Hypothesis & Decisions: 1. Plan, 2. Agenda, 3. Solution

3. Inference Engine

The deduction system used to infer results from user input & KB

It is the brain of the ES, the control structure(rule interpreter)

It provides methodology for reasoning

4. Explanation Subsystem (Justifier)

Traces responsibility & explains the ES behaviour by interactively answering question: Why?, How?, What?, Where?, When?, Who?

5. User Interface

Interfaces with user through Natural Language Processing (NLP), or menus & graphics. Acts as Language Processor for friendly, problem-oriented communication

Shell = Inference Engine + User Interface

The Human Elements in ESs

Expert: Has the special knowledge, judgement, experience and methods to give advice and solve problems. Provides knowledge about task performance

Knowledge Engineer: Usually also the System Builder

Helps the expert(s) structure the problem area by interpreting and integrating human answers to questions, drawing analogies, posing counter examples, and bringing to light conceptual difficulties.

The Expert & the knowledge Engineer should Anticipate Users' needs & Limitations when designing Expert Systems

User: Possible Classes of Users can be

1. A non-expert client seeking direct advice (ES acts as a Consultant or Advisor)

2. A student who wants to learn (ES acts as an Instructor)

3. An ES builder improving or increasing the knowledge base(ES acts as a Partner)

4. An Expert (ES acts as a Colleague or an Assistant)

3.1.4 Expert Systems vs Traditional Systems

Q4. Differentiate between conventional systems and expert systems.

(OR)

Distinguish between traditional system and expert system.

Ans : **(Imp.)**

Table : Difference between conventional system and expert system

S.No.	Conventional System	Expert System
1.	Solves the generic numeric problems.	It solves the problem in very narrow domain.
2.	It is sequential program where information and processing are combined.	The knowledge base is separated from the processing (inference engine). The program may not be sequential.
3.	Tested program never makes mistakes	The well tested expert system may make mistakes and gives wrong answer.
4.	No explanation is provided for output	An explanation is provided in most cases.
5.	When incorrect information is provided, the system may not function.	The system can arrive at a conclusion, even when some information is missing or incomplete.

3.1.5 Truth Maintenance Systems

Q5. What is a Truth Maintenance System (TMS)? Expalin.

Ans : **(Imp.)**

Meaning

A Truth Maintenance System is a knowledge representation and reasoning tool that assists AI systems in maintaining and updating their beliefs according to the available evidence.

A TMS is designed to manage inconsistencies and contradictions, allowing AI systems to reason with incomplete or uncertain information.

It achieves this by tracking dependencies between beliefs and assumptions, enabling the system to make informed decisions based on the current state of knowledge.

There are primarily two types of TMS: Justification-based TMS (JTMS) and Assumption-based TMS (ATMS).

Types

1. Justification-based TMS

A JTMS is a TMS that represents knowledge in the form of justifications. Each justification consists of a set of premises and a conclusion. When the premises of a justification are satisfied, the conclusion becomes a valid belief. A JTMS maintains consistency by ensuring that conflicting beliefs do not coexist.

2. Assumption-based TMS

An ATMS, on the other hand, represents knowledge in terms of assumptions and their consequences. It focuses on exploring alternative sets of assumptions and their corresponding belief states. This approach is particularly useful in scenarios where multiple explanations or solutions are possible.

TMS Components

A typical TMS consists of three primary components:

i) Nodes

Nodes represent the beliefs or assertions in a TMS. They can be either true or false, depending on the available evidence and assumptions.

ii) Justifications

Justifications are the links between nodes, representing the reasoning behind a belief. They contain a set of premises and a conclusion. When all premises are true, the conclusion is also considered true.

iii) Inference Procedures

Inference procedures are algorithms that manipulate nodes and justifications to update the belief state of the TMS. They are responsible for maintaining consistency and resolving conflicts between beliefs.

TMS Applications in Artificial Intelligence

TMS has been employed in various AI applications, including:

i) Expert Systems

Expert systems are AI programs that emulate human experts' reasoning and decision-making processes. TMS helps maintain the consistency of the knowledge base, allowing the system to reason with incomplete or uncertain information.

ii) Planning Systems

In planning systems, TMS assists in managing alternative plans and their assumptions, enabling the AI system to choose the most suitable plan based on the available information.

iii) Diagnosis Systems

TMS plays a crucial role in diagnostic systems, as it helps manage multiple hypotheses and their corresponding evidence. This allows the AI system to provide accurate diagnoses based on the available data.

Q6. Expalin the benefits and challanges of TMS?

Ans :

Benefits of Using a Truth Maintenance System

There are several benefits to incorporating a TMS into an AI system:

1. Consistency Management

TMS ensures that the AI system's beliefs remain consistent and free from contradictions.

2. Handling Uncertainty

TMS allows AI systems to reason with incomplete or uncertain information, making them more robust and adaptable.

3. Belief Revision

TMS enables AI systems to update their beliefs based on new evidence or changes in assumptions, ensuring that the system remains relevant and up-to-date.

4. Efficient Reasoning: TMS can improve the efficiency of the reasoning process by keeping track of dependencies between beliefs and assumptions, which can help avoid unnecessary computations.

Challenges of Truth Maintenance Systems

Despite their advantages, TMS also faces some challenges:

1. Scalability

As the size of the knowledge base grows, maintaining consistency and updating beliefs can become computationally expensive, making it challenging to scale TMS to larger AI systems.

2. Complexity

Managing dependencies between beliefs and assumptions can be complex, especially when dealing with multiple, conflicting beliefs.

3. Integration with Other AI Techniques

Combining TMS with other AI techniques, such as machine learning or natural language processing, can be challenging due to their different knowledge representation and reasoning mechanisms.

3.1.6 Application of Expert Systems

Q7. What are the applications of Expert systems? Write.

Ans : **(Imp.)**

There are several major application areas of expert system such as agriculture, education, environment, law manufacturing, medicine power system etc. Expert system is used to develop a large number of new products as well as new configurations of established products.

1. Expert System in Education

i) In the field of education, many of the expert system's application are embedded inside the Intelligent Tutoring System (ITS) by using techniques from adaptive hypertext and hypermedia.

ii) Most of the system usually will assist student in their learning by using adaptation techniques to personalize with the environment prior knowledge of student and student's ability to learn.

iii) Expert system in education has expanded very consistently from micro computer to web based and agent based technology. Web based expert system can provide an excellent alternative to private tutoring at any time from any place where internet is provided.

iv) Agent based expert system will help users by finding materials from the web based on the user's profile.

v) Expert system also had tremendous changes in the applying of methods and techniques.

vi) Expert system are beneficial as a teaching tools because it has equipped with the unique features which allow users to ask question on how, why and what format.

vii) When it is used in the class environment, surely it will give many benefit to student as it prepare the answer without referring to the teacher. Beside that, expert system is able to give reasons towards the given answer.

viii) Expert system had been used in several fields of study including computer animation, computer science and engineering, language teaching business study etc.

2. Expert system in Agriculture

i) The expert system for agriculture is same as like other fields. Here also the expert system uses the rule based structure and the knowledge of a human expert is captured in the form of IF-THEN rules and facts which are used to solve problems by answering questions typed at a keyboard attached to a computer.

ii) For example, in pest control, the need to spray, selection of a chemical to spray, mixing and application etc. The early, state of developing the expert systems are in the 1960's and 1970's were typically written on a mainframe computer in the programming language based on LISP. Some examples of these expert systems are MACSYMA developed at the Massachusetts Institute of Technology (MIT) for assisting individuals in solving complex mathematical problems.

iii) Other examples may be MYCIN, DENDRAL, and CALEX etc. The rises of the agricultural expert system are to help the farmers to do single point decisions, which to have a well planning for before start to do anything on their land.

iv) It is used to design an irrigation system for their plantation use. Also some of the other functions of agricultural expert system are:

a) To predict the extreme events such as thunderstorms and frost.

b) To select the most suitable crop variety.

c) Diagnosis of liver stock disorder and many more.

3. Expert System for a Particular Decision Problem

The expert system can be used as a stand alone advisory system for the specific knowledge domain. It also can provide decision support for a high level human expert. The main purposes, the rises of the expert system are as a delivery system for extension information, to provide management education for decision makers and for dissemination of up-to-date scientific information in a readily accessible and easily understood form, to agricultural researchers, advisers and farmers. By the help of an expert system, the farmers can produce a more high quality product to the citizen.

4. Expert System for Text Animation (ESTA)

The idea behind creating an expert system is that it can enable many people to benefit from the knowledge of one person – the expert. By providing it with a knowledge base for a certain subject area, ESTA can be used to create an expert system for the subject:

ESTA + Knowledge base = Expert System

Each knowledge base contains rules for a specific domain. A knowledge base for an expert system to give tax advice might contain rules relating marital status, mortgage commitments and age to the advisability of taking out a new life insurance policy. ESTA has all facilities to write the rules that will make up a knowledge base.

ESTA has an inference engine which can use the rules in the knowledge base to determine which advice is to be given to the expert system user. ESTA also features the ability for the expert system user to obtain answers to questions such as "how" and "why". ESTA is used by a knowledge engineer to create a knowledge base and by the expert system user to consult a knowledge base. Knowledge representation in ESTA is based on the items like sections, parameters, title.

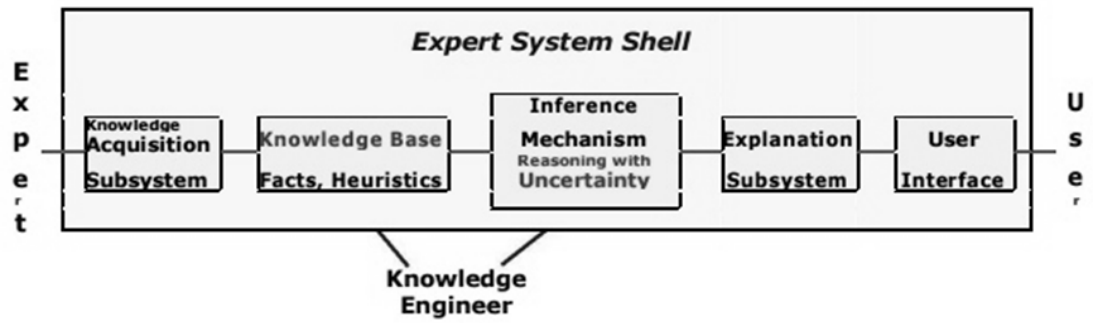
3.1.7 List Of Shells And Tools

Q8. Explain about expert system shells?

Ans :

Expert System Shells

An Expert system shell is a software development environment. It contains the basic components of expert systems. A shell is associated with a prescribed method for building applications by configuring and instantiating these components.



All these components are described in the next slide.

i) Knowledge Base

A store of factual and heuristic knowledge. Expert system tool provides one or more knowledge representation schemes for expressing knowledge about the application domain. Some tools use both Frames (objects) and IF-THEN rules. In PROLOG the knowledge is represented as logical statements.

ii) Reasoning Engine

Inference mechanisms for manipulating the symbolic information and knowledge in the knowledge base form a line of reasoning in solving a problem. The inference mechanism can range from simple modus ponens backward chaining of IF-THEN rules to Case-Based reasoning.

iii) Knowledge Acquisition Subsystem

A subsystem to help experts in build knowledge bases. However, collecting knowledge, needed to solve problems and build the knowledge base, is the biggest bottleneck in building expert systems.

iv) Explanation Subsystem

A subsystem that explains the system's actions. The explanation can range from how the final or intermediate solutions were arrived at justifying the need for additional data.

v) User Interface

A means of communication with the user. The user interface is generally not a part of the expert system technology. It was not given much attention in the past. However, the user interface can make a critical difference in the peeived utility of an Expert system.

Q9. Give the list of various Expert system shells and tools.

Ans : **(Imp.)**

Expert system shells and tools are software platforms and development environments that facilitate the creation of expert systems, which are computer programs designed to emulate the decision-making abilities of a human expert in a specific domain. Here is a list of some expert system shells and tools that were commonly used as of my last knowledge update in September 2021. Please note that the availability and popularity of these tools may have changed since then:

1. CLIPS (C Language Integrated Production System)

CLIPS is a widely used expert system development tool that provides a rule-based programming language for building expert systems. It is open-source and highly extensible.

2. Jess

Jess is a rule engine for the Java platform, which is similar in functionality to CLIPS. It allows developers to create expert systems using Java programming.

3. Drools

Drools is an open-source rules engine for Java that can be used to build expert systems and business rule management systems (BRMS). It provides a flexible and powerful rule-based system.

4. Prolog

Prolog is a logic programming language commonly used in expert system development. It is particularly well-suited for applications involving symbolic reasoning and knowledge representation.

5. Pyke

Pyke is a knowledge-based inference engine (expert system) in Python. It allows developers to represent knowledge and rules in a knowledge base and perform inference on that knowledge.

6. Cogito

Cogito is an expert system shell developed by NASA for use in various aerospace applications. It provides a rule-based reasoning engine and tools for knowledge acquisition and representation.

7. D3

D3 (Distributed Dynamic Development) is a distributed expert system development tool that allows for the creation of expert systems that can be distributed across multiple platforms.

8. ART (A Rule-Based Tool)

ART is a commercial expert system development tool that offers a graphical interface for creating and managing rule-based expert systems.

9. FuzzyCLIPS

FuzzyCLIPS is an extension of the CLIPS expert system tool that adds support for fuzzy logic, making it suitable for applications involving uncertain or imprecise information.

10. Knowledge Works

KnowledgeWorks is an expert system development tool for the Lisp programming language. It provides a range of features for building knowledge-based systems.

11. Expert System Shells in AI Frameworks

Some AI frameworks, like TensorFlow, PyTorch, and scikit-learn, provide libraries and tools for building expert systems in specific domains, such as machine learning and natural language processing.

12. OPS5

OPS5 is a production rule system developed for building expert systems. While it's not as widely used today, it was influential in the early development of expert systems.

3.2 Uncertainty Measure-Probability Theory

3.2.1 Introduction

Q10. What is uncertainity? State its causes.

Ans :

Learned knowledge representation using first-order logic and propositional logic with certainty, which means we were sure about the predicates. With this knowledge representation, we might write A'!B, which means if A is true then B is true, but consider a situation where we are not sure about whether A is true or not then we cannot express this statement, this situation is called uncertainty.

So to represent uncertain knowledge, where we are not sure about the predicates, we need uncertain reasoning or probabilistic reasoning.

Causes

Following are some leading causes of uncertainty to occur in the real world.

1. Information occurred from unreliable sources.
2. Experimental Errors
3. Equipment fault
4. Temperature variation
5. Climate change.

3.2.2 Probability Theory

Q11. Expalin briefly about probability theory.

Ans :

Meaning

Probability can be defined as a chance that an uncertain event will occur. It is the numerical measure of the likelihood that an event will occur. The value of probability always remains between 0 and 1 that represent ideal uncertainties.

i) $0 \leq P(A) \leq 1$, where $P(A)$ is the probability of an event A.

ii) $P(A) = 0$, indicates total uncertainty in an event A.

iii) $P(A) = 1$, indicates total certainty in an event A.

We can find the probability of an uncertain event by using the below formula.

$$\text{Probability of occurrence} = \frac{\text{Number of desired outcomes}}{\text{Total number of outcomes}}$$

1. $P(\neg A)$ = probability of a not happening event.

2. $P(\neg A) + P(A) = 1$.

1. Event

Each possible outcome of a variable is called an event.

2. Sample space

The collection of all possible events is called sample space.

3. Random variables

Random variables are used to represent the events and objects in the real world.

4. Prior probability

The prior probability of an event is probability computed before observing new information.

5. Posterior Probability

The probability that is calculated after all evidence or information has taken into account. It is a combination of prior probability and new information.

6. Conditional probability

Conditional probability is a probability of occurring an event when another event has already happened.

Let's suppose, we want to calculate the event A when event B has already occurred, "the probability of A under the conditions of B", it can be written as:

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

Where

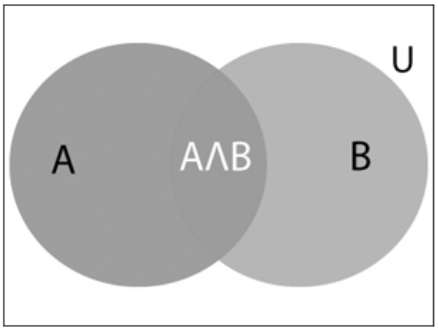
$P(A \wedge B)$ = Joint probability of a and B

P(B) = Marginal probability of B.

If the probability of A is given and we need to find the probability of B, then it will be given as:

$$P(B|A) = \frac{P(A \wedge B)}{P(A)}$$

It can be explained by using the below Venn diagram, where B is occurred event, so sample space will be reduced to set B, and now we can only calculate event A when event B is already occurred by dividing the probability of $P(A \wedge B)$ by P(B).



Example

In a class, there are 70% of the students who like English and 40% of the students who likes English and mathematics, and then what is the percent of students those who like English also like mathematics?

Solution :

Let, A is an event that a student likes Mathematics

B is an event that a student likes English.

$$P(A|B) = \frac{P(A \wedge B)}{P(A)}$$

$$= \frac{0.4}{0.7} = 57\%$$

Hence, 57% are the students who like English also like Mathematics.

3.2.3 Bayesian Belief Networks

Q12. Explain Bayes' Theorem.

Ans :

Bayes' Theorem

Bayes' theorem is also known as Bayes' rule, Bayes' law, or Bayesian reasoning, which determines the probability of an event with uncertain knowledge.

In probability theory, it relates the conditional probability and marginal probabilities of two random events.

It is a way to calculate the value of P(B|A) with the knowledge of P(A|B).

Bayes' theorem allows updating the probability prediction of an event by observing new information of the real world.

Example

If cancer corresponds to one's age then by using Bayes' theorem, we can determine the probability of cancer more accurately with the help of age.

Bayes' theorem can be derived using product rule and conditional probability of event A with known event B:

As from product rule we can write:

$P(A \wedge B) = P(A|B)\ P(B)$ or

Similarly, the probability of event B with known event A

$P(A \wedge B) = P(B|A)\ P(A)$

Equating right hand side of both the equations, we will get:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad \ldots (a)$$

The above equation (a) is called as Bayes' rule or Bayes' theorem. This equation is basic of most modern AI systems for probabilistic inference.

It shows the simple relationship between joint and conditional probabilities. Here,

P(A|B) is known as posterior, which we need to calculate, and it will be read as Probability of hypothesis A when we have occurred an evidence B.

P(B|A) is called the likelihood, in which we consider that hypothesis is true, then we calculate the probability of evidence.

P(A) is called the prior probability, probability of hypothesis before considering the evidence

P(B) is called marginal probability, pure probability of an evidence.

In the equation (a), in general, we can write P (B) = P(A)*P(B|Ai), hence the Bayes' rule can be written as:

$$P(A_i|B) = \frac{P(A_i) * P(B \mid A_i)}{\Sigma_{i=1}^{k} p(A_i) * P(B \mid A_i)}$$

Where $A_1, A_2, A_3, \ldots\ldots, A_n$ is a set of mutually exclusive and exhaustive events.

Applying Bayes' Rule

Bayes' rule allows us to compute the single term P(B|A) in terms of P(A|B), P(B), and P(A). This is very useful in cases where we have a good probability of these three terms and want to determine the fourth one. Suppose we want to perceive the effect of some unknown cause, and want to compute that cause, then the Bayes' rule becomes:

$$P(\text{cause}|\text{effect}) = \frac{P(\text{effect} \mid \text{cause})P(\text{cause})}{P(\text{effect})}$$

Q13. Solve the following examples using Bayes' theorem.

Ans : **(Imp.)**

Example - 1

What is the probability that a patient has diseases meningitis with a stiff neck?

Sol :

Given Data

A doctor is aware that disease meningitis causes a patient to have a stiff neck, and it occurs 80% of the time. He is also aware of some more facts, which are given as follows:

1. The Known probability that a patient has meningitis disease is 1/30,000.

2. The Known probability that a patient has a stiff neck is 2%.

Let a be the proposition that patient has stiff neck and b be the proposition that patient has meningitis. , so we can calculate the following as:

P(a|b) = 0.8

P(b) = 1/30000

P(a)= .02

$$P(b|a) = \frac{P(a \mid b)P(b)}{P(a)}$$

$$= \frac{0.8 * \left(\frac{1}{30000}\right)}{0.02}$$

$$= 0.001333333$$

Hence, we can assume that 1 patient out of 750 patients has meningitis disease with a stiff neck.

Example - 2

From a standard deck of playing cards, a single card is drawn. The probability that the card is king is 4/52, then calculate posterior probability P(King|Face), which means the drawn face card is a king card.

Sol :

$$P(king|face) = \frac{P(Face|king) * P(Face|king) * P(King)}{P(Face)} \quad \text{... (i)}$$

P(king): probability that the card is King = 4/52 = 1/13

P(face): probability that a card is a face card = 3/13

P(Face|King): probability of face card when we assume it is a king = 1

Putting all values in equation (i) we will get:

$$P(king|face) = \frac{1 * \left(\frac{1}{13}\right)}{\left(\frac{3}{13}\right)} = \frac{1}{3}$$, it is a probability that a face card is a king card.

Q14. Explain about Bayesian Belief Network in artificial intelligence.

Ans : **(Imp.)**

Bayesian belief network is key computer technology for dealing with probabilistic events and to solve a problem which has uncertainty. We can define a Bayesian network as:

"A Bayesian network is a probabilistic graphical model which represents a set of variables and their conditional dependencies using a directed acyclic graph."

It is also called a Bayes network, belief network, decision network, or Bayesian model.

Bayesian networks are probabilistic, because these networks are built from a probability distribution, and also use probability theory for prediction and anomaly detection.

Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

Real world applications are probabilistic in nature, and to represent the relationship between multiple events, we need a Bayesian network. It can also be used in various tasks including prediction, anomaly detection, diagnostics, automated insight, reasoning, time series prediction, and decision making under uncertainty.

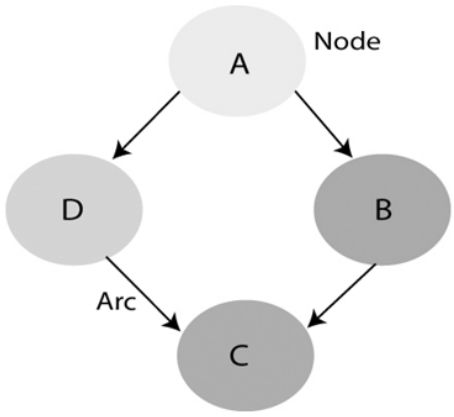
Bayesian Network can be used for building models from data and experts opinions, and it consists of two parts:

i) Directed Acyclic Graph

ii) Table of conditional probabilities.

The generalized form of Bayesian network that represents and solve decision problems under uncertain knowledge is known as an Influence diagram.

A Bayesian network graph is made up of nodes and Arcs (directed links), where:



1. Each node corresponds to the random variables, and a variable can be continuous or discrete.

2. Arc or directed arrows represent the causal relationship or conditional probabilities between random variables. These directed links or arrows connect the pair of nodes in the graph.

   These links represent that one node directly influence the other node, and if there is no directed link that means that nodes are independent with each other.

   i) In the above diagram, A, B, C, and D are random variables represented by the nodes of the network graph.

   ii) If we are considering node B, which is connected with node A by a directed arrow, then node A is called the parent of Node B.

   iii) Node C is independent of node A.

Note:

The Bayesian network graph does not contain any cyclic graph. Hence, it is known as a directed acyclic graph or DAG.

The Bayesian network has mainly two components:

i) Causal Component

ii) Actual numbers

Each node in the Bayesian network has condition probability distribution $P(X_i \mid Parent(X_i))$, which determines the effect of the parent on that node.

Bayesian network is based on Joint probability distribution and conditional probability. So let's first understand the joint probability distribution:

Joint Probability Distribution

If we have variables x1, x2, x3,....., xn, then the probabilities of a different combination of x1, x2, x3.. xn, are known as Joint probability distribution.

$P[x_1, x_2, x_3,.....,x_n]$, it can be written as the following way in terms of the joint probability distribution.

$= P[x_1 \mid x_2, x_3,....., x_n]P[x_2, x_3,....., x_n]$

$= P[x_1 \mid x_2, x_3,....., x_n]P[x_2 \mid x_3,....., x_n]....P[x_{n-1} \mid x_n]P[x_n].$

In general for each variable Xi, we can write the equation as:

$P(X_i \mid X_{i-1},....., X_1) = P(X_i \mid Parents(X_i))$

Q15. Solve the following problem by using Bayesian network.

Ans :

Let's understand the Bayesian network through an example by creating a directed acyclic graph:

Example

Harry installed a new burglar alarm at his home to detect burglary. The alarm reliably responds at detecting a burglary but also responds for minor earthquakes. Harry has two neighbors David and Sophia, who have taken a responsibility to inform Harry at work when they hear the alarm. David always calls Harry when he hears the alarm, but sometimes he got confused with the phone ringing and calls at that time too. On the other hand, Sophia likes to listen to high music, so sometimes she misses to hear the alarm. Here we would like to compute the probability of Burglary Alarm.

Calculate the probability that alarm has sounded, but there is neither a burglary, nor an earthquake occurred, and David and Sophia both called the Harry.

Sol :

1. The Bayesian network for the above problem is given below. The network structure is showing that burglary and earthquake is the parent node of the alarm and directly affecting the probability of alarm's going off, but David and Sophia's calls depend on alarm probability.

2. The network is representing that our assumptions do not directly perceive the burglary and also do not notice the minor earthquake, and they also not confer before calling.

3. The conditional distributions for each node are given as conditional probabilities table or CPT.

4. Each row in the CPT must be sum to 1 because all the entries in the table represent an exhaustive set of cases for the variable.

5. In CPT, a boolean variable with k boolean parents contains 2^K probabilities. Hence, if there are two parents, then CPT will contain 4 probability values.

List of all events occurring in this Betwork

i) Burglary (B)

ii) Earthquake(E)

iii) Alarm(A)

iv) David Calls(D)

v) Sophia calls(S)

We can write the events of problem statement in the form of probability: P[D, S, A, B, E], can rewrite the above probability statement using joint probability distribution:

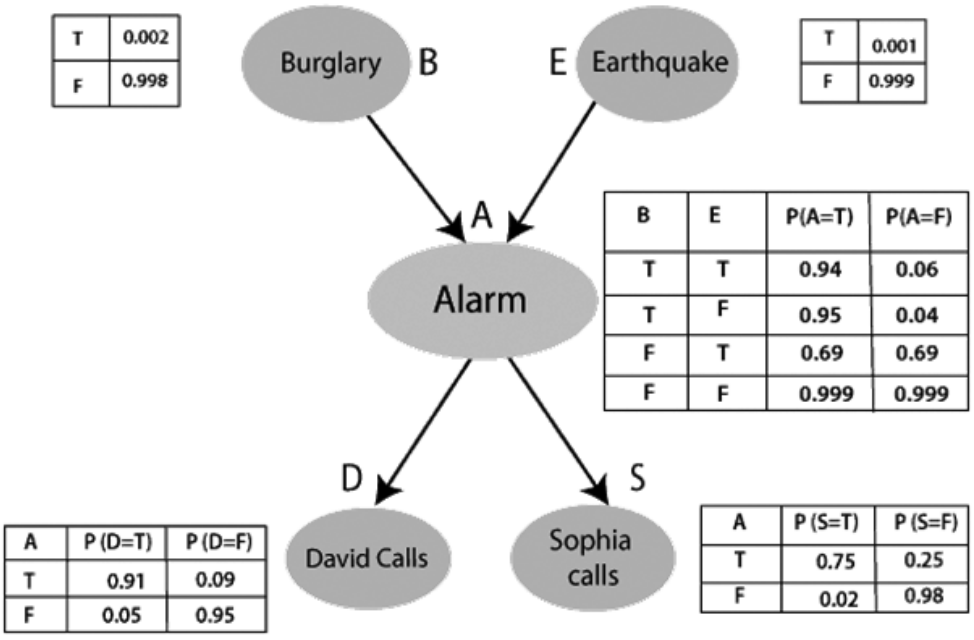
P[D, S, A, B, E] = P[D | S, A, B, E]. P[S, A, B, E]

=P[D | S, A, B, E]. P[S | A, B, E].P[A, B, E]

= P [D| A]. P [S| A, B, E]. P[A, B, E]

= P[D | A]. P[S | A]. P[A| B, E]. P[B, E]

= P[D | A]. P[S | A].P[A| B, E]. P[B |E]. P[E]



T	0.002
F	0.998

T	0.001
F	0.999

B	E	P(A=T)	P(A=F)
T	T	0.94	0.06
T	F	0.95	0.04
F	T	0.69	0.69
F	F	0.999	0.999

A	P (D=T)	P (D=F)
T	0.91	0.09
F	0.05	0.95

A	P (S=T)	P (S=F)
T	0.75	0.25
F	0.02	0.98

Let's take the observed probability for the Burglary and earthquake component:

P(B= True) = 0.002, which is the probability of burglary.

P(B= False)= 0.998, which is the probability of no burglary.

P(E= True)= 0.001, which is the probability of a minor earthquake

P(E= False)= 0.999, Which is the probability that an earthquake not occurred.

We can provide the conditional probabilities as per the below tables:

Conditional probability table for Alarm A:

The Conditional probability of Alarm A depends on Burglar and earthquake:

B	E	P(A= True)	P(A= False)
True	True	0.94	0.06
True	False	0.95	0.04
False	True	0.31	0.69
False	False	0.001	0.999

Conditional probability table for David Calls:

The Conditional probability of David that he will call depends on the probability of Alarm.

A	P(D= True)	P(D= False)
True	0.91	0.09
False	0.05	0.95

Conditional probability table for Sophia Calls

The Conditional probability of Sophia that she calls is depending on its Parent Node "Alarm."

A	P(S= True)	P(S= False)
True	0.75	0.25
False	0.02	0.98

From the formula of joint distribution, we can write the problem statement in the form of probability distribution:

$P(S, D, A, \neg B, \neg E) = P(S|A) * P(D|A) * P(A|\neg B \wedge \neg E) * P(\neg B) * P(\neg E).$

$= 0.75 * 0.91 * 0.001 * 0.998 * 0.999$

$= 0.00068045.$

Hence, a Bayesian network can answer any query about the domain by using Joint distribution.

The semantics of Bayesian Network

There are two ways to understand the semantics of the Bayesian network, which is given below:

1. To understand the network as the representation of the Joint probability distribution.

It is helpful to understand how to construct the network.

2. To understand the network as an encoding of a collection of conditional independence statements.

It is helpful in designing inference procedure.

3.2.4 Certainty Factor Theory

Q16. Explin the uses of certainity factor in AI?

Ans :

Certainty Factor

The Certainty factor is a measure of the degree of confidence or belief in the truth of a proposition or hypothesis. In AI, the certainty factor is often used in rule-based systems to evaluate the degree of certainty or confidence of a given rule.

Certainty factors are used to combine and evaluate the results of multiple rules to make a final decision or prediction. For example, in a medical diagnosis system, different symptoms can be associated with different rules that determine the likelihood of a particular disease. The certainty factors of each rule can be combined to produce a final diagnosis with a degree of confidence.

In Artificial Intelligence, the numerical values of the certainty factor represent the degree of confidence or belief in the truth of a proposition or hypothesis. The numerical scale typically ranges from -1 to 1, and each value has a specific meaning:

i) -1: Complete disbelief or negation

This means that the proposition or hypothesis is believed to be false with absolute certainty.

ii) 0: Complete uncertainty

This means that there is no belief or confidence in the truth or falsehood of the proposition or hypothesis.

iii) +1: Complete belief or affirmation

This means that the proposition or hypothesis is believed to be true with absolute certainty.

Values between 0 and 1 indicate varying degrees of confidence that the proposition or hypothesis is true.

Values between 0 and -1 indicate varying degrees of confidence that the proposition or hypothesis is false.

For example, a certainty factor of 0.7 indicates a high degree of confidence that the proposition or hypothesis is true, while a certainty factor of -0.3 indicates a moderate degree of confidence that the proposition or hypothesis is false.

Practical Applications of Certainty Factor

Certainty factor has practical applications in various fields of artificial intelligence, including:

1. Medical Diagnosis

In medical diagnosis systems, certainty factors are used to evaluate the probability of a patient having a particular disease based on the presence of specific symptoms.

2. Fraud Detection

In financial institutions, certainty factors can be used to evaluate the likelihood of fraudulent activities based on transaction patterns and other relevant factors.

3. Customer Service

In customer service systems, certainty factors can be used to evaluate customer requests or complaints and provide appropriate responses.

4. Risk Analysis

In risk analysis applications, certainty factors can be used to assess the likelihood of certain events occurring based on historical data and other factors.

5. Natural Language Processing

In natural language processing applications, certainty factors can be used to evaluate the accuracy of language models in interpreting and generating human language.

Limitations of Certainty Factor

Although the certainty factor is a useful tool for representing and reasoning about uncertain or incomplete information in artificial intelligence, there are some limitations to its use. Here are some of the main limitations of the certainty factor:

1. Difficulty in assigning accurate certainty values

Assigning accurate certainty values to propositions or hypotheses can be challenging, especially when dealing with complex or ambiguous situations. This can lead to faulty results and outcomes.

2. Difficulty in combining certainty values

Combining certainty values from multiple sources can be complex and difficult to achieve accurately. Different sources may have different levels of certainty and reliability, which can lead to inconsistent or conflicting results.

3. Inability to handle conflicting evidence

In some cases, conflicting evidence may be presented, making it difficult to determine the correct certainty value for a proposition or hypothesis.

4. Limited range of values

The numerical range of the certainty factor is limited to -1 to 1, which may not be sufficient to capture the full range of uncertainty in some situations.

5. Subjectivity

The Certainty factor relies on human judgment to assign certainty values, which can introduce subjectivity and bias into the decision-making process.

3.2.5 Dempster-Shafer Theory

Q17. Explain briefly about Dempster Shafer Theory

Ans : **(Imp.)**

Dempster-Shafer Theory was given by Arthur P. Dempster in 1967 and his student Glenn Shafer in 1976. This theory was released because of the following reason:

i) Bayesian theory is only concerned about single evidence.

ii) Bayesian probability cannot describe ignorance.

DST is an evidence theory, it combines all possible outcomes of the problem. Hence it is used to solve problems where there may be a chance that a piece of different evidence will lead to some different result.

The uncertainty in this model is given by:

1. Consider all possible outcomes.

2. Belief will lead to belief in some possibility by bringing out some evidence. (What is this supposed to mean?)

3. Plausibility will make evidence compatible with possible outcomes.

Example

Let us consider a room where four people are present, A, B, C, and D. Suddenly the lights go out and when the lights come back, B has been stabbed in the back by a knife, leading to his death. No one came into the room and no one left the room. We know that B has not committed suicide. Now we have to find out who the murderer is.

To solve these there are the following possibilities:

i) Either {A} or {C} or {D} has killed him.

ii) Either {A, C} or {C, D} or {A, D} have killed him.

iii) Or the three of them have killed him i.e; {A, C, D}

iv) None of them have killed him {o} (let's say).

There will be possible evidence by which we can find the murderer by the measure of plausibility.

Using the above example we can say:

Set of possible conclusion (P): {p1, p2….pn}

where P is a set of possible conclusions and cannot be exhaustive, i.e. at least one (p) I must be true.

(p)I must be mutually exclusive.

Power Set will contain 2^n elements where n is the number of elements in the possible set.

For eg:-

If P = { a, b, c}, then Power set is given as

{o, {a}, {b}, {c}, {a, d}, {d ,c}, {a, c}, {a, c ,d }}= 2^3 elements.

Mass function m(K)

It is an interpretation of m({K or B}) i.e; it means there is evidence for {K or B} which cannot be divided among more specific beliefs for K and B.

Belief in K

The belief in element K of Power Set is the sum of masses of the element which are subsets of K. This can be explained through an example

Lets say K = {a, d, c}

Bel(K) = m(a) + m(d) + m(c) + m(a, d) + m(a, c) + m(d, c) + m(a, d, c)

Plausibility in K

It is the sum of masses of the set that intersects with K.

i.e; Pl(K) = m(a) + m(d) + m(c) + m(a, d) + m(d, c) + m(a, c) + m(a, d, c)

Characteristics of Dempster Shafer Theory:

1. It will ignorance part such that the probability of all events aggregate to 1. (What is this supposed to mean?)

2. Ignorance is reduced in this theory by adding more and more evidence.

3. Combination rule is used to combine various types of possibilities.

Advantages

1. As we add more information, the uncertainty interval reduces.
2. DST has a much lower level of ignorance.
3. Diagnose hierarchies can be represented using this.
4. Person dealing with such problems is free to think about evidence.

Disadvantages

In this, computation effort is high, as we have to deal with 2^n sets

UNIT IV

Machine-Learning Paradigms: Introduction, Machine Learning Systems, Supervised and Unsupervised Learning, Inductive Learning, Learning Decision Trees (Suggested Reading 2), Deductive Learning, Clustering, Support Vector Machines.

Artificial Neural Networks: Introduction, Artificial Neural Networks, Single-Layer Feed-Forward Networks, Multi-Layer Feed-Forward Networks, Radial-Basis Function Networks, Design Issues of Artificial Neural Networks, Recurrent Networks.

4.1 Machine-Learning Paradigms

4.1.1 Introduction

Q1. What is machine learning? State the components of machine learning.

Ans :

ii) Learning

"Learning denotes changes in a system that enables system to do the same task more efficiently next time."

ii) Machine Learning Definition

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P,if its performance at tasks in T,as measured by P, improves with experience E.

Components of Learning System

i) Performance Element

The performance element is the agent that acts in the world .It percepts and decides on external actions.

ii) Learning Element

It responsible for making improvements, takes knowledge about performance element and some feedback ,determines how to modify performance element.

iii) Critic

It tells the learning element how agent is doing by comparing with the fixed standard of performance.

iv) Problem Generator

This component suggests problems or actions that will generate new examples or experience that helps the system to train further.

Let us see the role of each component with an example.

Example: Automated Taxi on city roads

i) Performance Element: Consists of knowledge and procedures for driving actions.

Eg: turning, accelerating,breaking are the performance elements on roads.

ii) Learning Element: It formulates goals.

Eg: learn rules for breaking,accelerating,learn geography of the city.

iii) Critic: Observes world and passes information to learning element.

Eg:quick right turn across three lanes of traffic, observe reaction of other drivers.

Problem Generator: Try south city road

4.1.2 Machine Learning Systems

Q2. Explain the main key components of Machine Learning Systems?

Ans : **(Imp.)**

Machine Learning (ML) systems in artificial intelligence (AI) are a subset of AI that focus on creating algorithms and models that enable computers to learn from data and improve their performance on specific tasks without being explicitly programmed. These systems enable computers to automatically acquire knowledge and improve their performance through experience, similar to how humans learn.

Components

Machine Learning systems involve several key components and processes, which can be categorized as follows:

1. **Data Collection and Preprocessing**

   i) **Data Collection:** ML systems require relevant and representative data for training and evaluation.

   ii) **Data Preprocessing:** Raw data is often noisy and may contain inconsistencies. Preprocessing involves cleaning, transforming, and formatting data for use in training ML models.

2. **Feature Extraction and Selection:**

   i) **Feature Extraction:** Relevant information is extracted from raw data and represented as features that the ML model can understand.

   ii) **Feature Selection:** Choosing the most informative features to reduce dimensionality and improve model performance.

3. **Model Selection and Training:**

   i) **Model Selection:** Choosing an appropriate ML algorithm or model architecture based on the problem's characteristics.

   ii) **Model Training:** The selected model is trained using labeled data (supervised learning) or unlabelled data (unsupervised learning). During training, the model learns patterns and relationships in the data.

4. **Evaluation and Validation:**

   i) **Model Evaluation:** The trained model's performance is assessed using validation and test datasets to ensure it generalizes well to new, unseen data.

   ii) **Hyperparameter Tuning:** Adjusting hyperparameters (settings that control the learning process) to optimize the model's performance.

5. **Deployment and Inference:**

   i) **Deployment:** The trained ML model is deployed to a production environment to make predictions or decisions.

   ii) **Inference:** The deployed model processes new data and produces predictions or outputs.

6. **Continuous Learning and Improvement:**

   i) **Feedback Loop:** ML systems often incorporate a feedback loop where new data is continuously collected and used to retrain and update the model, ensuring it adapts to changing conditions.

Machine Learning systems can be categorized into different types based on the learning paradigm:

1. **Supervised Learning**

   Models learn from labeled training data, where input-output pairs are provided. The goal is to learn a mapping from inputs to outputs, enabling predictions on new, unseen data.

2. **Unsupervised Learning**

   Models learn from unlabelled data and discover patterns, structures, or relationships in the data. Clustering and dimensionality reduction are common tasks.

3. **Semi-Supervised Learning**

   Combines elements of supervised and unsupervised learning, using a mix of labeled and unlabelled data for training.

4. **Reinforcement Learning**

   Agents learn to make sequential decisions by interacting with an environment. They receive feedback (rewards) for their actions and learn to maximize cumulative rewards.

5. **Deep Learning**

   Utilizes artificial neural networks with multiple layers (deep architectures) to model complex patterns and features. Deep learning has achieved significant success in various domains like image recognition, natural language processing, and game playing.

Machine Learning systems have been applied to a wide range of applications, including image and speech recognition, natural language processing, recommendation systems, medical diagnosis, autonomous vehicles, finance, and more. These systems have the capability to extract insights and make predictions from large and complex datasets, leading to advancements in AI and technology as a whole.

Q3. Explain briefly about learning paradigms.

Ans :

1. Rote Learning

Rote learning technique avoids understanding the inner complexities but focuses on memorizing the material so that it can be recalled by the learner exactly the way it read or heard.

Learning by memorization: which avoids understanding the inner complexities the subject that is being learned.

Learning something from Repeating:saying the same thing and trying to remember how to say it;it does not help to understand ,it helps to remember ,like we learn a poem,song ,etc.

$$\phi\{z(n), y(n)\}_{x=1-N} \Rightarrow (A \Rightarrow C)$$

There are two types of inductive learning,

i) Supervised

ii) Unsupervised

2. Supervised learning: (The machine has access to a teacher who corrects it.)

Learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). Example : Face recognition.

Unsupervised Learning: (No access to teacher. Instead, the machine must search for "order" and "structure" in the environment).

Since there is no desired output in this case that is provided therefore categorization is done so that the algorithm differentiates correctly between the face of a horse, cat or human (clustering of data)

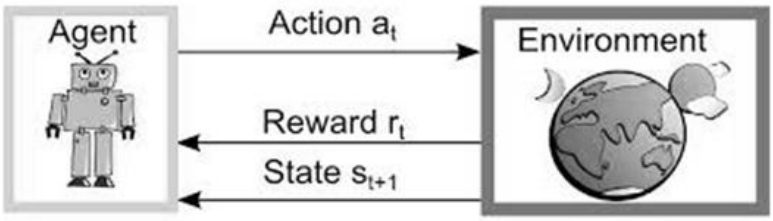
3. Clustering

In clustering or unsupervised learning, the target features are not given in the training examples. The aim is to construct a natural classification that can be used to cluster the data. The general idea behind clustering is to partition the examples into clusters or classes. Each class predicts feature values for the examples in the class. Each clustering has a prediction error on the predictions. The best clustering is the one that minimizes the error.

Example: An intelligent tutoring system may want to cluster students' learning behavior so that strategies that work for one member of a class may work for other members.

4. Reinforcement Learning:

Imagine a robot that can act in a world, receiving rewards and punishments and determining from these what it should do. This is the problem of reinforcement learning. Most Reinforcement Learning research is conducted with in the mathematical framework of Markov Decision Process.



Reinforcement Learning Setup

4.1.3 Supervised and Unsupervised Learning

Q4. Explain the categories of supervised machine learning.

Ans :

Supervised Machine Learning

As the name suggests, Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cat and dog images. So, first, we will provide the training to the machine to understand the images, such as the shape & size of the tail of a cat and a dog, the Shape of eyes, color, and height (dogs are taller, cats are smaller), etc. After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, color, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y). Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "Yes" or No, Male or Female, Red or Blue, etc. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are Spam Detection, Email filtering, etc.

Some popular classification algorithms are given below:

i) Random Forest Algorithm

ii) Decision Tree Algorithm

iii) Logistic Regression Algorithm

iv) Support Vector Machine Algorithm

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather predictions, etc.

Some popular Regression algorithms are given below:

i) Simple Linear Regression Algorithm

ii) Multivariate Regression Algorithm

iii) Decision Tree Algorithm

iv) Lasso Regression

State the Advantages and Disadvantages applicatons of Supervised Learning

Advantages

1. Since supervised learning work with the labeled dataset so we can have an exact idea about the classes of objects.
2. These algorithms are helpful in predicting the output on the basis of prior experience.

Disadvantages

1. These algorithms are not able to solve complex tasks.
2. It may predict the wrong output if the test data is different from the training data.
3. It requires lots of computational time to train the algorithm.

Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

1. Image Segmentation

Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.

2. Medical Diagnosis

Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labeled data with labels for disease conditions. With such a process, the machine can identify a disease for the new pat

3. Fraud Detection

Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.

4. Spam Detection

In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.

5. Speech Recognition

Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

Q5. Explain the technique of unsupervised learning in Machine Language.

Ans :

Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means that in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categorize the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as color difference, and shape difference, and predict the output when it is tested with the test dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

1. Clustering

2. Association

1. Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

i) K-Means Clustering algorithm

ii) Mean-shift algorithm

iii) DBSCAN Algorithm

iv) Principal Component Analysis

v) Independent Component Analysis

2. Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production, etc.

Some popular algorithms of Association rule learning are Apriori Algorithm, Eclat, FP-growth algorithm.

Q6. Explain the Advantages and Disadvantages and applications of Unsupervised Learning.

Ans :

Advantages

1. These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.

2. Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

Disadvantages

1. The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.

2. Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

Applications of Unsupervised Learning

1. Network Analysis

Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.

2. Recommendation Systems

Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.

3. Anomaly Detection

Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent trans

4. Singular Value Decomposition

Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

4.1.4 Inductive Learning

Q7. What is Inductive Learning Algorithm?

Ans :

Inductive Learning Algorithm (ILA) is an iterative and inductive machine learning algorithm that is used for generating a set of classification rules, which produces rules of the form "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

There are basically two methods for knowledge extraction firstly from domain experts and then with machine learning. For a very large amount of data, the domain experts are not very useful and reliable. So we move towards the machine learning approach for this work. To use machine learning One method is to replicate the expert's logic in the form of algorithms but this work is very tedious, time taking, and expensive. So we move towards the inductive algorithms which generate the strategy for performing a task and need not instruct separately at each step.

Why you should use Inductive Learning?

The ILA is a new algorithm that was needed even when other reinforcement learnings like ID3 and AQ were available.

1. The need was due to the pitfalls which were present in the previous algorithms, one of the major pitfalls was the lack of generalization of rules.
2. The ID3 and AQ used the decision tree production method which was too specific which were difficult to analyze and very slow to perform for basic short classification problems.
3. The decision tree-based algorithm was unable to work for a new problem if some attributes are missing.
4. The ILA uses the method of production of a general set of rules instead of decision trees, which overcomes the above problems

Basic Requirements to Apply Inductive Learning Algorithm

1. List the examples in the form of a table 'T' where each row corresponds to an example and each column contains an attribute value.
2. Create a set of m training examples, each example composed of k attributes and a class attribute with n possible decisions.
3. Create a rule set, R, having the initial value false.
4. Initially, all rows in the table are unmarked.

Necessary Steps for Implementation

Step 1

Divide the table 'T' containing m examples into n sub-tables (t1, t2,.....tn). One table for each possible value of the class attribute. (repeat steps 2-8 for each sub-table)

Step 2

Initialize the attribute combination count ' j ' = 1.

Step 3

For the sub-table on which work is going on, divide the attribute list into distinct combinations, each combination with 'j ' distinct attributes.

Step 4

For each combination of attributes, count the number of occurrences of attribute values that appear under the same combination of attributes in unmarked rows of the sub-table under consideration, and at the same time, not appears under the same combination of attributes of other sub-tables. Call the first combination with the maximum number of occurrences the max-combination ' MAX'.

Step 5

If 'MAX' = = null, increase ' j ' by 1 and go to Step 3.

Step 6

Mark all rows of the sub-table where working, in which the values of 'MAX' appear, as classified.

Step 7

Add a rule (IF attribute = "XYZ" –> THEN decision is YES/ NO) to R whose left-hand side will have attribute names of the 'MAX' with their values separated by AND, and its right-hand side contains the decision attribute value associated with the sub-table.

Step 8:

If all rows are marked as classified, then move on to process another sub-table and go to Step 2. Else, go to Step 4. If no sub-tables are available, exit with the set of rules obtained till then.

An example showing the use of ILA suppose an example set having attributes Place type, weather, location, decision, and seven examples, our task is to generate a set of rules that under what condition is the decision.

Example

S.No.	Place type	Weather	Location	Decision
1.	hilly	winter	kullu	Yes
2.	mountain	windy	Mumbai	No
3.	mountain	windy	Shimla	Yes
4.	beach	windy	Mumbai	No
5.	beach	warm	goa	Yes
6.	beach	windy	goa	No
7.	beach	warm	Shimla	Yes

Subset – 1

S.No.	Place type	Weather	Location	Decision
1.	hilly	winter	kullu	Yes
2.	mountain	windy	Shimla	Yes
3.	beach	warm	goa	Yes
4.	beach	warm	Shimla	Yes

Subset – 2

S.No.	Place type	Weather	Location	Decision
5.	mountain	windy	Mumbai	No
6.	beach	windy	Mumbai	No
7.	beach	windy	goa	No

i) **At iteration 1** rows 3 & 4 column weather is selected and rows 3 & 4 are marked. the rule is added to R IF the weather is warm then a decision is yes.

ii) **At iteration 2** row 1 column place type is selected and row 1 is marked. the rule is added to R IF the place type is hilly then the decision is yes.

iiii) **At iteration 3** row 2 column location is selected and row 2 is marked. the rule is added to R IF the location is Shimla then the decision is yes.

iv) **At iteration 4** row 5&6 column location is selected and row 5&6 are marked. the rule is added to R IF the location is Mumbai then a decision is no.

v) **At iteration 5** row 7 column place type & the weather is selected and row 7 is marked. the rule is added to R IF the place type is beach AND the weather is windy then the decision is no.

Finally, we get the rule set:- Rule Set

i) Rule 1: IF the weather is warm THEN the decision is yes.

ii) Rule 2: IF the place type is hilly THEN the decision is yes.

iii) Rule 3: IF the location is Shimla THEN the decision is yes.

iv) Rule 4: IF the location is Mumbai THEN the decision is no.

v) Rule 5: IF the place type is beach AND the weather is windy THEN the decision is no.

4.1.5 Learning Decision Trees

Q8. Explain briefly about learning decision trees in ML.

Ans :

A decision tree is a type of supervised learning algorithm that is commonly used in machine learning to model and predict outcomes based on input data. It is a tree-like structure where each internal nodetestson attribute, each branch corresponds to attribute value and each leaf node represents the final decision or prediction.

Example

Example of a decision tree Suppose we want to build a decision tree to predict whether a person is likely to buy a new car based on their demographic and behavior data. The decision tree starts with the root node, which represents the entire dataset. The root node splits the dataset based on the "income" attribute. If the person's income is less than or equal to $50,000, the decision tree follows the left branch, and if the income is greater than $50,000, the decision tree follows the right branch.

The left branch leads to a node that represents the "age" attribute. If the person's age is less than or equal to 30, the decision tree follows the left branch, and if the age is greater than 30, the decision tree follows the right branch. The right branch leads to a leaf node that predicts that the person is unlikely to buy a new car.

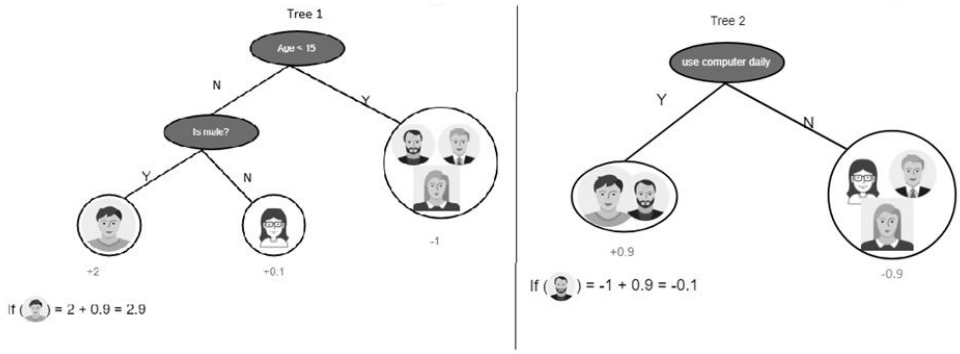
Below are some assumptions that we made while using the decision tree:

1. At the beginning, we consider the whole training set as the root.

2. Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.

3. On the basis of attribute values, records are distributed recursively.

4. We use statistical methods for ordering attributes as root or the internal node.



As you can see from the above image the Decision Tree works on the Sum of Product form which is also known as Disjunctive Normal Form. In the above image, we are predicting the use of computer in the daily life of people. In the Decision Tree, the major challenge is the identification of the attribute for the root node at each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain

2. Gini Index

1. Information Gain

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy. ***Definition***: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

Entropy

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content. ***Definition***: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

Example:

For the set X = {a,a,a,b,b,b,b,b}

Total instances: 8

Instances of b: 5

Instances of a: 3

Entropy H(X) = -\left [\left (\frac{3}{8} \right)log_{2}\frac{3}{8} + \left (\frac{5}{8} \right)log_{2}\frac{5}{8} \right]

= -[0.375 * (-1.415) + 0.625 * (-0.678)]

= -(-0.53-0.424)

= 0.954

Q9. How to build the decision tree by using Information Gain.

Ans :

1. Start with all training instances associated with the root node

2. Use info gain to choose which attribute to label each node with

3. Note: No root-to-leaf path should contain the same discrete attribute twice

4. Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

5. If all positive or all negative training instances remain, the label that node "yes" or "no" accordingly

6. If no attributes remain, label with a majority vote of training instances left at that node

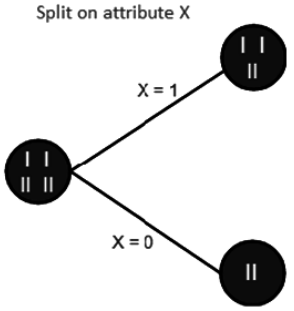
7. If no instances remain, label with a majority vote of the parent's training instances.

Example:

Now, let us draw a Decision Tree for the following data using Information gain. Training set: 3 features and 2 classes

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

Here, we have 3 features and 2 output classes. To build a decision tree using Information gain. We will take each of the features and calculate the information for each feature.



$E_{parent} = 1$

$GAIN = 1 - (3/4)(0.9184) - (1/4)(0) = 0.3112$

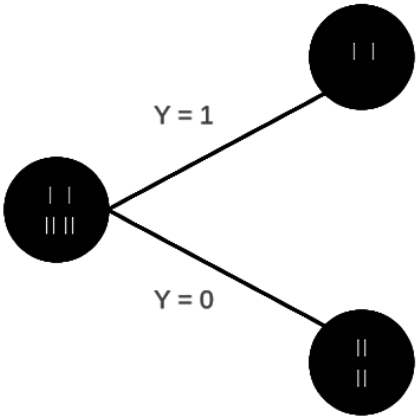
$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$

$= 0.5284 + 0.39$

$= 0.9184$

$E_{child2} = 0$

Split on feature X



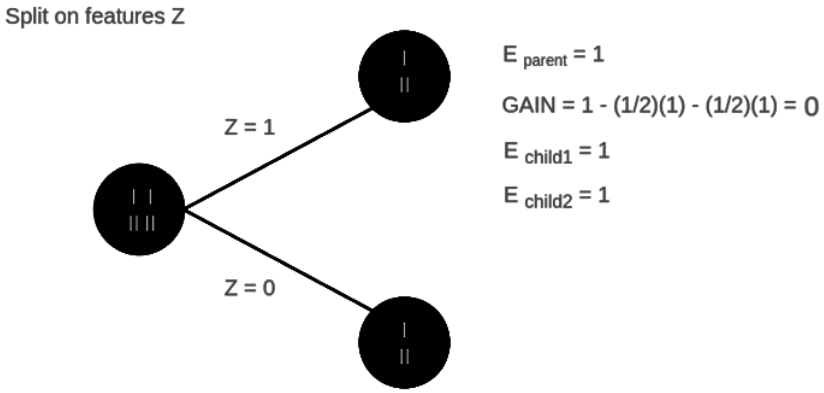
$E_{parent} = 1$

$GAIN = 1 - (1/2)(0) - (1/2)(0) = 1$

$E_{child1} = 0$

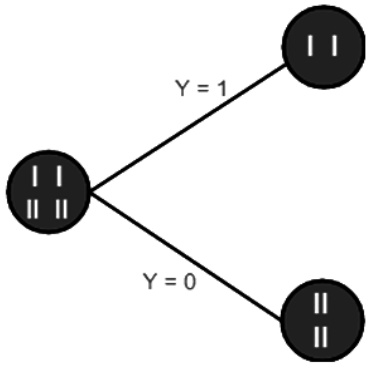
$E_{child2} = 0$

Split on feature Y



Split on feature Z

From the above images, we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best-suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains a pure subset of the target variable. So we don't need to further split the dataset. The final tree for the above dataset would look like this:



2. Gini Index

i) Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.

ii) It means an attribute with a lower Gini index should be preferred.

iii) Sklearn supports "Gini" criteria for Gini Index and by default, it takes "gini" value.

iv) The Formula for the calculation of the Gini Index is given below.

The Gini Index is a measure of the inequality or impurity of a distribution, commonly used in decision trees and other machine learning algorithms. It ranges from 0 to 1, where 0 represents perfect equality (all values are the same) and 1 represents perfect inequality (all values are different).

Some additional features and characteristics of the Gini Index are:

i) It is calculated by summing the squared probabilities of each outcome in a distribution and subtracting the result from 1.

ii) A lower Gini Index indicates a more homogeneous or pure distribution, while a higher Gini Index indicates a more heterogeneous or impure distribution.

iii) In decision trees, the Gini Index is used to evaluate the quality of a split by measuring the difference between the impurity of the parent node and the weighted impurity of the child nodes.

iv) Compared to other impurity measures like entropy, the Gini Index is faster to compute and more sensitive to changes in class probabilities.

v) One disadvantage of the Gini Index is that it tends to favor splits that create equally sized child nodes, even if they are not optimal for classification accuracy.

vi) In practice, the choice between using the Gini Index or other impurity measures depends on the specific problem and dataset, and often requires experimentation and tuning

4.1.6 Deductive Learning

Q10. What is deductive learning ? Write about it.

Ans :

Meaning

Deductive learning refers to a method of teaching that may be more suitable in introductory level courses who need a clear foundation from which to begin with a new language item. Learners who are accustomed to a more traditional approach to learning and therefore lack the training to find rules themselves may struggle with this method.

Deductive learning in AI refers to a process where an AI system draws specific conclusions from general principles or premises. It is a top-down approach where the AI derives logical conclusions based on established facts or rules. In other words, deductive learning involves reasoning from general information to make specific predictions or decisions.

Here's an elaborated example to help you understand deductive learning in AI:

Example: Sudoku Solver

Imagine you are developing an AI system to solve Sudoku puzzles using deductive learning. Sudoku is a logic-based number placement puzzle where you have a 9x9 grid divided into nine 3x3 subgrids or regions. The goal is to fill in the grid with digits from 1 to 9, such that each row, column, and subgrid contains all the digits without repetition.

1. Initial Setup

The AI is given a partially filled Sudoku puzzle as input. Some cells already have numbers placed, while others are blank.

2. General Principles

The AI is equipped with a set of rules or principles that define the valid solutions for a Sudoku puzzle: a. Each row must contain all digits from 1 to 9 without repetition. b. Each column must contain all digits from 1 to 9 without repetition. c. Each 3x3 subgrid must contain all digits from 1 to 9 without repetition.

3. Deductive Reasoning

The AI examines the puzzle and applies deductive reasoning based on the general principles to deduce the possible values for each empty cell.

For example, let's say the AI observes the following situation:

i) In a certain row, the numbers 1, 3, 4, 5, and 7 are already filled.

ii) In the same row, only two cells are empty.

4. Deductive Inference

The AI deduces that the numbers 2, 6, 8, and 9 must fill the remaining two empty cells in that row to satisfy the row constraint.

5. Iterative Process

The AI continues to apply deductive reasoning, updating its deductions and making new conclusions based on the interactions between rows, columns, and subgrids. It eliminates possibilities for each cell based on the values already placed in related cells.

6. Solution

Through successive rounds of deductive reasoning, the AI eventually deduces the correct numbers for all cells in the Sudoku puzzle, adhering to the rules of Sudoku.

In this example, the AI used deductive learning to derive specific conclusions about the values of individual cells in the Sudoku puzzle based on the general principles of the game. Deductive reasoning allowed the AI to make informed decisions and solve the puzzle logically.

4.1.7 Clustering

Q11. What is clustering? Explain various types of clustering methods

Ans : **(Imp.)**

Meaning

It is basically a type of unsupervised learning method. An unsupervised learning method is a method in which we draw references from datasets consisting of input data without labeled responses. Generally, it is used as a process to find meaningful structure, explanatory underlying processes, generative features, and groupings inherent in a set of examples.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

Example

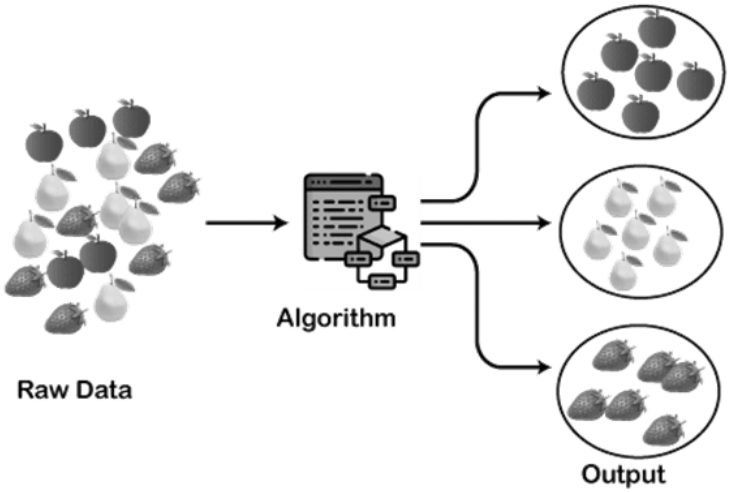
The clustering technique can be widely used in various tasks.

Some most common uses of this technique are:

1. Market Segmentation
2. Statistical data analysis
3. Social network analysis
4. Image segmentation
5. Anomaly detection, etc.

Apart from these general usages, it is used by the Amazon in its recommendation system to provide the recommendations as per the past search of products. Netflix also uses this technique to recommend the movies and web-series to its users as per the watch history.

The below diagram explains the working of the clustering algorithm. We can see the different fruits are divided into several groups with similar properties.



Types

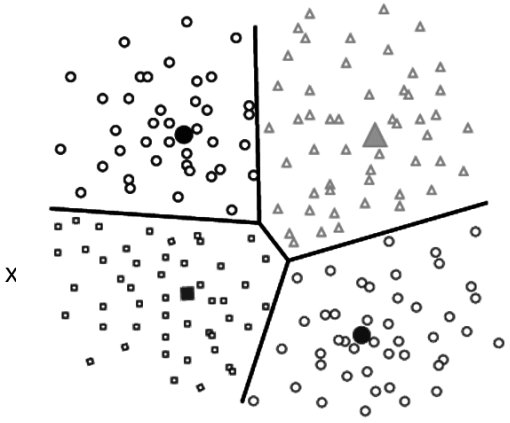
The clustering methods are broadly divided into Hard clustering (datapoint belongs to only one group) and Soft Clustering (data points can belong to another group also). But there are also other various approaches of Clustering exist. Below are the main clustering methods used in Machine learning:

1. Partitioning Clustering
2. Density-Based Clustering
3. Distribution Model-Based Clustering
4. Hierarchical Clustering
5. Fuzzy Clustering

1. Partitioning Clustering

It is a type of clustering that divides the data into non-hierarchical groups. It is also known as the centroid-based method. The most common example of partitioning clustering is the K-Means Clustering algorithm.

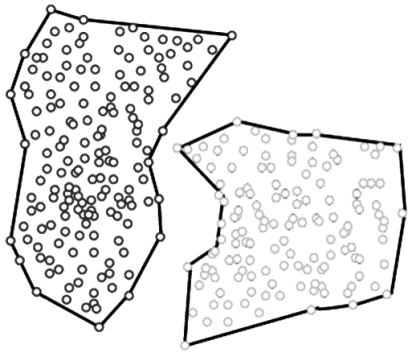
In this type, the dataset is divided into a set of k groups, where K is used to define the number of pre-defined groups. The cluster center is created in such a way that the distance between the data points of one cluster is minimum as compared to another cluster centroid.



2. Density-Based Clustering

The density-based clustering method connects the highly-dense areas into clusters, and the arbitrarily shaped distributions are formed as long as the dense region can be connected. This algorithm does it by identifying different clusters in the dataset and connects the areas of high densities into clusters. The dense areas in data space are divided from each other by sparser areas.

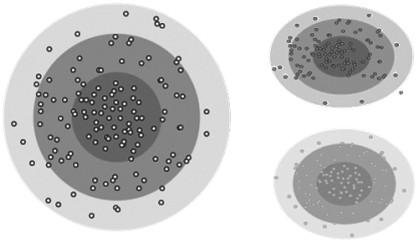
These algorithms can face difficulty in clustering the data points if the dataset has varying densities and high dimensions.



3. Distribution Model-Based Clustering

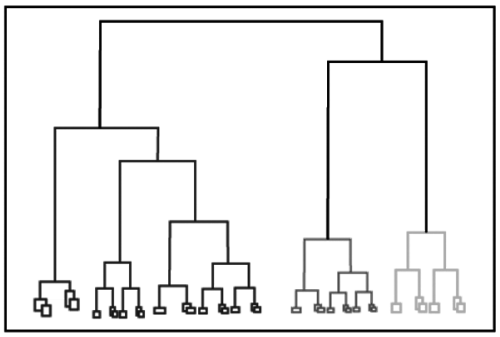
In the distribution model-based clustering method, the data is divided based on the probability of how a dataset belongs to a particular distribution. The grouping is done by assuming some distributions commonly Gaussian Distribution.

The example of this type is the Expectation-Maximization Clustering algorithm that uses Gaussian Mixture Models (GMM).



4. Hierarchical Clustering

Hierarchical clustering can be used as an alternative for the partitioned clustering as there is no requirement of pre-specifying the number of clusters to be created. In this technique, the dataset is divided into clusters to create a tree-like structure, which is also called a dendrogram. The observations or any number of clusters can be selected by cutting the tree at the correct level. The most common example of this method is the Agglomerative Hierarchical algorithm.



5. Fuzzy Clustering

Fuzzy clustering is a type of soft method in which a data object may belong to more than one group or cluster. Each dataset has a set of membership coefficients, which depend on the degree of membership to be in a cluster. Fuzzy C-means algorithm is the example of this type of clustering; it is sometimes also known as the Fuzzy k-means algorithm.

Q12. Expalin briefly about Clustering Algorithms.

Ans :

The Clustering algorithms can be divided based on their models that are explained above. There are different types of clustering algorithms published, but only a few are commonly used. The clustering algorithm is based on the kind of data that we are using. Such as, some algorithms need to guess the number of clusters in the given dataset, whereas some are required to find the minimum distance between the observation of the dataset.

Here we are discussing mainly popular Clustering algorithms that are widely used in machine learning:

1. K-Means algorithm

The k-means algorithm is one of the most popular clustering algorithms. It classifies the dataset by dividing the samples into different clusters of equal variances. The number of clusters must be specified in this algorithm. It is fast with fewer computations required, with the linear complexity of O(n).

2. Mean-shift algorithm

Mean-shift algorithm tries to find the dense areas in the smooth density of data points. It is an example of a centroid-based model, that works on updating the candidates for centroid to be the center of the points within a given region.

3. DBSCAN Algorithm

It stands for Density-Based Spatial Clustering of Applications with Noise. It is an example of a density-based model similar to the mean-shift, but with some remarkable advantages. In this algorithm, the areas of high density are separated by the areas of low density. Because of this, the clusters can be found in any arbitrary shape.

4. Expectation-Maximization Clustering using GMM

This algorithm can be used as an alternative for the k-means algorithm or for those cases where K-means can be failed. In GMM, it is assumed that the data points are Gaussian distributed.

5. Agglomerative Hierarchical algorithm

The Agglomerative hierarchical algorithm performs the bottom-up hierarchical clustering. In this, each data point is treated as a single cluster at the outset and then successively merged. The cluster hierarchy can be represented as a tree-structure.

6. Affinity Propagation

It is different from other clustering algorithms as it does not require to specify the number of clusters. In this, each data point sends a message between the pair of data points until convergence. It has $O(N^2T)$ time complexity, which is the main drawback of this algorithm.

4.1.8 Support Vector Machines

Q13. What are SVMs? Explain, the working mechanism of SVM.

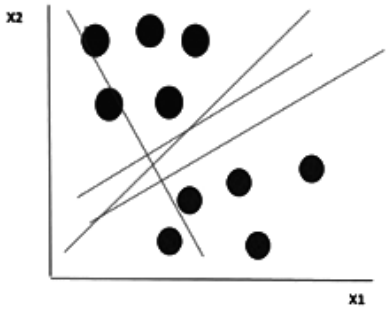
Ans :

Meaning

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification. The main objective of the SVM algorithm is to find the optimal hyperplane

in an N-dimensional space that can separate the data points in different classes in the feature space. The hyperplane tries that the margin between the closest points of different classes should be as maximum as possible. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let's consider two independent variables x_1, x_2, and one dependent variable which is either a blue circle or a red circle.

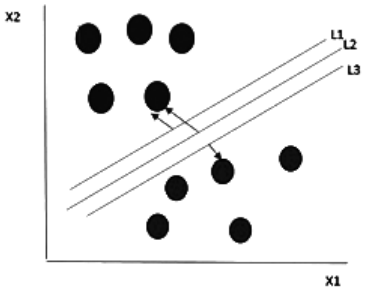


Linearly Separable Data Points

From the figure above it's very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x_1, x_2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

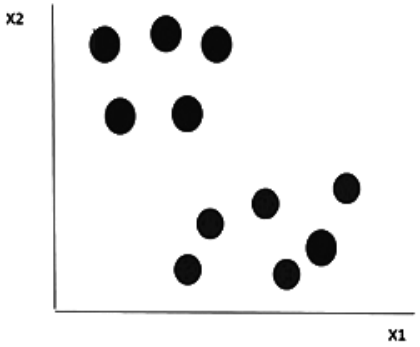
Working Mechanism

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.



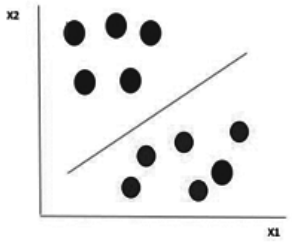
Multiple hyperplanes separate the data from two classes

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from the above figure, we choose L2. Let's consider a scenario like shown below



Selecting hyperplane for data with outlier

Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

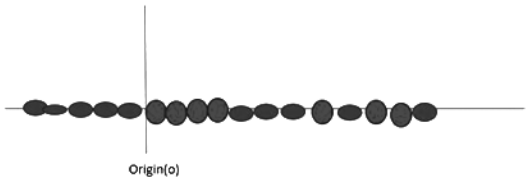


Hyperplane which is the most optimized one

So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called soft margins. When there is a soft margin to the data set, the SVM tries to minimize (1/ margin+'"("penalty)). Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

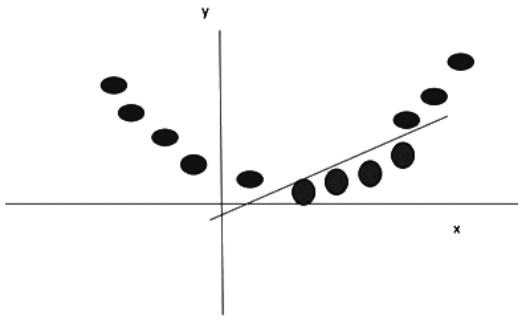
Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable

by a straight line/linear line). What to do if data are not linearly separable?



Original 1D dataset for classification

Say, our data is shown in the figure above. SVM solves this by creating a new variable using a kernel. We call a point x_i on the line and we create a new variable y_i as a function of distance from origin o.so if we plot this we get something like as shown below



Mapping 1D data to 2D to become able to separate the two classes

In this case, the new variable y is created as a function of distance from the origin. A non-linear function that creates a new variable is referred to as a kernel.

Mathematical intuition of Support Vector Machine

Consider a binary classification problem with two classes, labeled as +1 and -1. We have a training dataset consisting of input feature vectors X and their corresponding class labels Y.

The equation for the linear hyperplane can be written as:

$$w^x x + b = 0$$

The vector W represents the normal vector to the hyperplane. i.e the direction perpendicular to the hyperplane. The parameter b in the equation represents the offset or distance of the hyperplane from the origin along the normal vector w.

The distance between a data point x_i and the decision boundary can be calculated as:

$$d_i = \frac{w^x x_i + b}{||w||}$$

where $||w||$ represents the Euclidean norm of the weight vector w. Euclidean norm of the normal vector W

For Linear SVM classifier :

$$\hat{y} = \begin{cases} 1 : w^x x + b \geq 0 \\ 0 : w^x x + b < 0 \end{cases}$$

Optimization

For Hard margin linear SVM classifier

$$\underset{w,b}{\text{minimum}} \frac{1}{2} wT_w = \underset{w,b}{\text{minimum}} \frac{1}{2} ||w||^2$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1 \text{ for } i = 1, 2, 3, \ldots, m$$

The target variable or label for the i^{th} training instance is denoted by the symbol t_i in this statement. And t_i = -1 for negative occurrences (when y_i = 0) and t_i = 1positive instances (when $y_i = 1$) respectively. Because we require the decision boundary that satisfy the constraint:

For Soft margin linear SVM Classifier

$$\underset{w,b}{\text{minimum}} \frac{1}{2} wT_w = C\Sigma_{i=1}^{m} \zeta_i$$
$$\text{subject to } y_i(w^T x_i + b) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0 \text{ for } i = 1, 2, 3, \ldots, m$$

Dual Problem

A dual Problem of the optimisation problem that requires locating the Lagrange multipliers related to the support vectors can be used to solve SVM. The optimal Lagrange multipliers á(i) that maximize the following dual objective function

$$\underset{\alpha}{\text{maximize}} \frac{1}{2} \sum_{i \to m} \sum_{j \to m} \alpha_i \alpha_j t_i t_j K(x_i, x_j) - \sum_{i \to m} \alpha_i$$

Where,

α_i is the Lagrange multiplier associated with the ith training sample.

$K(x_i, x_j)$ is the kernel function that computes the similarity between two samples x_i and x_j. It allows SVM to handle nonlinear classification problems by implicitly mapping the samples into a higher-dimensional feature space.

The term $\Sigma\alpha_i$ represents the sum of all Lagrange multipliers.

The SVM decision boundary can be described in terms of these optimal Lagrange multipliers and the support vectors once the dual issue has been solved and the optimal Lagrange multipliers have been discovered. The training samples that have i > 0 are the support vectors, while the decision boundary is supplied by:

$$w = \sum_{i \to m} \alpha_i t_i K(x_i, x) + b$$
$$t_i(w^T x_i - b) = 1 \Leftrightarrow b - w^T x_i - t_i$$

4.2 ARTIFICIAL NEURAL NETWORKS

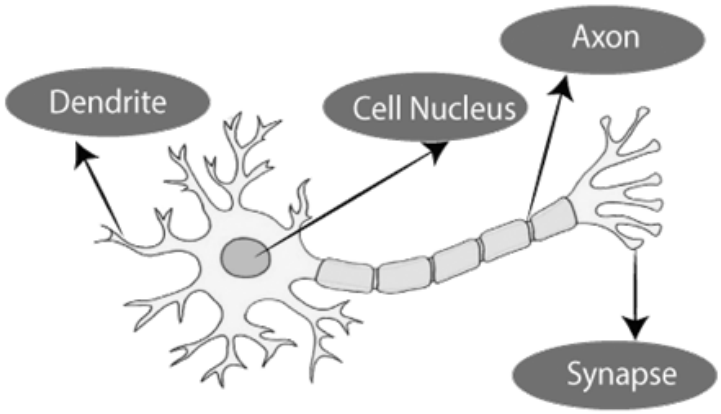
4.2.1 Introduction

Q14. Explain briefly about Artificial Neural Network?

Ans :

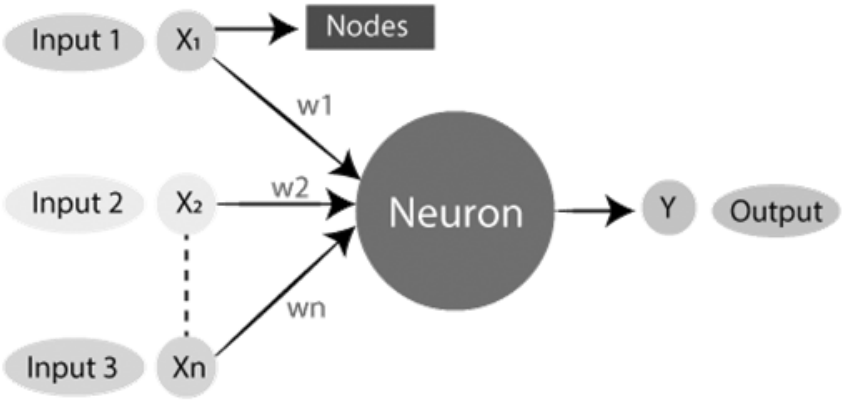
i) The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain.

ii) Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



The given figure illustrates the typical diagram of Biological Neural Network.

The typical Artificial Neural Network looks something like the given figure.



iii) Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

iv) An Artificial Neural Network in the field of Artificial intelligence where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

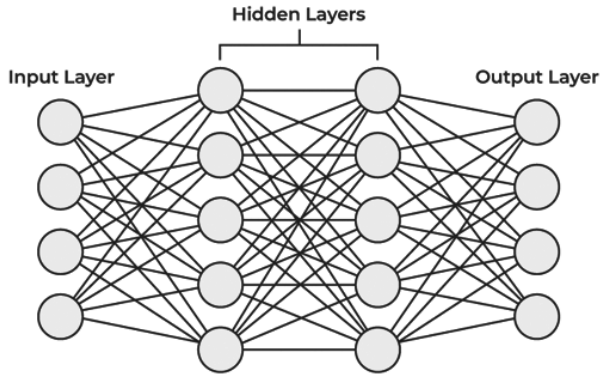
v) There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

vi) Artificial Neural Networks contain artificial neurons which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset.

vii) Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer.

viii) Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

ix) In the majority of neural networks, units are interconnected from one layer to another. Each of these connections has weights that determine the influence of one unit on another unit. As the data transfers from one unit to another, the neural network learns more and more about the data which eventually results in an output from the output layer.



x) The structures and operations of human neurons serve as the basis for artificial neural networks. It is also known as neural networks or neural nets.

xi) The input layer of an artificial neural network is the first layer, and it receives input from external sources and releases it to the hidden layer, which is the second layer. In the hidden layer, each neuron receives input from the previous layer neurons, computes the weighted sum, and sends it to the neurons in the next layer.

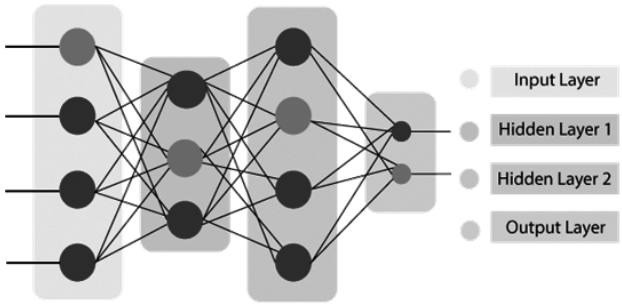
xii) These connections are weighted means effects of the inputs from the previous layer are optimized more or less by assigning different-different weights to each input and it is adjusted during the training process by optimizing these weights for improved model performance.

4.2.2 Artificial Neural Networks

Q15. Explain the architecture of Artificial Neural Network.

Ans :

Artificial Neural Network primarily consists of three layers:



i) Input Layer

As the name suggests, it accepts inputs in several different formats provided by the programmer.

ii) Hidden Layer

The hidden layer presents in-between input and output layers. It performs all the calculations to find hidden features and patterns.

iii) Output Layer

The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer.

The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi + xi + b$$

It determines weighted total is passed as an input to an activation function to produce the output. Activation functions choose whether a node should fire or not. Only those who are fired make it to the output layer. There are distinctive activation functions available that can be applied upon the sort of task we are performing.

Q16. Explain the advantages and disadvantags of Artificial Neural Network.

Ans:

Advantages of Artificial Neural Network (ANN)

i) Parallel Processing Capability

Artificial neural networks have a numerical value that can perform more than one task simultaneously.

ii) Storing Data on the Entire Network

Data that is used in traditional programming is stored on the whole network, not on a database. The disappearance of a couple of pieces of data in one place doesn't prevent the network from working.

iii) Capability to work with Incomplete Knowledge

After ANN training, the information may produce output even with inadequate data. The loss of performance here relies upon the significance of missing data.

iv) Having a Memory Distribution

For ANN is to be able to adapt, it is important to determine the examples and to encourage the network according to the desired output by demonstrating these examples to the network. The succession of the network is directly proportional to the chosen instances, and if the event can't appear to the network in all its aspects, it can produce false output.

v) Having Fault toLerance

Extortion of one or more cells of ANN does not prohibit it from generating output, and this feature makes the network fault-tolerance.

Disadvantages

i) Assurance of Proper Network Structure

There is no particular guideline for determining the structure of artificial neural networks. The appropriate network structure is accomplished through experience, trial, and error.

ii) Unrecognized behavior of the Network

It is the most significant issue of ANN. When ANN produces a testing solution, it does not provide insight concerning why and how. It decreases trust in the network.

iii) Hardware Dependence

Artificial neural networks need processors with parallel processing power, as per their structure. Therefore, the realization of the equipment is dependent.

iv) Difficulty of showing the issue to the Network

ANNs can work with numerical data. Problems must be converted into numerical values before being introduced to ANN. The presentation mechanism to be resolved here will directly impact the performance of the network. It relies on the user's abilities.

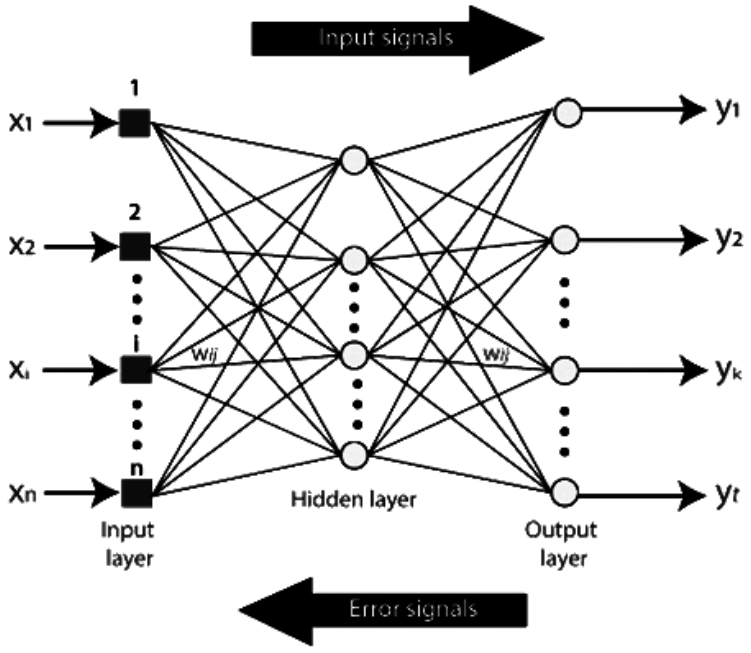
v) The Duration of the Network is Unknown

The network is reduced to a specific value of the error, and this value does not give us optimum results.

Q17. Explain the working mechanism of ANN. How do artificial neural networks work?

Ans :

Artificial Neural Network can be best represented as a weighted directed graph, where the artificial neurons form the nodes. The association between the neurons outputs and neuron inputs can be viewed as the directed edges with weights. The Artificial Neural Network receives the input signal from the external source in the form of a pattern and image in the form of a vector. These inputs are then mathematically assigned by the notations x(n) for every n number of inputs.



Afterward, each of the input is multiplied by its corresponding weights (these weights are the details utilized by the artificial neural networks to solve a specific problem). In general terms, these weights normally represent the strength of the interconnection between neurons inside the artificial neural network. All the weighted inputs are summarized inside the computing unit.

If the weighted sum is equal to zero, then bias is added to make the output non-zero or something else to scale up to the system's response. Bias has the same input, and weight equals to 1. Here the total of weighted inputs can be in the range of 0 to positive infinity. Here, to keep the response in the limits of the desired value, a certain maximum value is benchmarked, and the total of weighted inputs is passed through the activation function.

The activation function refers to the set of transfer functions used to achieve the desired output. There is a different kind of the activation function, but primarily either linear or non-linear sets of functions. Some of the commonly used sets of activation functions are the Binary, linear, and Tan hyperbolic sigmoidal activation functions. Let us take a look at each of them in details:

i) Binary

In binary activation function, the output is either a one or a 0. Here, to accomplish this, there is a threshold value set up. If the net weighted input of neurons is more than 1, then the final output of the activation function is returned as one or else the output is returned as 0.

ii) Sigmoidal Hyperbolic

The Sigmoidal Hyperbola function is generally seen as an "S" shaped curve. Here the tan hyperbolic function is used to approximate output from the actual net input. The function is defined as:

F(x) = (1/1 + exp(-????x))

Where is considered the Steepness parameter.

Types of Artificial Neural Network

There are various types of Artificial Neural Networks (ANN) depending upon the human brain neuron and network functions, an artificial neural network similarly performs tasks. The majority of the artificial neural networks will have some similarities with a more complex biological partner and are very effective at their expected tasks.

i) Feedback ANN

In this type of ANN, the output returns into the network to accomplish the best-evolved results internally. As per the University of Massachusetts, Lowell Centre for Atmospheric Research. The feedback networks feed information back into itself and are well suited to solve optimization issues. The Internal system error corrections utilize feedback ANNs.

ii) Feed-Forward ANN

A feed-forward network is a basic neural network comprising of an input layer, an output layer, and at least one layer of a neuron. Through assessment of its output by reviewing its input, the intensity of the network can be noticed based on group behavior of the associated neurons, and the output is decided. The primary advantage of this network is that it figures out how to evaluate and recognize input patterns.

4.2.3 Single-layer Feed-forward Networks

Q18. What is a feed forward neural network? Explain the working mechanism of feed forward neural network.

Ans :

Meaning

Feed forward neural networks are artificial neural networks in which nodes do not form loops. This type of neural network is also known as a multi-layer neural network as all information is only passed forward.

During data flow, input nodes receive data, which travel through hidden layers, and exit output nodes. No links exist in the network that could get used to by sending information back from the output node.

A feed forward neural network approximates functions in the following way:

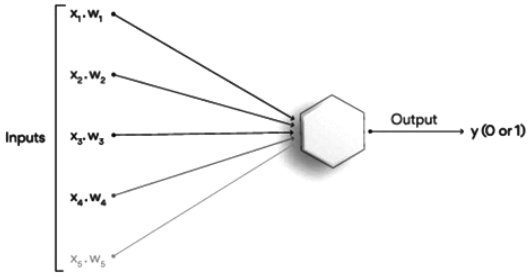
i) An algorithm calculates classifiers by using the formula y = f* (x).

ii) Input x is therefore assigned to category y.

iii) According to the feed forward model, y = f (x; θ). This value determines the closest approximation of the function.

Feed forward neural networks serve as the basis for object detection in photos, as shown in the Google Photos app.

Working principle of a feed forward neural network?



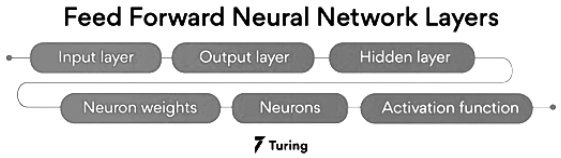
When the feed forward neural network gets simplified, it can appear as a single layer perceptron.

This model multiplies inputs with weights as they enter the layer. Afterward, the weighted input values get added together to get the sum. As long as the sum of the values rises above a certain threshold, set at zero, the output value is usually 1, while if it falls below the threshold, it is usually -1.

As a feed forward neural network model, the single-layer perceptron often gets used for classification. Machine learning can also get integrated into single-layer perceptrons. Through training, neural networks can adjust their weights based on a property called the delta rule, which helps them compare their outputs with the intended values.

As a result of training and learning, gradient descent occurs. Similarly, multi-layered perceptrons update their weights. But, this process gets known as back-propagation. If this is the case, the network's hidden layers will get adjusted according to the output values produced by the final layer.

Layers of feed forward neural network



Input layer

The neurons of this layer receive input and pass it on to the other layers of the network. Feature or attribute numbers in the dataset must match the number of neurons in the input layer.

Output layer

According to the type of model getting built, this layer represents the forecasted feature.

Hidden layer

Input and output layers get separated by hidden layers. Depending on the type of model, there may be several hidden layers.

There are several neurons in hidden layers that transform the input before actually transferring it to the next layer. This network gets constantly updated with weights in order to make it easier to predict.

Neuron weights

Neurons get connected by a weight, which measures their strength or magnitude. Similar to linear regression coefficients, input weights can also get compared.

Weight is normally between 0 and 1, with a value between 0 and 1.

Neurons

Artificial neurons get used in feed forward networks, which later get adapted from biological neurons. A neural network consists of artificial neurons.

Neurons function in two ways: first, they create weighted input sums, and second, they activate the sums to make them normal.

Activation functions can either be linear or nonlinear. Neurons have weights based on their inputs. During the learning phase, the network studies these weights.

Activation Function

Neurons are responsible for making decisions in this area.

According to the activation function, the neurons determine whether to make a linear or nonlinear decision. Since it passes through so many layers, it prevents the cascading effect from increasing neuron outputs.

An activation function can be classified into three major categories: sigmoid, Tanh, and Rectified Linear Unit (ReLu).

Sigmoid

Input values between 0 and 1 get mapped to the output values.

Tanh

A value between -1 and 1 gets mapped to the input values.

Rectified linear Unit

Only positive values are allowed to flow through this function. Negative values get mapped to 0.

Function in Feed Forward Neural Network

Cost function

In a feed forward neural network, the cost function plays an important role. The categorized data points are little affected by minor adjustments to weights and biases.

Thus, a smooth cost function can get used to determine a method of adjusting weights and biases to improve performance.

Following is a definition of the mean square error cost function:

$$C(w, b) = \frac{1}{2n} \sum_{n} ||\, y(x) - a \,||^2$$

Where,

w = the weights gathered in the network

b = biases

n = number of inputs for training

a = output vectors

x = input

v = vector v's normal length

Loss Function

The loss function of a neural network gets used to determine if an adjustment needs to be made in the learning process.

Neurons in the output layer are equal to the number of classes. Showing the differences between predicted and actual probability distributions. Following is the cross-entropy loss for binary classification.

$$L(\theta) \begin{cases} -\log(\hat{y}) & \text{if } y = 1 \\ -\log(1 - \hat{y}) & \text{if } y = 0 \end{cases}$$

As a result of multiclass categorization, a cross-entropy loss occurs:

$$L(\theta) = \sum_{i=1}^{k} y_i \log(\hat{y}_i)$$

Gradient Learning Algorithm

In the gradient descent algorithm, the next point gets calculated by scaling the gradient at the current position by a learning rate. Then subtracted from the current position by the achieved value.

To decrease the function, it subtracts the value (to increase, it would add). As an example, here is how to write this procedure:

$$P_{n+1} = P_n \; \eta \nabla f(p_n)$$

The gradient gets adjusted by the parameter ç, which also determines the step size. Performance is significantly affected by the learning rate in machine learning.

Output units

In the output layer, output units are those units that provide the desired output or prediction, thereby fulfilling the task that the neural network needs to complete.

There is a close relationship between the choice of output units and the cost function. Any unit that can serve as a hidden unit can also serve as an output unit in a neural network.

Q19. Explain the Advantages of Feed Forward Neural Networks.

Ans :

1. Machine learning can be boosted with feed forward neural networks' simplified architecture.
2. Multi-network in the feed forward networks operate independently, with a moderated intermediary.
3. Complex tasks need several neurons in the network.
4. Neural networks can handle and process nonlinear data easily compared to perceptrons and sigmoid neurons, which are otherwise complex.
5. A neural network deals with the complicated problem of decision boundaries.
6. Depending on the data, the neural network architecture can vary. For example, convolutional neural networks (CNNs) perform exceptionally well in image processing, whereas recurrent neural networks (RNNs) perform well in text and voice processing.
7. Neural networks need graphics processing units (GPUs) to handle large datasets for massive computational and hardware performance. Several GPUs get used widely in the market, including Kaggle Notebooks and Google Collab Notebooks.

Q20. Write the architecture and applications of feed forward network.

Ans :

The Architecture of the Network

In a network, the architecture refers to the number of hidden layers and units in each layer that make up the network.

A feed forward network based on the Universal Approximation Theorem must have a "squashing" activation function at least on one hidden layer.

The network can approximate any Borel measurable function within a finite-dimensional space with at least some amount of non-zero error when there are enough hidden units.

It simply states that we can always represent any function using the multi-layer perceptron (MLP), regardless of what function we try to learn.

Thus, we now know there will always be an MLP to solve our problem, but there is no specific method for finding it.

It is impossible to say whether it will be possible to solve the given problem if we use N layers with M hidden units.

Research is still ongoing, and for now, the only way to determine this configuration is by experimenting with it.

While it is challenging to find the appropriate architecture, we need to try many configurations before finding the one that can represent the target function.

There are two possible explanations for this. Firstly, the optimization algorithm may not find the correct parameters, and secondly, the training algorithms may use the wrong function because of overfitting.

Applications

There are many applications for these neural networks. The following are a few of them.

1. **Physiological feed forward system**

   It is possible to identify feed forward management in this situation because the central involuntary regulates the heartbeat before exercise.

2. **Gene regulation and feed forward**

   Detecting non-temporary changes to the atmosphere is a function of this motif as a feed forward system. You can find the majority of this pattern in the illustrious networks.

3. **Automation and machine management**

   Automation control using feed forward is one of the disciplines in automation.

4. **Parallel feed forward compensation with derivative**

   An open-loop transfer converts non-minimum part systems into minimum part systems using this technique.

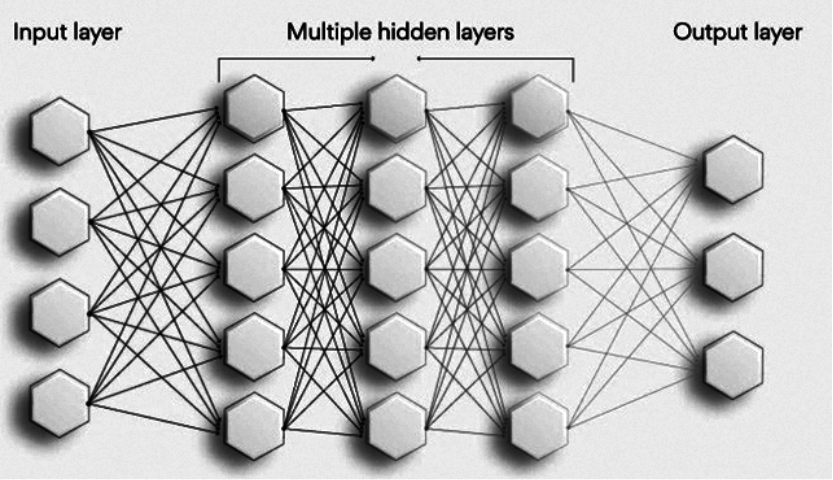
5. **Understanding the math behind neural networks**

   Typical deep learning algorithms are neural networks (NNs). As a result of their unique structure, their popularity results from their 'deep' understanding of data.

Furthermore, NNs are flexible in terms of complexity and structure. Despite all the advanced stuff, they can't work without the basic elements: they may work better with the advanced stuff, but the underlying structure remains the same.

Let's begin. NNs get constructed similarly to our biological neurons, and they resemble the following:

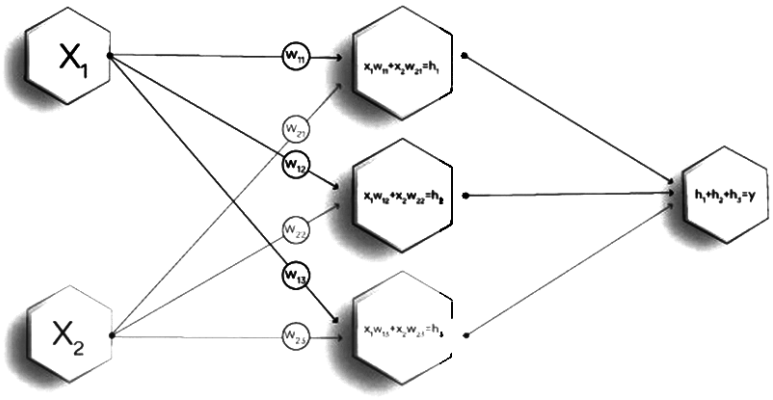
Deep Neural Network



Neurons are hexagons in this image. In neural networks, neurons get arranged into layers: input is the first layer, and output is the last with the hidden layer in the middle.

NN consists of two main elements that compute mathematical operations. Neurons calculate weighted sums using input data and synaptic weights since neural networks are just mathematical computations based on synaptic links.

The following is a simplified visualization:



In a matrix format, it looks as follows:

$$[X_1\ X_2]\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} = \begin{bmatrix} X_1w_{11} + X_2w_{21} \\ X_1w_{12} + X_2w_{22} \\ X_1w_{13} + X_2w_{23} \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

In the third step, a vector of ones gets multiplied by the output of our hidden layer:

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}' \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = h_1 + h_2 + h_3 = y$$

Using the output value, we can calculate the result. Understanding these fundamental concepts will make building NN much easier, and you will be amazed at how quickly you can do it. Every layer's output becomes the following layer's input.

Q21. Explain the backpropagation in feed forward neural network?

Ans :

Backpropagation is a technique based on gradient descent. Each stage of a gradient descent process involves iteratively moving a function in the opposite direction of its gradient (the slope).

The goal is to reduce the cost function given the training data while learning a neural network. Network weights and biases of all neurons in each layer determine the cost function. Backpropagation gets used to calculate the gradient of the cost function iteratively. And then update weights and biases in the opposite direction to reduce the gradient.

We must define the error of the backpropagation formula to specify i-th neuron in the l-th layer of a network for the j-th training. Example as follows (in $Z^{[i](j)}_i$ which represents the weighted input to the neuron, and L represents the loss.)

$$\delta_i^{[i](j)} = \frac{\partial L(\hat{y}^{(j)}, y^{(j)})}{\partial z_i^{[i](j)}}$$

In backpropagation formulas, the error is defined as above:

Below is the full derivation of the formulas. For each formula below, L stands for the output layer, g for the activation function, " the gradient, W[l]T layer l weights transposed.

A proportional activation of neuron i at layer l based on bli bias from layer i to layer i, wlik weight from layer l to layer l-1, and akl"1 activation of neuron k at layer l-1 for training example j.

$$\delta^{[L](j)} = \nabla_{\hat{y}(j)} L \odot (g^{[L]})'(z^{[L](j)}) = \hat{y}^{(j)} - y^{(j)}$$

$$\delta^{(l)(j)} = W^{[l+1]^x} \delta[l+1](j) \odot (g^{[l]})'(z^{[l](j)})$$

$$\frac{\partial L}{\partial b_i^{[l]}} = \delta_i^{[l](j0}$$

$$\frac{\partial L}{\partial w_{ik}^{[l]}} = \delta_i^{[l](j)} \alpha_k^{[l-1](j)}$$

The first equation shows how to calculate the error at the output layer for sample j. Following that, we can use the second equation to calculate the error in the layer just before the output layer.

Based on the error values for the next layer, the second equation can calculate the error in any layer. Because this algorithm calculates errors backward, it is known as backpropagation.

For sample j, we calculate the gradient of the loss function by taking the third and fourth equations and dividing them by the biases and weights. We can update biases and weights by averaging gradients of the loss function relative to biases and weights for all samples using the average gradients.

The process is known as batch gradient descent. We will have to wait a long time if we have too many samples. If each sample has a gradient, it is possible to update the biases/weights accordingly. The process is known as stochastic gradient descent.

Even though this algorithm is faster than batch gradient descent, it does not yield a good estimate of the gradient calculated using a single sample.

It is possible to update biases and weights based on the average gradients of batches. It gets referred to as mini-batch gradient descent and gets preferred over the other two.

4.2.4 Multi-layer Feed-forward Networks

Q22. Explain about Multi Layer Feed Forward Networks.

Ans :

Meaning

Multilayer Feed-Forward Neural Network(MFFNN) is an interconnected Artificial Neural Network with multiple layers that has neurons with weights associated with them and they compute the result using activation functions. It is one of the types of Neural Networks in which the flow of the network is from input to output units and it does not have any loops, no feedback, and no signal moves in backward directions that is from output to hidden and input layer.

The ANN is a self-learning network that learns from sample data sets and signals, it is based on the function of the biological nervous system. The type of activation function depends on the desired output. It is a part of machine learning and AI, which are the fastest-growing fields, and lots of research is going on to make it more effective.

The Architecture of the Multilayer Feed-Forward Neural Network:

This Neural Network or Artificial Neural Network has multiple hidden layers that make it a multilayer neural Network and it is feed-forward because it is a network that follows a top-down approach to train the network. In this network there are the following layers:

1. Input Layer

It is starting layer of the network that has a weight associated with the signals.

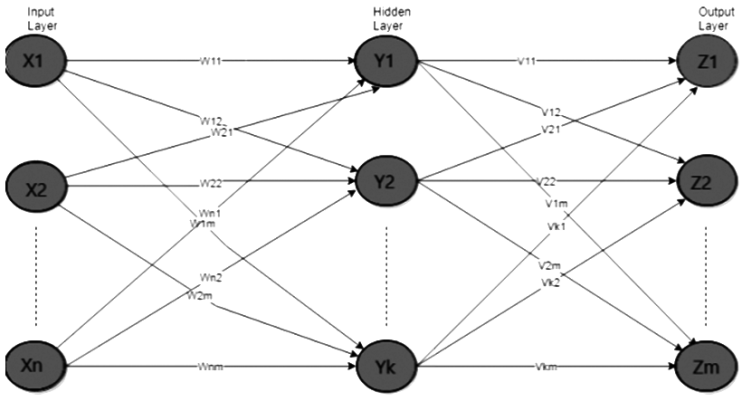
2. Hidden Layer

This layer lies after the input layer and contains multiple neurons that perform all computations and pass the result to the output unit.

3. Output Layer

It is a layer that contains output units or neurons and receives processed data from the hidden layer, if there are further hidden layers connected to it then it passes the weighted unit to the connected hidden layer for further processing to get the desired result.

The input and hidden layers use sigmoid and linear activation functions whereas the output layer uses a Heaviside step activation function at nodes because it is a two-step activation function that helps in predicting results as per requirements. All units also known as neurons have weights and calculation at the hidden layer is the summation of the dot product of all weights and their signals and finally the sigmoid function of the calculated sum. Multiple hidden and output layer increases the accuracy of the output.



Application of Multilayer Feed-Forward Neural Network

1. Medical field
2. Speech regeneration
3. Data processing and compression
4. Image processing

Limitations

This ANN is a basic form of Neural Network that has no cycles and computes only in the forward direction. It has some limitations like sometimes information about the neighborhood is lost and in that case, it becomes difficult to process further all steps are needed to be performed again and it does not support back propagation so the network cannot learn or correct the fault of the previous stage.

4.2.5 Radial-Basis Function Networks

Q23. What are Radial Basis Functions?

Ans :

Meaning

Radial Basis Functions are a special class of feed-forward neural networks consisting of three layers: an input layer, a hidden layer, and the output layer. This is fundamentally different from most neural network architectures, which are composed of many layers and bring about nonlinearity by recurrently applying non-linear activation functions. The input layer receives input data and passes it into the hidden layer, where the computation occurs. The hidden layer of Radial Basis Functions Neural Network is the most powerful and very different from most Neural networks. The output layer is designated for prediction tasks like classification or regression.

Working Mechanism

RBF Neural networks are conceptually similar to K-Nearest Neighbor (k-NN) models, though the implementation of both models is starkly different. The fundamental idea of Radial Basis Functions is that an item's predicted target value is likely to be the same as other items with close values of predictor variables. An RBF Network places one or many RBF neurons in the space described by the predictor variables. The space has multiple dimensions corresponding to the number of predictor variables present. We calculate the Euclidean distance from the evaluated point to the center of each neuron. A Radial Basis Function (RBF), also known as kernel function, is applied to the distance to calculate every neuron's weight (influence). The name of the Radial Basis Function comes from the radius distance, which is the argument to the function. Weight = RBF[distance)The greater the distance of a neuron from the point being evaluated, the less influence (weight) it has.

Radial Basis Functions

A Radial Basis Function is a real-valued function, the value of which depends only on the distance from the origin. Although we use various types of radial basis functions, the Gaussian function is the most common.

In the instance of more than one predictor variable, the Radial basis Functions Neural Network has the same number of dimensions as there are variables. If three neurons are in a space with two predictor variables, we can predict the value from the RBF functions. We can calculate the best-predicted value for the new point by adding the output values of the RBF functions multiplied by the weights processed for each neuron.

The radial basis function for a neuron consists of a center and a radius (also called the spread). The radius may vary between different neurons. In DTREG-generated RBF networks, each dimension's radius can differ.

As the spread grows larger, neurons at a distance from a point have more influence.

RBF Network Architecture

The typical architecture of a radial basis functions neural network consists of an input layer, hidden layer, and summation layer.

Input Layer

The input layer consists of one neuron for every predictor variable. The input neurons pass the value to each neuron in the hidden layer. N-1 neurons are used for categorical values, where N denotes the number of categories. The range of values is standardized by subtracting the median and dividing by the interquartile range.

Hidden Layer

The hidden layer contains a variable number of neurons (the ideal number determined by the training process). Each neuron comprises a radial basis function centered on a point. The number of dimensions coincides with the number of predictor variables. The radius or spread of the RBF function may vary for each dimension.

When an x vector of input values is fed from the input layer, a hidden neuron calculates the Euclidean distance between the test case and the neuron's center point. It then applies the kernel function using the spread values. The resulting value gets fed into the summation layer.

Output Layer or Summation Layer

The value obtained from the hidden layer is multiplied by a weight related to the neuron and passed to the summation. Here the weighted values are added up, and the sum is presented as the network's output. Classification problems have one output per target category, the value being the probability that the case evaluated has that category.

The Input Vector

It is the n-dimensional vector that you're attempting to classify. The whole input vector is presented to each of the RBF neurons.

The RBF Neurons

Every RBF neuron stores a prototype vector (also known as the neuron's center) from amongst the vectors of the training set. An RBF neuron compares the input vector with its prototype, and outputs a value between 0 and 1 as a measure of similarity. If an input is the same as the prototype, the neuron's output will be 1. As the input and prototype difference grows, the output falls exponentially towards 0. The shape of the response by the RBF neuron is a bell curve. The response value is also called the activation value.

The Output Nodes

The network's output comprises a set of nodes for each category you're trying to classify. Each output node computes a score for the concerned category. Generally, we take a classification decision by assigning the input to the category with the highest score.

The score is calculated based on a weighted sum of the activation values from all RBF neurons. It usually gives a positive weight to the RBF neuron belonging to its category and a negative weight to others. Each output node has its own set of weights.

Example

Let us consider a fully trained Radial Basis Function Example.

A dataset has two-dimensional data points belonging to two separate classes. An RBF Network has been trained with 20 RBF neurons on the said data set. We can mark the prototypes selected and view the category one score on the input space. For viewing, we can draw a 3-D mesh or a contour plot.

The areas of highest and lowest category one score should be marked separately.

In the case of category one output node:

1. All the weights for category 2 RBF neurons will be negative.
2. All the weights for category 1 RBF neurons will be positive.

Finally, an approximation of the decision boundary can be plotted by computing the scores over a finite grid.

What are Radial Basis Functions Neural Networks? Everything You Need to Know

4.2.6 Design Issues of Artificial Neural Networks

Q24. Explain the Design Issues of Artificial Neural Networks.

Ans : **(Imp.)**

Artificial Neural Networks (ANNs) have proven to be powerful tools for a wide range of tasks, from image recognition to natural language processing. However, like any technology, they come with their own set of design issues and challenges. Here are some key design issues of artificial neural networks:

1. Architecture Design

i) **Topology**: Choosing the right network architecture, including the number of layers, types of layers (e.g., fully connected, convolutional, recurrent), and connections between nodes, is crucial. The architecture should be tailored to the specific task and dataset.

ii) **Overfitting and Underfitting**: Finding the right balance between model complexity and generalization is a common challenge. Overly complex models can overfit to the training data and perform poorly on new data, while overly simple models might underfit and lack the capacity to capture patterns in the data.

2. Hyperparameter Tuning

i) **Learning Rate**: The learning rate controls the step size in gradient descent optimization. Choosing an appropriate learning rate can greatly affect the convergence speed and final performance of the network.

ii) **Batch Size**: The number of samples used in each iteration of training can impact convergence speed and memory usage.

iii) **Regularization Parameters**: Techniques like L1/L2 regularization and dropout can help prevent overfitting, but selecting the right values is essential.

iv) **Activation Functions**: Choosing the appropriate activation functions for different layers affects the network's ability to learn complex patterns.

3. Data Preprocessing

i) **Normalization**: Scaling input data to a similar range helps gradient-based optimization methods converge more effectively.

ii) **Handling Imbalanced Data**: When classes are not evenly represented in the dataset, the network might perform poorly on minority classes. Techniques like oversampling, undersampling, or using different loss functions can help mitigate this issue.

4. Dataset Quality and Quantity

i) **Data Availability**: Neural networks require substantial amounts of labeled data to perform well. Lack of quality data can hinder performance and lead to biases.

ii) **Data Augmentation**: Generating new training samples from existing data through transformations can help improve the model's generalization.

5. **Computational Resources**:

i) **Hardware and Memory**: Larger and deeper networks require more computational resources and memory capacity, which can be limiting factors in some applications.

ii) **Training Time**: Training complex networks can be time-consuming, and it might not be feasible for real-time or time-sensitive applications.

6. **Interpretability and Explainability**

Black Box Nature: Neural networks are often considered black box models, meaning it's challenging to understand how they arrive at decisions. This lack of interpretability can be problematic in domains where explanations are crucial (e.g., healthcare, finance).

7. **Ethical and Bias Concerns**:

i) **Bias in Data**: If training data is biased, the network can perpetuate or even amplify those biases. Careful attention is needed to address potential ethical and fairness issues.

ii) **Fairness**: Ensuring that the network's decisions are fair and unbiased across different demographic groups is a critical concern.

8. **Transfer Learning and Domain Adaptation**:

Transferring Knowledge: Neural networks trained on one task or dataset might not generalize well to a different but related task or dataset. Techniques like transfer learning and domain adaptation attempt to address this issue.

9. **Gradient Vanishing and Exploding**:

i) **Vanishing Gradient**: In deep networks, gradients can become extremely small during backpropagation, leading to slow or stalled learning in earlier layers.

ii) **Exploding Gradient**: Conversely, gradients can become extremely large, causing training instability.

10. **Model Deployment and Maintenance**:

i) **Scalability**: Deploying neural networks in production environments requires considerations of resource availability and scalability.

ii) **Adaptability**: Models may degrade over time due to changing data distributions. Continuous monitoring and retraining are often necessary.

Addressing these design issues requires a combination of expertise, experimentation, and domain-specific knowledge. As the field of neural networks evolves, researchers and practitioners continually develop new techniques to mitigate these challenges and improve the overall performance and reliability of artificial neural networks.

4.2.7 Recurrent Networks

Q25. Explain briefly about Recurrent Neural Network (RNN)?

Ans : **(Imp.)**

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

Below is how you can convert a Feed-Forward Neural Network into a Recurrent Neural Network:

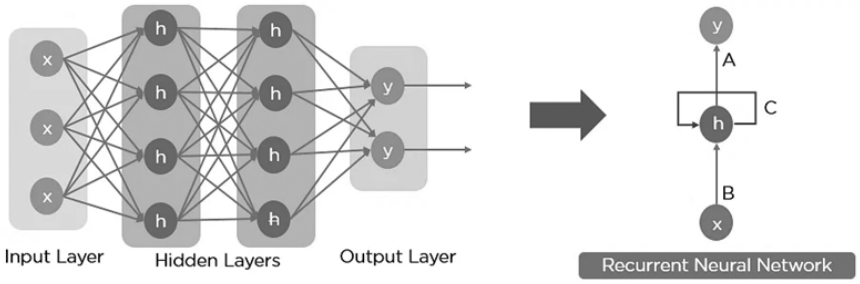


Fig: Simple Recurrent Neural Network

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.

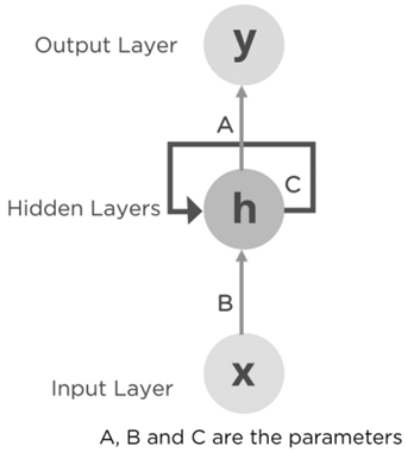


Fig: Fully connected Recurrent Neural Network

Here, "x" is the input layer, "h" is the hidden layer, and "y" is the output layer. A, B, and C are the network parameters used to improve the output of the model. At any given time t, the current input is a combination of input at x(t) and x(t-1). The output at any given time is fetched back to the network to improve on the output.

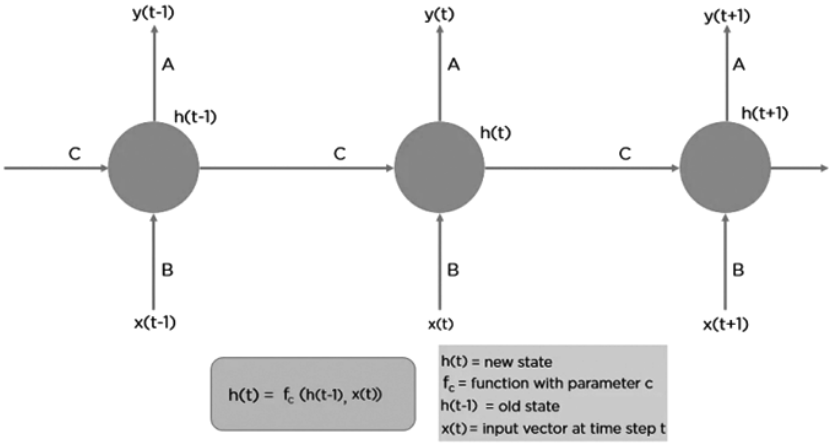


Fig: Fully connected Recurrent Neural Network

Now that you understand what a recurrent neural network is let's look at the different types of recurrent neural networks.

Why Recurrent Neural Networks?

RNN were created because there were a few issues in the feed-forward neural network:

1. Cannot handle sequential data
2. Considers only the current input
3. Cannot memorize previous inputs

The solution to these issues is the RNN. An RNN can handle sequential data, accepting the current input data, and previously received inputs. RNNs can memorize previous inputs due to their internal memory.

How Does Recurrent Neural Networks Work?

In Recurrent Neural networks, the information cycles through a loop to the middle hidden layer.

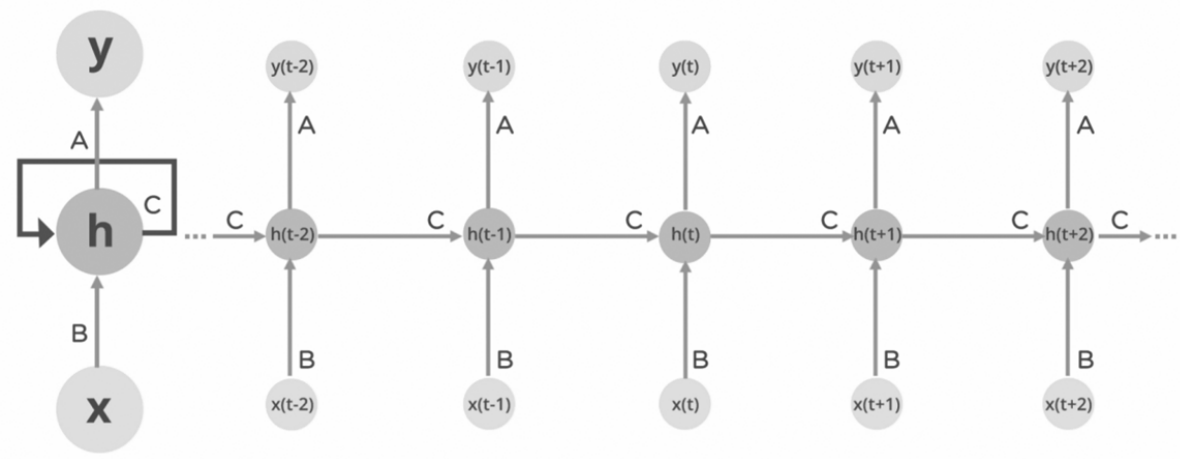


Fig: Working of Recurrent Neural Network

The input layer 'x' takes in the input to the neural network and processes it and passes it onto the middle layer.

The middle layer 'h' can consist of multiple hidden layers, each with its own activation functions and weights and biases. If you have a neural network where the various parameters of different hidden layers are not affected by the previous layer, ie: the neural network does not have memory, then you can use a recurrent neural network.

The Recurrent Neural Network will standardize the different activation functions and weights and biases so that each hidden layer has the same parameters. Then, instead of creating multiple hidden layers, it will create one and loop over it as many times as required.

Q26. Differentiate between Feed- Forward Neural Networks and Recurrent Neural Networks.

Ans :

A feed-forward neural network allows information to flow only in the forward direction, from the input nodes, through the hidden layers, and to the output nodes. There are no cycles or loops in the network.

Below is how a simplified presentation of a feed-forward neural network looks like:

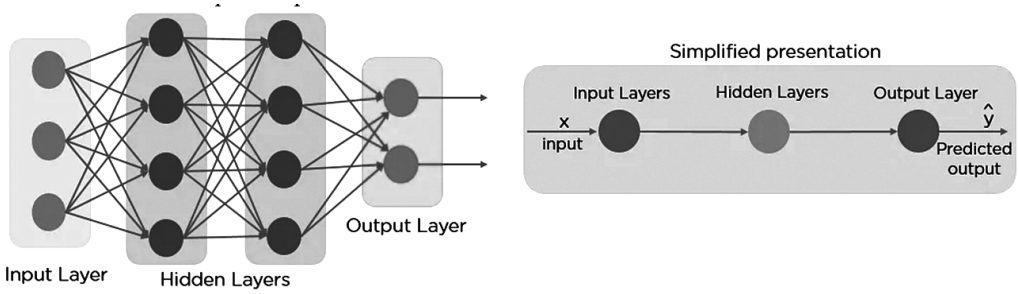


Fig: Feed-forward Neural Network

In a feed-forward neural network, the decisions are based on the current input. It doesn't memorize the past data, and there's no future scope. Feed-forward neural networks are used in general regression and classification problems.

Q27. Explain the application, advantages and disadvantages of Recurrent Neural Networks?

Ans :

Applications

1. Image Captioning

RNNs are used to caption an image by analyzing the activities present.

2. Time Series Prediction

Any time series problem, like predicting the prices of stocks in a particular month, can be solved using an RNN.

3. Natural Language Processing

Text mining and Sentiment analysis can be carried out using an RNN for Natural Language Processing (NLP).

4. Machine Translation

Given an input in one language, RNNs can be used to translate the input into different languages as output.

Advantages

Recurrent Neural Networks (RNNs) have several advantages over other types of neural networks, including:

1. Ability To Handle Variable-Length Sequences

RNNs are designed to handle input sequences of variable length, which makes them well-suited for tasks such as speech recognition, natural language processing, and time series analysis.

2. Memory of Past Inputs

RNNs have a memory of past inputs, which allows them to capture information about the context of the input sequence. This makes them useful for tasks such as language modeling, where the meaning of a word depends on the context in which it appears.

3. Parameter Sharing

RNNs share the same set of parameters across all time steps, which reduces the number of parameters that need to be learned and can lead to better generalization.

4. Non-Linear Mapping

RNNs use non-linear activation functions, which allows them to learn complex, non-linear mappings between inputs and outputs.

5. Sequential Processing

RNNs process input sequences sequentially, which makes them computationally efficient and easy to parallelize.

6. Flexibility

RNNs can be adapted to a wide range of tasks and input types, including text, speech, and image sequences.

7. Improved Accuracy

RNNs have been shown to achieve state-of-the-art performance on a variety of sequence modeling tasks, including language modeling, speech recognition, and machine translation.

These advantages make RNNs a powerful tool for sequence modeling and analysis, and have led to their widespread use in a variety of applications, including natural language processing, speech recognition, and time series analysis.

Disadvantages

Here are some of the main disadvantages of RNNs:

1. Vanishing And Exploding Gradients

RNNs can suffer from the problem of vanishing or exploding gradients, which can make it difficult to train the network effectively. This occurs when the gradients of the loss function with respect to the parameters become very small or very large as they propagate through time.

2. Computational Complexity

RNNs can be computationally expensive to train, especially when dealing with long sequences. This is because the network has to process each input in sequence, which can be slow.

3. Difficulty In Capturing Long-Term Dependencies

Although RNNs are designed to capture information about past inputs, they can struggle to capture long-term dependencies in the input sequence. This is because the gradients can become very small as they propagate through time, which can cause the network to forget important information.

4. Lack of Parallelism

RNNs are inherently sequential, which makes it difficult to parallelize the computation. This can limit the speed and scalability of the network.

5. Difficulty In Choosing The Right Architecture

There are many different variants of RNNs, each with its own advantages and disadvantages. Choosing the right architecture for a given task can be challenging, and may require extensive experimentation and tuning.

6. Difficulty In Interpreting The Output

The output of an RNN can be difficult to interpret, especially when dealing with complex inputs such as natural language or audio. This can make it difficult to understand how the network is making its predictions.

These disadvantages are important when deciding whether to use an RNN for a given task. However, many of these issues can be addressed through careful design and training of the network and through techniques such as regularization and attention mechanisms.

Q28. Explain about the types of Recurrent Neural Networks.

Ans :

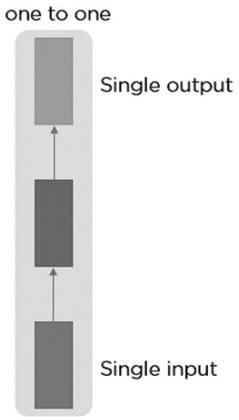
Types

There are four types of Recurrent Neural Networks:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

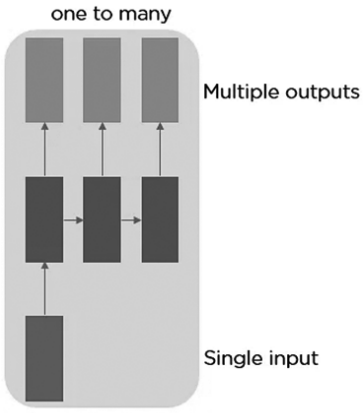
1. One to One RNN

This type of neural network is known as the Vanilla Neural Network. It's used for general machine learning problems, which has a single input and a single output.



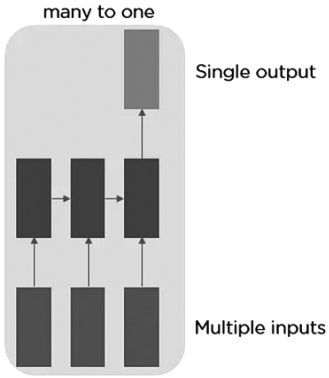
2. SOne to Many RNN

This type of neural network has a single input and multiple outputs. An example of this is the image caption.



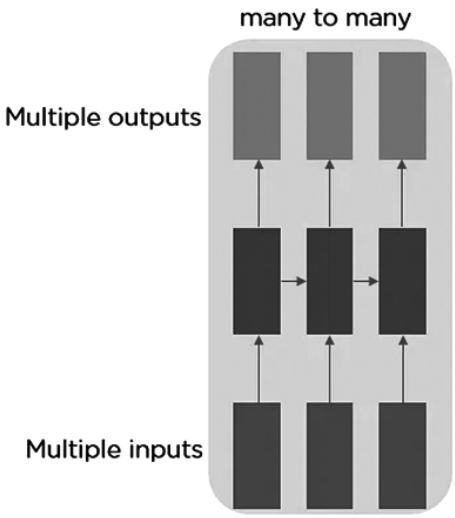
3. Many to One RNN

This RNN takes a sequence of inputs and generates a single output. Sentiment analysis is a good example of this kind of network where a given sentence can be classified as expressing positive or negative sentiments.



5. Many to Many RNN

This RNN takes a sequence of inputs and generates a sequence of outputs. Machine translation is one of the examples.



UNIT V

Advanced Knowledge Representation Techniques: Case Grammars, Semantic Web.

Natural Language Processing: Introduction, Sentence Analysis Phases, Grammars and Parsers, Types of Parsers, Semantic Analysis, Universal Networking Knowledge.

5.1 Advanced Knowledge Representation Techniques

5.1.1 Case Grammars

Q1. Explain various cases in case grammars.

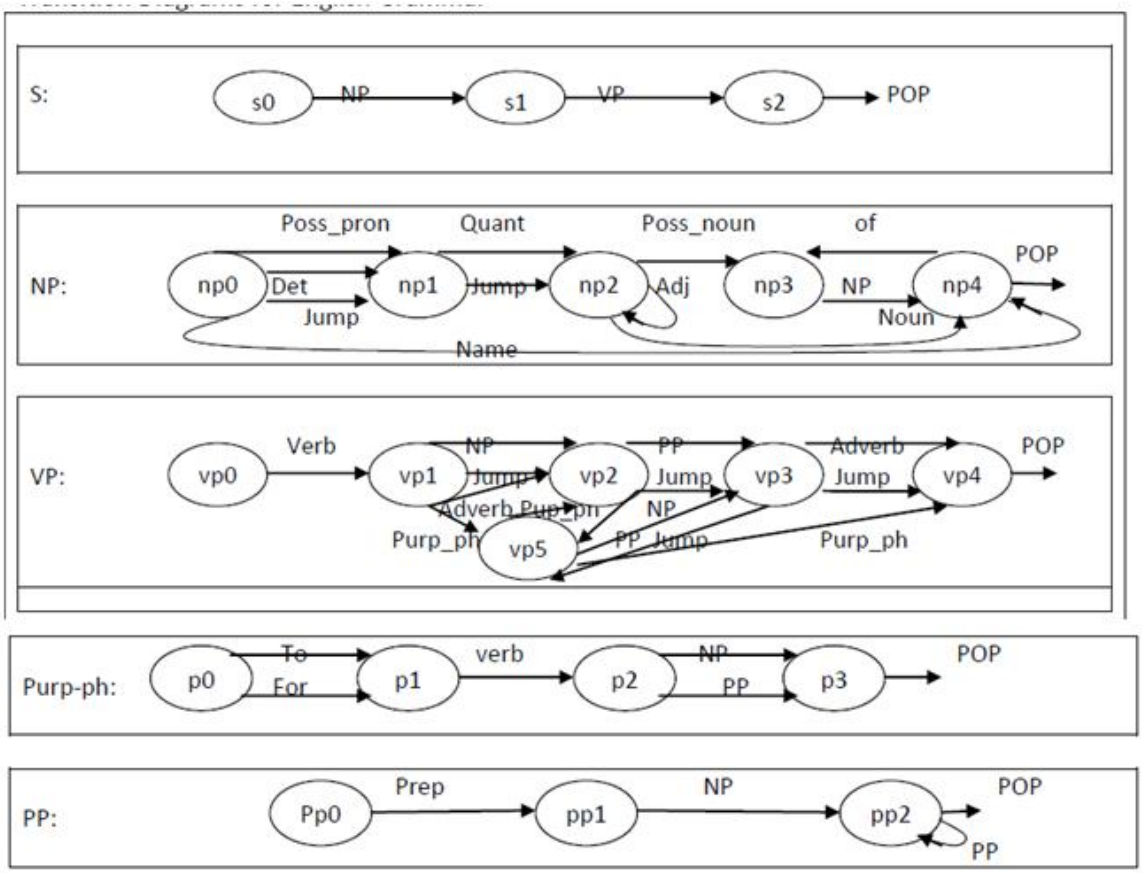
Ans : **(Imp.)**

Extract meaning of a sentence and express in the form of cases. Six Cases were proposed by Fillmore (1968).

They are

1. AGENTIVE (agent)
2. OBJECTIVE (object)
3. INSTRUMENTAL (instrument)
4. DATIVE (which covers experience)
5. FACTIVE (which covers an action)
6. LOCATIVE (Location of an action)

An example Case	Frame Case Frame
Cases	Values
Action	Give
Agent	John
Object	Apple
Beneficiary	Mike
Time	Past
Location	Kitchen



Cases finding Rules for English

Case	Rule	Example
Agent Case	1. For active sentence, if it has { < NP > < VP > } form, then noun of NP is an agent.	**John** gets food for Mary.
	2. For passive sentence, { < NP1 > < third form of verb > < NP2 > < by > < NP3 > }, then animate noun of NP3 is an agent.	Mike was given a book by **John**.
Object Case	1. If { < verb > < NP1 > < to/for > < verb > < NP2 > }, then non animate noun of NP2 is an object.	John went to market to buy **fruits**.
	2. If { < verb > < NP > < to/for > < verb > }, then non-animate noun of NP is an object.	John gave a **book** to read.
	3. If { < verb > < NP containing non animate noun > }, then non animate noun of NP is an object.	John gets **food** for Mary.
Beneficiary Case	1. If { < for > < NP with animate noun > }, then animate noun of NP is beneficiary.	John gets food **for Mary**.
	2. If { < from > < NP > }, then animate subject (noun) of a sentence is beneficiary.	**Mike** buys a book for John.
	3. If { < to > < NP with animate noun and verb is 'give' > }, then noun of NP is beneficiary.	John gives a book to **Mary**.

Source and Goal Location Case	1. If { < from > < NP with Non animate noun > }, then non animate noun of NP following 'from' is Source Location Case.	John is coming from **India**.
	2. If { < to > < NP with non animate noun > }, then non animate noun is NP is Goal Location Case.	John is coming to **India**.
	3. If { < to > < verb > }, then **To_purpose** Case is filled	He is going **to eat** an apple.
Location Case	1. If { < in > | < on > | < at > | < NP > }, then noun is NP fills Location Case.	Book is on the **table**.
Instrument Case	1. If { < with/by > < NP with non animate noun > }, then noun in NP fill Instrument Case.	John went to market by **car**.
Co-Agent Case	1. If { < with > < NP with animate noun > }, then animate noun of NP fills Co-Agent Case.	Mike is going with John.
	2. If { < NP with animate noun > and < NP with animate noun at subject place in the sentence }, then animate noun of NP followed by 'and' fills Co-Agent Case.	John and **Mary** are eating food.
Purpose Case	1. If { < for/ to > < verb > } then verb fills up Purpose Case.	John is buying fruits to **eat**.
Comparator Case	1. If { < than > < noun > }, then we get Comparator Case.	John is taller than **Mike**.

5.1.2 Semantic Web

Q2. Explain importance of Semantic web in AI?

Ans : **(Imp.)**

Semantic Web is an extension to the World Wide Web. The purpose of the semantic web is to provide structure to the web and data in general. It emphasizes on representing a web of data instead of web of documents. It allows computers to intelligently search, combine and process the web content based on the meaning that the content has. Three main models of the semantic web are:

1. Building models
2. Computing with Knowledge
3. Exchanging Information

1. Building Models

Model is a simplified version or description of certain aspects of the real-time entities. Model gathers information which is useful for the understanding of the particular domain.

2. Computing with Knowledge

Conclusions can be obtained from the knowledge present.

Example: If two sentences are given as *'John is the son of Harry'* and another sentence given is- *'Hary's father is Joey'*, then the knowledge that can be computed from it is – *'John is the grandson of Joey'*

Similarly, another example useful in the understanding of computing knowledge is-

'All A is B' and *'All B is C'*, then the conclusion that can be drawn from it is – *'All A are C'* respectively.

3. Exchanging Information

It is an important aspect. Various communication protocols have been implemented for the exchange of information like the TCP/IP, HTML, WWW. Web Services have also been used for the exchange of the data.

The technologies associated with the semantic web are:

i) RDF (Resource Description Framework)

ii) OWL (Web Ontology Language)

iii) DL (Description Language)

The query language used is:

i) SPARQL (SPARQL Protocol and RDF query language).

ii) SHACL (Shape Constraint Language). SHACL is used for validating the RDF graphs against a set of conditions.

The working of the Semantic Web

Here we discuss the working of the semantic web:

i) Since Tim Berners-conception Lee's of the web, it has been developing toward semantics. Instead of people manually looking through a restricted list of links, algorithms now search through a massive number of more organized material sets to answer or act on a given query.

ii) The addition of semantics, structure and meaningful, machine-interpretable linkages to data allows computers to more precisely retrieve and modify information on our behalf. This leads to better content discovery and searches experiences, as well as more chances for seamless data sharing, recombination, analysis, and reuse, with less human-manual-human contact in the loop.

iii) The use of those three technologies is one method to distinguish a Semantic Web application from any other application. The Semantic Web has been dubbed a variety of names, including Web 3.0 and the Linked Data Web. Even in terms of the technical stack, some of these names have a lot of weight.

iv) The "knowledge graph" is a modern embodiment of Semantic Web technology. The Semantic Web goal has been hampered throughout time due to a variety of factors, including misdirected applications, a lack of scale, and perceived complexity. The knowledge graph concept has arisen to assist developers and decision-makers in constraining the development and deployment of Semantic Web standards more strictly.

What is RDF in Semantic Web?

Resource Description Framework, as the name implies, aids in the description of any resource that has a unique identity. To put it another way, RDF assists us in defining data about other data, i.e. metadata.

RDF is made up of three parts: subject, predicate, and object. It's a remark regarding the subject's connection with the object. As a result, the location of Villa Nellcôte in France may be described as an RDF Triple. URIs, literals, and blank nodes are used to represent all three components of the triple. An RDF Graph is created when many of these assertions are combined. The graph's nodes are subjects and objects. The connecting arcs are formed by predicates.

RDF does not provide meaning to data on its own. RDF is a type of data paradigm that allows you to represent relationships. Vocabularies and ontologies are defined to provide meaning. These are usually expressed in terms of classes, their attributes, and their connections to other classes. For example, an RDF triple can convey that Paris is France's capital, but this makes little sense to a computer. Capital is a sort of city, city belongs to a nation, and country is a political entity, according to a dictionary. This aids the computer in grasping the context, yet it will never fully comprehend what people do.

Importance of Semantic Web

The Semantic Web's expansion and the tools it brings to the table are putting machines' analytical skills to work in the areas of content creation, management, learning, support, media, ecommerce, scientific research, knowledge management, and publishing in general. Knowledge will become meaningful anywhere we convey it.

Although SEO and SERP ranking may be sufficient reasons, content discovery and display on Google and Bing is merely the tip of the iceberg. The developing semantic web of material and data is a big potential to tap into when it comes to intelligent content, semantic search, and smart devices. The Semantic Web will continue to give birth to new careers, businesses, and global innovators.

Publishers can use Semantic Web Technologies to:

i) Create intelligent digital content infrastructures

ii) Connect disparate content silos throughout a large corporation.

iii) Make use of information to create more immersive experiences.

iv) More effectively curate and utilise content

v) Connect content sets from both inside and outside the company.

vi) Invest in real-world augmented and artificial intelligence.

vii) Enhance your authoring experiences and workflows.

To engineer in a way that anticipates shifting content ecosystems, we must first comprehend the importance of semantic data linkages and gradually incorporate semantic information and relationships into every piece of material we create.

Real-world applications of Semantic Web

Here are some applications of semantic web in real world:

i) The oil and gas industry was reported to be using RDF/OWL in 2007 to combine data from various sources and standardise data exchange, sharing, and integration across partners or applications. It was also feasible to handle knowledge collaboratively.

ii) The BBC website employed semantic web technology to dynamically display material during the 2010 FIFA World Cup. We used SPARQL queries and OWL 2 RL reasoning. With the success of this project, in January 2013, BBC committed to the development of Linked Data Platform to enable dynamic semantic publishing. BBC's Music site from 2008 was also an early example of using the semantic web.

iii) Facebook announced Open Network in April 2010, allowing web publishers to incorporate their websites into Facebook's social graph. Facebook may then utilise this information to figure out what a user likes, provide customised suggestions, and connect individuals with similar interests. It was decided to use a reduced version of RDFa.

5.2 NATURAL LANGUAGE PROCESSING

5.2.1 Introduction

Q3. What is Natural Language Processing? State its advantages and disadvantages.

Ans :

Meaning

Natural Language Processing (NLP) is a branch of AI that helps computers to understand, interpret and manipulate human languages like English or Hindi to analyze and derive it's meaning. NLP helps developers to organize and structure knowledge to perform tasks like translation, summarization, named entity recognition, relationship extraction, speech recognition, topic segmentation, etc.

Advantages

1. NLP helps users to ask questions about any subject and get a direct response within seconds.
2. NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.
3. NLP helps computers to communicate with humans in their languages.
4. It is very time efficient.
5. Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages

A list of disadvantages of NLP is given below:

1. NLP may not show context.
2. NLP is unpredictable
3. NLP may require more keystrokes.
4. NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

Q4. Expalin the components and applications of NLP.

Ans :

Components of NLP

There are the following two components of NLP:

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks :

i) It is used to map the given input into useful representation.

ii) It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Applications

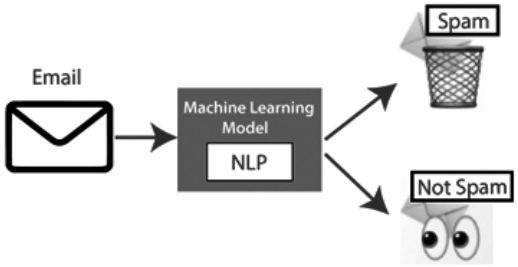
There are the following applications of NLP:

1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

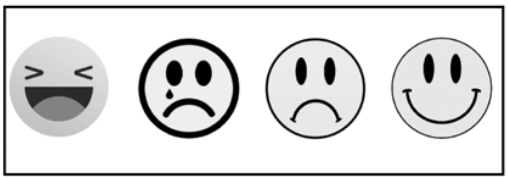
2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox.



3. Sentiment Analysis

Sentiment Analysis is also known as opinion mining. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)



4. Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

Example: Google Translator

5. Spelling correction

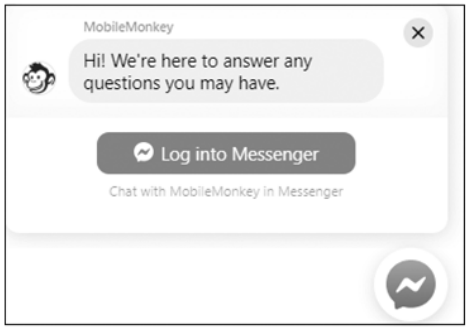
Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.

6. Speech Recognition

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

7. Chatbot

Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.



8. Information extraction

Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

9. Natural Language Understanding (NLU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

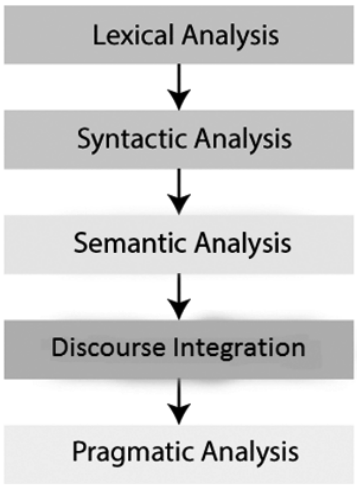
5.2.2 Sentence Analysis Phases

Q5. Explain about the phases of NLP.

Ans :

Phases of NLP

There are the following five phases of NLP:



1. Lexical Analysis and Morphological

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes. It divides the whole text into paragraphs, sentences, and words.

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example: "Open the door" is interpreted as a request instead of an order.

Q6. Explain about the steps to build NLP Pipeline.

Ans :

How to build an NLP pipeline

There are the following steps to build an NLP pipeline:

Step 1: Sentence Segmentation

Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

Example: Consider the following paragraph -

Independence Day is one of the important festivals for every Indian citizen. It is celebrated on the 15th of August each year ever since India got independence from the British rule. The day celebrates independence in the true sense.

Sentence Segment produces the following result:

1. "Independence Day is one of the important festivals for every Indian citizen."
2. "It is celebrated on the 15th of August each year ever since India got independence from the British rule."
3. "This day celebrates independence in the true sense."

Step 2: Word Tokenization

Word Tokenizer is used to break the sentence into separate words or tokens.

Example: JavaTpoint offers Corporate Training, Summer Training, Online Training, and Winter Training.

Word Tokenizer generates the following result:

"JavaTpoint", "offers", "Corporate", "Training", "Summer", "Training", "Online", "Training", "and", "Winter", "Training", "."

Step 3: Stemming

Stemming is used to normalize words into its base form or root form. For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate."

The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

For Example, intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning.

Step 4: Lemmatization

Lemmatization is quite similar to the Stamming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

For example: In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.

Step 5: Identifying Stop Words

In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. **Stop words** might be filtered out before doing any statistical analysis.

Example: He **is a** good boy.

Note: When you are building a rock band search engine, then you do not ignore the word "The."

Step 6: Dependency Parsing

Dependency Parsing is used to find that how all the words in the sentence are related to each other.

Step 7: POS tags

POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates that how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.

Example: "Google" something on the Internet.

In the above example, Google is used as a verb, although it is a proper noun.

Step 8: Named Entity Recognition (NER)

Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

Example: Steve Jobs introduced iPhone at the Macworld Conference in San Francisco, California.

Step 9: Chunking

Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

5.2.3 Grammars And Parsers

Q7. What is Grammar in AI? Explain.

Ans :

A grammar is a declarative representation that defines the syntactic facts of a language. The most common way to represent grammars is as a set of production rules, and the simplest structure for them to build is a parse tree which records the rules and how they are matched.

A grammar G can be formally written as a 4-tuple (N, T, S, P) where,

i) N or V_N is a set of variables or non-terminal symbols.

ii) T or Σ is a set of Terminal symbols.

iii) S is a special variable called the Start symbol, S $\in$ N

iv) P is Production rules for Terminals and Non-terminals. A production rule has the form $\alpha \rightarrow \beta$, where á and â are strings on $V_N \cup \Sigma$ and least one symbol of á belongs to V_N.

Example

Grammar G1

$(\{S, A, B\}, \{a, b\}, S, \{S \rightarrow AB, A \rightarrow a, B \rightarrow b\})$

Here,

i) S, A, and B are Non-terminal symbols;

ii) a and b are Terminal symbols

iii) S is the Start symbol, S " N

iv) Productions, $P : S \rightarrow AB, A \rightarrow a, B \rightarrow b$

Example

Grammar G2

$(\{S, A\}, \{a, b\}, S, \{S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \varepsilon\})$

Here,

i) S and A are Non-terminal symbols.

ii) a and b are Terminal symbols.

iii) ε is an empty string.

iv) S is the Start symbol, $S \in N$

v) Production $P : S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \varepsilon$

Derivations from a Grammar

Strings may be derived from other strings using the productions in a grammar. If a grammar G has a production $\alpha \rightarrow \beta$, we can say that $x\alpha y$ derives $x\beta y$ in G. This derivation is written as,

$$x\alpha yK \Rightarrow Gx\beta y$$

Example

Let us consider the grammar:

$G2 = (\{S, A\}, \{a, b\}, S, \{S \rightarrow aAb, aA \rightarrow aaAb, A \rightarrow \varepsilon\})$

Some of the strings that can be derived are,

$S \Rightarrow$ aAb using production $S \rightarrow aAb$

$\Rightarrow$ aaAbb using production $aA \rightarrow aAb$

$\Rightarrow$ aaaAbbb using production $aA \rightarrow aaAb$

$\Rightarrow$ aaabbb using production $A \rightarrow \varepsilon$

Q8. Explain the types of grammars in AI

OR

Explain the Chomsky classification of grammar.

Ans :

Types of Grammars

By introducing more or less restrictive criteria on the form of the grammar rules, we obtain hierarchical grammar classes (types of grammars), ordered by inclusion. The classification of grammars, defined in 1957 by Noam Chomsky, distinguishes four classes.

Chomsky Classification

Type 0: no restriction on the rules.

Type 1: context sensitive or contextual grammars. The rules of R are of the form:

i) $uAv \rightarrow uwv$ with $A \in N$, $u, v \in (N \cup T)^*$ and $w \in (N \cup T)^+$

ii) In other words, the non-terminal symbol A is replaced by w if we have the contexts u on the left and v on the right.

Type 2: context-free grammars. The rules of R are of the form:

i) $A \rightarrow w$ with $A \in N$ and $w \in (N \cup T)^*$

ii) In other words, the left side of each rule consists of a single non-terminal symbol.

Type 3: regular grammars

i) to the right. The rules of R are of the form

$A \rightarrow aB$ or $A \rightarrow a$ with $A, B \in N$ and $a \in T$

ii) to the left. The rules of R are of the form

$A \rightarrow Ba$ or $A \rightarrow a$ with $A, B \in N$ and $a \in T$

iii) In other words, the left side of each rule consists of a single non-terminal symbol, and the right side consists of a terminal symbol and possibly a non-terminal symbol. For regular grammars on the right, the non-terminal symbol must always be to the right of the terminal symbol while for regular grammars on the left it must be on the left.

Example 1

We consider the language L of words on {0, 1} which represent even unsigned base 2 integers (the words of this language all end with 0 and do not start with 0, except for the zero integer). Formally define L and construct a regular grammar describing L.

The language is of the following form: $L = \{0, 1u0 / u \in \{0, 1\}^*\}$

We define the regular right-hand grammar G = (T, N, S, R) where

$T = \{0, 1\}$

$N = \{S, U\}$

$R = \{S \rightarrow 0 \mid 1U$

$U \rightarrow 1U \mid 0U \mid 0\}$

as well as the regular left grammar G = (T, N, S, R) where

$T = \{0, 1\}$

$N = \{S, U\}$

$R = \{S \rightarrow 0 \mid U0$

$U \rightarrow U1 \mid U0 \mid 1\}$

Preferably, we always take the regular right-hand grammar for the study of future states in automata.

Example 2

Here are the grammar productions: $P \rightarrow aP$, $P \rightarrow aQ$, $Q \rightarrow bP$, $Q \rightarrow R$, $R \rightarrow bR$, $R \rightarrow cQ$, $R \rightarrow bP$, $R \rightarrow \varepsilon$. The non-terminals of the grammar are {P, Q, R}, the initial symbol is P.

By denoting with X_p, X_q, X_r the languages accepted from the states P, Q and R respectively, the system of equations for these languages is:

$$\begin{cases} X_p = aX_p = aX_q \\ X_q = bX_p + X_r \\ X_r = bX_r = cX_q + bX_{p+e} \end{cases}$$

$$\begin{cases} X_p = aX_p = abX_p + aX_r \\ X_r = bX_r + cbX_p + cX_r + bX_p + c \end{cases}$$

On factorise X_r dans la deuxieme equation

$$X_r = (b + c)^* (cbX_p + bX_p + c)$$

On templace Xr dans 1a premiere equation

$$X_p = (a + ab)X_p + a(b+c)^*(cb + b)X_p + a(b+ c)^*$$

On factorise Xp dans la premiree equation

$$X_p = (a + ab + a(b + c)^* (cb + b))^* a(b + c)^*$$

5.2.4 Types of Parsers

Q9. Explain briefly about Parsing.

Ans :

The NLP process starts by first converting our input text into a series of tokens (called the Doc object) and then performing several operations of the Doc object. A typical NLP processing process consists of stages like tokenizer, tagger, lemmatizer, parser, and entity recognizer. In every stage, the input is the Doc object and the output is the processed Doc, so, every stage makes some kind of respective change to the Doc object and feeds it to the subsequent stage of the process. The processing of our text (which is termed as Doc) is done in various incremental stages. So, whenever we provide the input text to the NLP, the processing starts by first tokenizing it (converting the text into tokens), and then it is converted into an object namely- Doc.

Now, in all of the above components, the parser plays a very vital role as it reports any kind of syntax error in the text. The parser not only tells us about the syntax errors but also helps us to recover from the most commonly occurring errors so that the processing of the rest of the program does not get halted.

Note: The parser is very important in the world of TOC as well, as it helps to generate the parse tree for the compiler.

The word parsing comes from the Latin word pars which means part. The parser helps us to get the meaning of the provided text (like the dictionary meaning of the provided text). As we discussed that the parser helps us to analyze the syntax error in the text; so, the parsing process is also known as the syntax analysis or the Syntactic analysis.

The syntax analysis uses the formal grammar rules to check the text and if the text follows the rules of grammar then it accepts the text else it rejects the text. For example, if the input text is: Give me hot ice cream.. Then the parser or the syntactic analyzer will reject the text as it does not follow the rules of formal grammar.

The formal definition of a parser in NLP can be a component that process and analyzes the input string of symbols and checks or confirms whether the string is following the rules of the formal grammar or not.

The main functions of a parser are as follows:

i) The parser reports to us any syntax error in the syntax of the text.

ii) The parser helps us to recover from the most commonly occurring errors so that the processing of the rest of the program does not get halted.

iii) Parser helps in generating the parse tree.

iv) Parser also helps in creating the symbol table that is used by various stages of the NLP process and thus it is quite important.

v) Parses also help in the production of IR or Intermediate Representations.

Let us now look at some of the areas where Parsing plays a vital role in the NLP process.

1. Sentiment Analysis

2. Relation Extraction

3. Question Answering

4. Speech Recognition

5. Machine Translation

6. Grammar Checking

We mainly use two types of parsing namely- deep parsing and, shallow parsing. In shallow parsing, we only pass a limited portion of the entire syntactic information so, it is usually used for the less important or less complex NLP applications. Some of the use cases of shallow parsing are- information extraction, and text mining. On the other hand, in deep parsing, we follow a search strategy that results in the complete syntactic structure of a sentence so, it can be used for complex NLP applications. Some of the use cases of deep parsing are- dialogue systems, and summarization.

The parsing is done in several ways hence we can say that there are several parsers present in the NLP world. Let us look at some of the most commonly used parsers and the subsequent parsing techniques in the coming sections.

Q10. Explain various types of parsing in AI.

Ans :

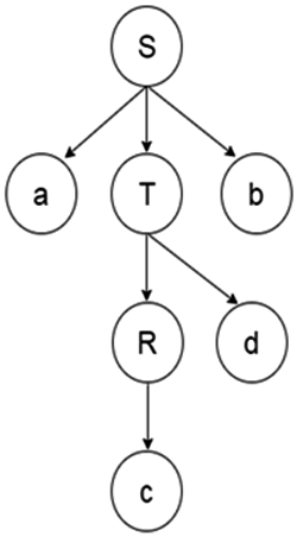
Top down paring

i) The top down parsing is known as recursive parsing or predictive parsing.

ii) Bottom up parsing is used to construct a parse tree for an input string.

iii) In the top down parsing, the parsing starts from the start symbol and transform it into the input symbol.

Parse Tree representation of input string "acdb" is as follows:



Bottom up parsing

i) Bottom up parsing is also known as shift-reduce parsing.

ii) Bottom up parsing is used to construct a parse tree for an input string.

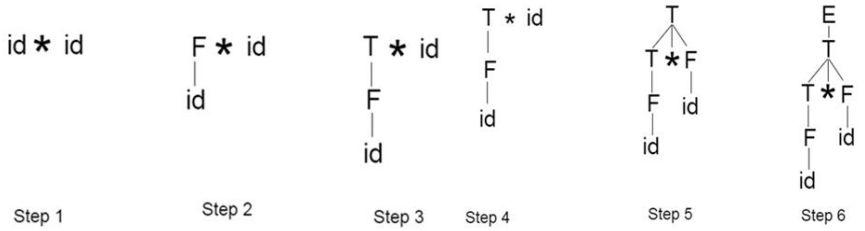
iii) In the bottom up parsing, the parsing starts with the input symbol and construct the parse tree up to the start symbol by tracing out the rightmost derivations of string in reverse.

Example

Production

1. $E \rightarrow T$
2. $T \rightarrow T * F$
3. $T \rightarrow id$
4. $F \rightarrow T$
5. $F \rightarrow id$

Parse Tree representation of input string "id * id" is as follows:



Bottom up parsing is classified in to various parsing. These are as follows:

1. Shift-Reduce Parsing
2. Operator Precedence Parsing
3. Table Driven LR Parsing
   - **a.** LR(1)
   - **b.** SLR(1)
   - **c.** CLR (1)
   - **d.** LALR(1)

5.2.5 Semantic Analysis

Q11. Define semantic analysis. Explain the elements of semantic analysis.

Ans :

Meaning

Semantic Analysis is a subfield of Natural Language Processing (NLP) that attempts to understand the meaning of Natural Language. Understanding Natural Language might seem a straightforward process to us as humans. However, due to the vast complexity and subjectivity involved in human language, interpreting it is quite a complicated task for machines. Semantic Analysis of Natural Language captures the meaning of the given text while taking into account context, logical structuring of sentences and grammar roles.

Parts of Semantic Analysis

Semantic Analysis of Natural Language can be classified into two broad parts:

1. Lexical Semantic Analysis

Lexical Semantic Analysis involves understanding the meaning of each word of the text individually. It basically refers to fetching the dictionary meaning that a word in the text is deputed to carry.

2. Compositional Semantics Analysis

Although knowing the meaning of each word of the text is essential, it is not sufficient to completely understand the meaning of the text.

For example, consider the following two sentences:

i) Sentence 1: Students love GeeksforGeeks.

ii) Sentence 2: GeeksforGeeks loves Students.

Although both these sentences 1 and 2 use the same set of root words {student, love, geeksforgeeks}, they convey entirely different meanings. Hence, under Compositional Semantics Analysis, we try to understand how combinations of individual words form the meaning of the text.

Tasks involved in Semantic Analysis

In order to understand the meaning of a sentence, the following are the major processes involved in Semantic Analysis:

1. Word Sense Disambiguation
2. Relationship Extraction

Word Sense Disambiguation

In Natural Language, the meaning of a word may vary as per its usage in sentences and the context of the text. Word Sense Disambiguation involves interpreting the meaning of a word based upon the context of its occurrence in a text.

For example, the word 'Bark' may mean 'the sound made by a dog' or 'the outermost layer of a tree.'

Likewise, the word 'rock' may mean '*a stone*' or '*a genre of music*' – hence, the accurate meaning of the word is highly dependent upon its context and usage in the text. Thus, the ability of a machine to overcome the ambiguity involved in identifying the meaning of a word based on its usage and context is called Word Sense Disambiguation.

Relationship Extraction

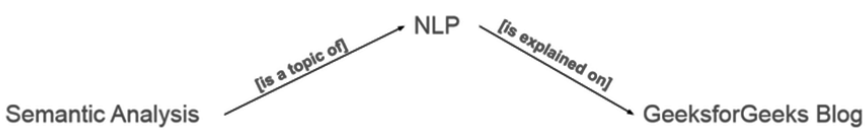
Another important task involved in Semantic Analysis is Relationship Extracting. It involves firstly identifying various entities present in the sentence and then extracting the relationships between those entities.

For example, consider the following sentence:

Semantic Analysis is a topic of NLP which is explained on the GeeksforGeeks Blog

Semantic Analysis [Entity] NLP [Entity] GeeksforGeeks Blog [Entity]

Entities



Relationships

Elements of Semantic Analysis

Some of the critical elements of Semantic Analysis that must be scrutinized and taken into account while processing Natural Language are:

1. Hyponymy

Hyponymys refers to a term that is an instance of a generic term. They can be understood by taking class-object as an analogy. For example: 'Color' is a hypernymy while 'grey', 'blue', 'red', etc, are its hyponyms.

2. Homonymy

Homonymy refers to two or more lexical terms with the same spellings but completely distinct in meaning. For example: 'Rose' might mean 'the past form of rise' or 'a flower', – same spelling but different meanings; hence, 'rose' is a homonymy.

3. Synonymy

When two or more lexical terms that might be spelt distinctly have the same or similar meaning, they are called Synonymy. For example: (Job, Occupation), (Large, Big), (Stop, Halt).

4. Antonymy

Antonymy refers to a pair of lexical terms that have contrasting meanings – they are symmetric to a semantic axis. For example: (Day, Night), (Hot, Cold), (Large, Small).

5. Polysemy

Polysemy refers to lexical terms that have the same spelling but multiple closely related meanings. It differs from homonymy because the meanings of the terms need not be closely related in the case of homonymy. For example: 'man' may mean 'the human species' or 'a male human' or 'an adult male human' – since all these different meanings bear a close association, the lexical term 'man' is a polysemy.

6. Meronomy

Meronomy refers to a relationship wherein one lexical term is a constituent of some larger entity. For example: 'Wheel' is a meronym of 'Automobile'

Q12. Define meaning representation in AI. Explain the approaches in meaning representation.

Ans :

Meaning Representation

While, as humans, it is pretty simple for us to understand the meaning of textual information, it is not so in the case of machines. Thus, machines tend to represent the text in specific formats in order to interpret its meaning. This formal structure that is used to understand the meaning of a text is called meaning representation.

Basic Units of Semantic System:

In order to accomplish Meaning Representation in Semantic Analysis, it is vital to understand the building units of such representations. The basic units of semantic systems are explained below:

1. Entity

An entity refers to a particular unit or individual in specific such as a person or a location. For example GeeksforGeeks, Delhi, etc.

2. Concept

A Concept may be understood as a generalization of entities. It refers to a broad class of individual units. For example Learning Portals, City, Students.

3. Relations

Relations help establish relationships between various entities and concepts. For example: 'GeeksforGeeks is a Learning Portal', 'Delhi is a City.', etc.

4. Predicate

Predicates represent the verb structures of the sentences.

In Meaning Representation, we employ these basic units to represent textual information.

Approaches to Meaning Representations

Now that we are familiar with the basic understanding of Meaning Representations, here are some

of the most popular approaches to meaning representation:

1. First-order predicate logic (FOPL)
2. Semantic Nets
3. Frames
4. Conceptual dependency (CD)
5. Rule-based architecture
6. Case Grammar
7. Conceptual Graphs

Q13. What are various techniques in Semantic Analysis.

Ans :

Semantic Analysis Techniques

Based upon the end goal one is trying to accomplish, Semantic Analysis can be used in various ways. Two of the most common Semantic Analysis techniques are:

Text Classification

In-Text Classification, our aim is to label the text according to the insights we intend to gain from the textual data.

For example

1. In Sentiment Analysis, we try to label the text with the prominent emotion they convey. It is highly beneficial when analyzing customer reviews for improvement.
2. In Topic Classification, we try to categories our text into some predefined categories. For example: Identifying whether a research paper is of Physics, Chemistry or Maths
3. In Intent Classification, we try to determine the intent behind a text message. For example: Identifying whether an e-mail received at customer care service is a query, complaint or request.

Text Extraction

In-Text Extraction, we aim at obtaining specific information from our text.

For Example,

1. In Keyword Extraction, we try to obtain the essential words that define the entire document.
2. In Entity Extraction, we try to obtain all the entities involved in a document.

Significance of Semantics Analysis

Semantics Analysis is a crucial part of Natural Language Processing (NLP). In the ever-expanding era of textual information, it is important for organizations to draw insights from such data to fuel businesses. Semantic Analysis helps machines interpret the meaning of texts and extract useful information, thus providing invaluable data while reducing manual efforts.

Besides, Semantics Analysis is also widely employed to facilitate the processes of automated answering systems such as chatbots – that answer user queries without any human interventions.

5.2.6 Universal Networking Knowledge

Q14. What is Universal Networking Language? Explain.

Ans :

Universal Networking Language (UNL) is a declarative formal language specifically designed to represent semantic data extracted from natural language texts. It can be used as a pivot language in interlingual machine translation systems or as a knowledge representation language in information retrieval applications.

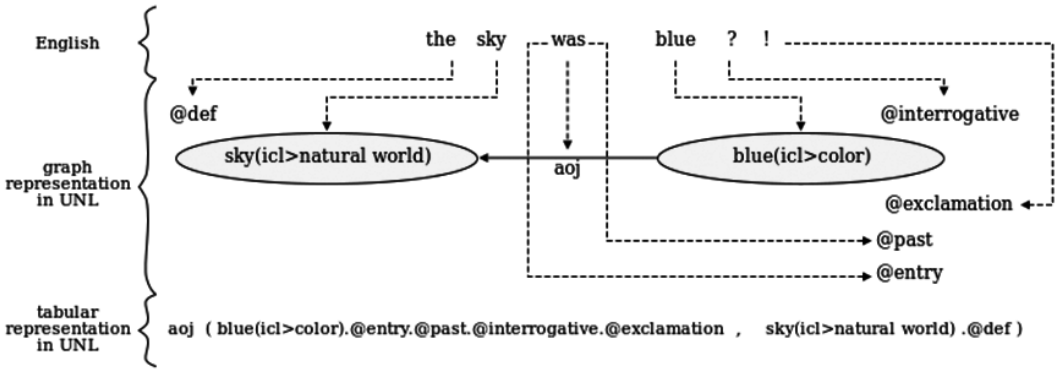
UNL is designed to establish a simple foundation for representing the most central aspects of information and meaning in a machine- and human-language-independent form. As a language-independent formalism, UNL aims to code, store, disseminate and retrieve information independently of the original language in which it was expressed. In this sense, UNL seeks to provide tools for overcoming the language barrier in a systematic way.

Structure

In the UNL approach, information conveyed by natural language is represented sentence by sentence as a hypergraph composed of a set of directed binary labeled links (referred to as relations) between nodes or

hypernodes (the Universal Words, or simply UWs), which stand for concepts. UWs can also be annotated with attributes representing context information.

As an example, the English sentence 'The sky was blue?!' can be represented in UNL as follows:



In the example above, sky(icl>natural world) and blue(icl>color), which represent individual concepts, are UWs; "aoj" (= attribute of an object) is a directed binary semantic relation linking the two UWs; and "@def", "@interrogative", "@past", "@exclamation" and "@entry" are attributes modifying UWs.

UWs are intended to represent universal concepts, but are expressed in English words or in any other natural language in order to be humanly readable. They consist of a "headword" (the UW root) and a "constraint list" (the UW suffix between parentheses), where the constraints are used to disambiguate the general concept conveyed by the headword. The set of UWs is organized in the UNL Ontology, in which high-level concepts are related to lower-level ones through the relations "icl" (= is a kind of), "iof" (= is an instance of) and "equ" (= is equal to).

Relations are intended to represent semantic links between words in every existing language. They can be ontological (such as "icl" and "iof," referred to above), logical (such as "and" and "or"), and thematic (such as "agt" = agent, "ins" = instrument, "tim" = time, "plc" = place, etc.). There are currently 46 relations in the UNL Specs. They jointly define the UNL syntax.

Attributes represent information that cannot be conveyed by UWs and relations. Normally, they represent information concerning time ("@past", "@future", etc.), reference ("@def", "@indef", etc.), modality ("@can", "@must", etc.), focus ("@topic", "@focus", etc.), and so on.

Within the UNL Program, the process of representing natural language sentences in UNL graphs is called UNLization, and the process of generating natural language sentences out of UNL graphs is called NLization. UNLization, which involves natural language analysis and understanding, is intended to be carried out semi-automatically (i.e., by humans with computer aids); and NLization is intended to be carried out fully automatically.

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - I

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. (a) What is Artificial Intelligence? Write about various approaches used to define AI. **(Unit-I, Q.No. 1)**

   (b) What is problem in AI? Write the steps for problem solving in AI. **(Unit-I, Q.No. 7)**

OR

2. (a) Write about problem characteristics. **(Unit-I, Q.No. 12)**

   (b) Explain Iterative Deepening Depth-First Search algorithm with an example **(Unit-I, Q.No. 20)**

3. (a) Explain about propositional logical connectives and logical equivalances. **(Unit-II, Q.No. 4)**

   (b) Explain about resolution refutation in propositional logic. **(Unit-II, Q.No. 9)**

OR

4. (a) Define inference rules? Explain the types of inference rules. **(Unit-II, Q.No. 5)**

   (b) What is the Relation between Knowledge & Intelligence? **(Unit-II, Q.No. 13)**

5. (a) Explain various steps to develop an Expert system. **(Unit-III, Q.No. 2)**

   (b) Explain about expert system shells? **(Unit-III, Q.No. 8)**

OR

6. (a) What is uncertainity? State its causes. **(Unit-III, Q.No. 10)**

   (b) Explain briefly about Expert System? **(Unit-III, Q.No. 1)**

7. (a) Explain the main key components of Machine Learning Systems? **(Unit-IV, Q.No. 2)**

   (b) Explain briefly about learning decision trees in ML. **(Unit-IV, Q.No. 8)**

OR

8. (a) What are SVMs? Explain, the working mechanism of SVM. **(Unit-IV, Q.No. 13)**

   (b) Explain the advantages and disadvantags of Artificial Neural Network. **(Unit-IV, Q.No. 16)**

9. (a) Explain various cases in case grammars. **(Unit-V, Q.No. 1)**

(b) Explain about the phases of NLP. **(Unit-V, Q.No. 5)**

OR

10. (a) Expalin the components and applications of NLP. **(Unit-V, Q.No. 4)**

(b) What are various techniques in Semantic Analysis. **(Unit-V, Q.No. 13)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - II

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. (a) What are intelligent systems? Write about its types. **(Unit-I, Q.No. 3)**

   (b) Explain about problem formulation. **(Unit-I, Q.No. 10)**

OR

2. (a) Explain how to analyse and represent the problem. **(Unit-I, Q.No. 13)**

   (b) Explain Best- Frist Search algorithm. **(Unit-I, Q.No. 19)**

3. (a) What is propositional logic? Explain the basic facts about propositional logic. **(Unit-II, Q.No. 3)**

   (b) Define predicate logic. Explain components and characteristics of predicate logic. **(Unit-II, Q.No. 10)**

OR

4. (a) Write about natural deduction system. **(Unit-II, Q.No. 6)**

   (b) What is the Relation between Knowledge & Intelligence? **(Unit-II, Q.No. 15)**

5. (a) Differentiate between conventional systems and expert systems. **(Unit-III, Q.No. 4)**

   (b) Explain Bayes' Theorem. **(Unit-III, Q.No. 12)**

OR

6. (a) Solve the following examples using Bayes' theorem. **(Unit-III, Q.No. 13)**

   (b) Explain the expert of architecture system. **(Unit-III, Q.No. 3)**

7. (a) Explain the categories of supervised machine learning. **(Unit-IV, Q.No. 4)**

   (b) How to build the decision tree by using Information Gain. **(Unit-IV, Q.No. 9)**

OR

8. (a) Explain briefly about Artificial Neural Network? **(Unit-IV, Q.No. 14)**

   (b) Write the architecture and applications of feed forward network. **(Unit-IV, Q.No. 20)**

9. (a) Explain importance of Semantic web in AI? **(Unit-V, Q.No. 2)**

(b) Explain about the steps to build NLP Pipeline. **(Unit-V, Q.No. 6)**

OR

10. (a) Explain briefly about Parsing. **(Unit-V, Q.No. 9)**

(b) What is Universal Networking Language? Explain. **(Unit-V, Q.No. 14)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - III

ARTIFICIAL INTELLIGENCE

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. (a) Explain about the sub areas of AI. **(Unit-I, Q.No. 5)**

   (b) Explain the steps needed to build a system to solve a particular problem. **(Unit-I, Q.No. 9)**

OR

2. (a) Solve Kanpsack problem using Exhaustic search method. **(Unit-I, Q.No. 15)**

   (b) Explain how to solve constraint satisfaction problems using Search. **(Unit-I, Q.No. 24)**

3. (a) Explain briefly about logic programming? **(Unit-II, Q.No. 1)**

   (b) Write about logical connectives and Quantifiers in predicate logic. **(Unit-II, Q.No. 11)**

OR

4. (a) Explain about semantic tableau system in propositional logic. **(Unit-II, Q.No. 8)**

   (b) Write about various approaches to knowledge representation in AI. **(Unit-II, Q.No. 17)**

5. (a) What are the applications of Expert systems? Write. **(Unit-III, Q.No. 7)**

   (b) Expalin briefly about probability theory. **(Unit-III, Q.No. 11)**

OR

6. (a) Solve the following problem by using Bayesian network. **(Unit-III, Q.No. 15)**

   (b) What is a Truth Maintenance System (TMS)? Expalin. **(Unit-III, Q.No. 5)**

7. (a) Explain the Advantages and Disadvantages and applications of Unsupervised Learning. **(Unit-IV, Q.No. 6)**

   (b) Expalin briefly about Clustering Algorithms. **(Unit-IV, Q.No. 12)**

OR

8. (a) Explain the working mechanism of ANN. How do artificial neural networks work? **(Unit-IV, Q.No. 17)**

   (b) Explain about Multi Layer Feed Forward Networks. **(Unit-IV, Q.No. 22)**

9. (a) What is Natural Language Processing? State its advantages and disadvantages. **(Unit-V, Q.No. 3)**

(b) What is Grammar in AI? Explain. **(Unit-V, Q.No. 7)**

OR

10. (a) Explain various types of parsing in AI. **(Unit-V, Q.No. 10)**

(b) Define semantic analysis. Explain the elements of semantic analysis. **(Unit-V, Q.No. 11)**