ANNAMACHARYA
INSTITUTE OF TECHNOLOGY AND SCIENCES
(AUTONOMOUS)

Approved by AICTE, New Delhi & Permanent Affiliation to JNTUA, Anantapur.

Three B. Tech Programmes (CSE , ECE & CE) are accredited by NBA, New Delhi,Accredited by NAAC with 'A' Grade , Bangalore.

A-grade awarded by AP Knowledge Mission. Recognized under sections 2(f) & 12(B) of UGC Act 1956.

Venkatapuram Village, Renigunta Mandal, Tirupati, Andhra Pradesh-517520.

Department of Artificial Intelligence

Academic Year 2023-24

III. B.Tech I Semester

ARTIFICIAL INTELLIGENCE

(20APC3017)

Prepared By

Mrs.H.Teja M.Tech,(Ph.D)
Assistant Professor, AITS

ARTIFICIAL INTELLIGENCE

(19A05502T)

Syllabus: UNIT – I

Introduction: What is AI, Foundations of AI, History of AI, The State of Art. Intelligent Agents: Agents and Environments, Good Behavior: The Concept of Rationality, The Nature of Environments And The Structure of Agents.

*****

INTRODUCTION:

- In today's world, technology is growing very fast, and we are getting in touch with different new technologies day by day.
- Here, one of the booming technologies of computer science is Artificial Intelligence which is ready to create a new revolution in the world by making intelligent machines.
- The Artificial Intelligence is now all around us. It is currently working with a variety of subfields, ranging from general to specific, such as self-driving cars, playing chess, proving theorems, playing music, Painting, etc.
- AI is one of the fascinating and universal fields of Computer science which has a great scope in future. AI holds a tendency to cause a machine to work as a human.

WHAT IS AI?

Artificial Intelligence is composed of two words **Artificial** and **Intelligence**, where Artificial defines ***"man-made,"*** and intelligence defines ***"thinking power",*** hence AI means *"a man-made thinking power".*

So, we can define AI as:

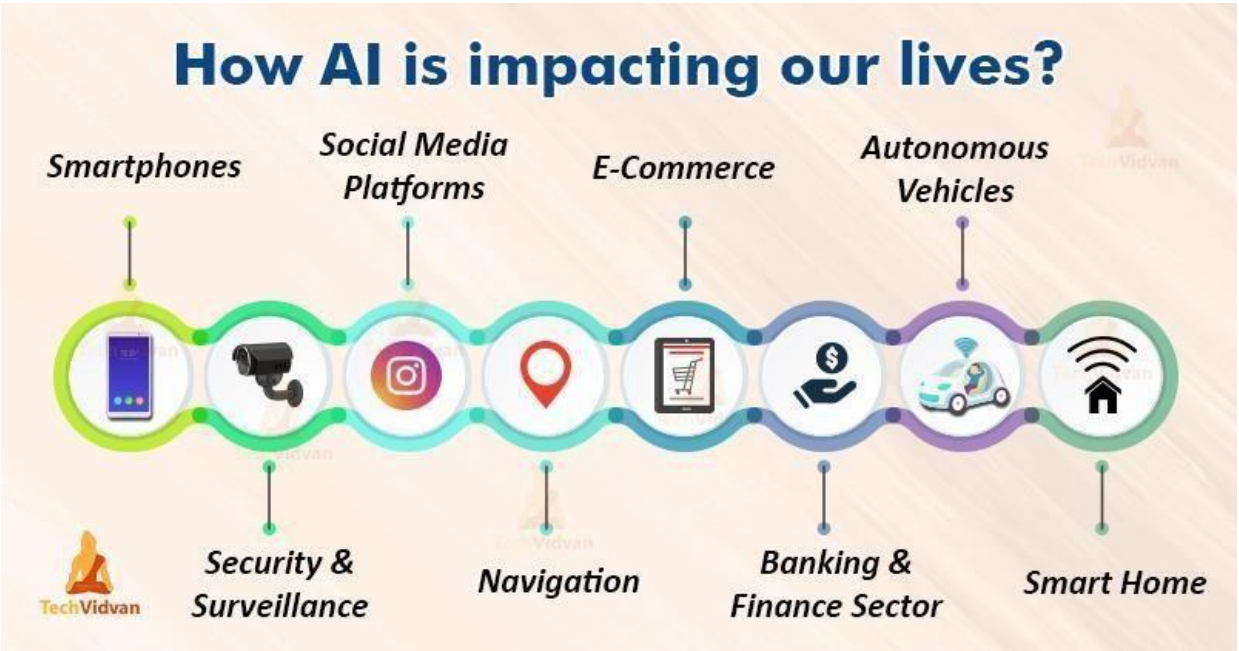
"It is a branch of computer science by which we can create intelligent machines which can behave like a human, think like humans and able to make decisions."

(OR)

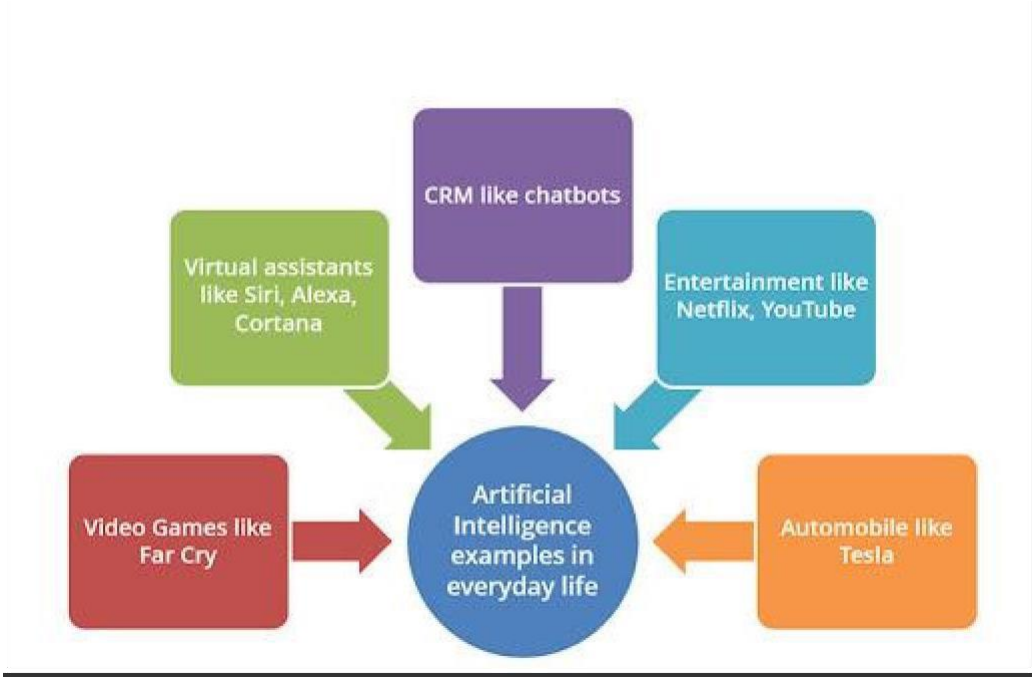
"Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems".

- Artificial Intelligence exists when a machine can have human based skills such as learning, reasoning, and solving problems
- With Artificial Intelligence you do not need to preprogram a machine to do some work, despite that you can create a machine with programmed algorithms which can work with own intelligence, and that is the awesomeness of AI.

IMPACT OF AI IN OUR LIVES:



EXAMPLES OF AI:



WHY ARTIFICIAL INTELLIGENCE?

Before Learning about Artificial Intelligence, we should know that what is the importance of AI and why should we learn it. Following are some main reasons to learn about AI:

- With the help of AI, you can create such software or devices which can solve real-world problems very easily and with accuracy such as health issues, marketing, traffic issues, etc.
- With the help of AI, you can create your personal virtual Assistant, such as Cortana, Google Assistant, Siri, etc.
- With the help of AI, you can build such Robots which can work in an environment where survival of humans can be at risk.
- AI opens a path for other new technologies, new devices, and new Opportunities.

GOALS OF ARTIFICIAL INTELLIGENCE:

Following are the main goals of Artificial Intelligence:

1. Replicate human intelligence
2. Solve Knowledge-intensive tasks
3. An intelligent connection of perception and action
4. Building a machine which can perform tasks that requires human intelligence such as:
    - Proving a theorem
    - Playing chess
    - Plan some surgical operation
    - Driving a car in traffic
5. Creating some system which can exhibit intelligent behavior, learn new things by itself, demonstrate, explain, and can advise to its user.

ADVANTAGES OF ARTIFICIAL INTELLIGENCE:

Following are some main advantages of Artificial Intelligence:

- **High Accuracy with fewer errors:** AI machines or systems are prone to less errors and high accuracy as it takes decisions as per pre-experience or information.
- **High-Speed:** AI systems can be of very high-speed and fast-decision making, because of that AI systems can beat a chess champion in the Chess game.
- **High reliability:** AI machines are highly reliable and can perform the same action multiple times with high accuracy.
- **Useful for risky areas:** AI machines can be helpful in situations such as defusing a bomb, exploring the ocean floor, where to employ a human can be risky.
- **Digital Assistant:** AI can be very useful to provide digital assistant to the users such as AI technology is currently used by various E-commerce websites to show the products as per customer requirement.
- **Useful as a public utility:** AI can be very useful for public utilities such as a self-driving car which can make our journey safer and hassle-free, facial recognition for security

purpose, Natural language processing to communicate with the human in human-language, etc.

DISADVANTAGES OF ARTIFICIAL INTELLIGENCE

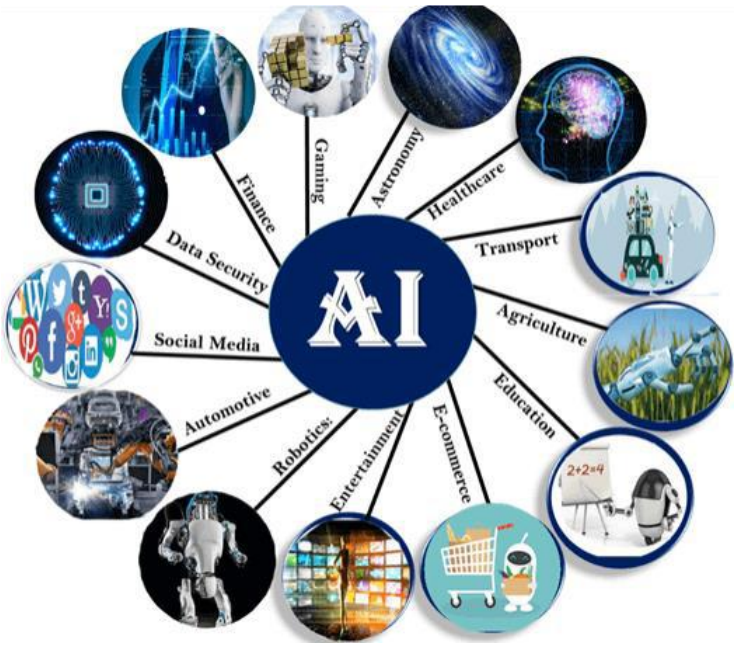
Every technology has some disadvantages, and the same goes for Artificial intelligence. Being so advantageous technology still, it has some disadvantages which we need to keep in our mind while creating an AI system. Following are the disadvantages of AI:

- **High Cost:** The hardware and software requirement of AI is very costly as it requires lots of maintenance to meet current world requirements.
- **Can't think out of the box:** Even we are making smarter machines with AI, but still they cannot work out of the box, as the robot will only do that work for which they are trained, or programmed.
- **No feelings and emotions:** AI machines can be an outstanding performer, but still it does not have the feeling so it cannot make any kind of emotional attachment with human, and may sometime be harmful for users if the proper care is not taken.
- **Increase dependency on machines:** With the increment of technology, people are getting more dependent on devices and hence they are losing their mental capabilities.
- **No Original Creativity:** As humans are so creative and can imagine some new ideas but still AI machines cannot beat this power of human intelligence and cannot be creative and imaginative.

APPLICATIONS OF AI:

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:



1. AI in Astronomy

- Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

2. AI in Healthcare

- In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
- Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

3. AI in Gaming

- AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

4. AI in Finance

- AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

5. AI in Data Security

- The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform, are used to determine software bug and cyber-attacks in a better way.

6. AI in Social Media

- Social Media sites such as Face book, Twitter, and Snap chat contain billions of user profiles, which need to be stored and managed in a very efficient way. AI can organize and manage massive amounts of data. AI can analyze lots of data to identify the latest trends, hash tag, and requirement of different users.

7. AI in Travel & Transport

- AI is becoming highly demanding for travel industries. AI is capable of doing various travel related works such as from making travel arrangement to suggesting the hotels, flights, and best routes to the customers. Travel industries are using AI-powered chatbots which can make human-like interaction with customers for better and fast response.

8. AI in Automotive Industry

- Some Automotive industries are using AI to provide virtual assistant to their user for better performance. Such as Tesla has introduced TeslaBot, an intelligent virtual assistant.
- Various Industries are currently working for developing self-driven cars which can make your journey more safe and secure.

9. AI in Robotics:

- Artificial Intelligence has a remarkable role in Robotics. Usually, general robots are programmed such that they can perform some repetitive task, but with the help of AI, we can create intelligent robots which can perform tasks with their own experiences without pre-programmed.
- Humanoid Robots are best examples for AI in robotics, recently the intelligent Humanoid robot named as Erica and Sophia has been developed which can talk and behave like humans.

10. AI in Entertainment

- We are currently using some AI based applications in our daily life with some entertainment services such as Netflix or Amazon. With the help of ML/AI algorithms, these services show the recommendations for programs or shows.

11. AI in Agriculture

- Agriculture is an area which requires various resources, labor, money, and time for best result. Now a day's agriculture is becoming digital, and AI is emerging in this field. Agriculture is applying AI as agriculture robotics, solid and crop monitoring, predictive analysis. AI in agriculture can be very helpful for farmers.

12. AI in E-commerce

- AI is providing a competitive edge to the e-commerce industry, and it is becoming more demanding in the e-commerce business. AI is helping shoppers to discover associated products with recommended size, color, or even brand.

13. AI in education:

- AI can automate grading so that the tutor can have more time to teach. AI chatbot can communicate with students as a teaching assistant.
- AI in the future can be work as a personal virtual tutor for students, which will be accessible easily at any time and any place.

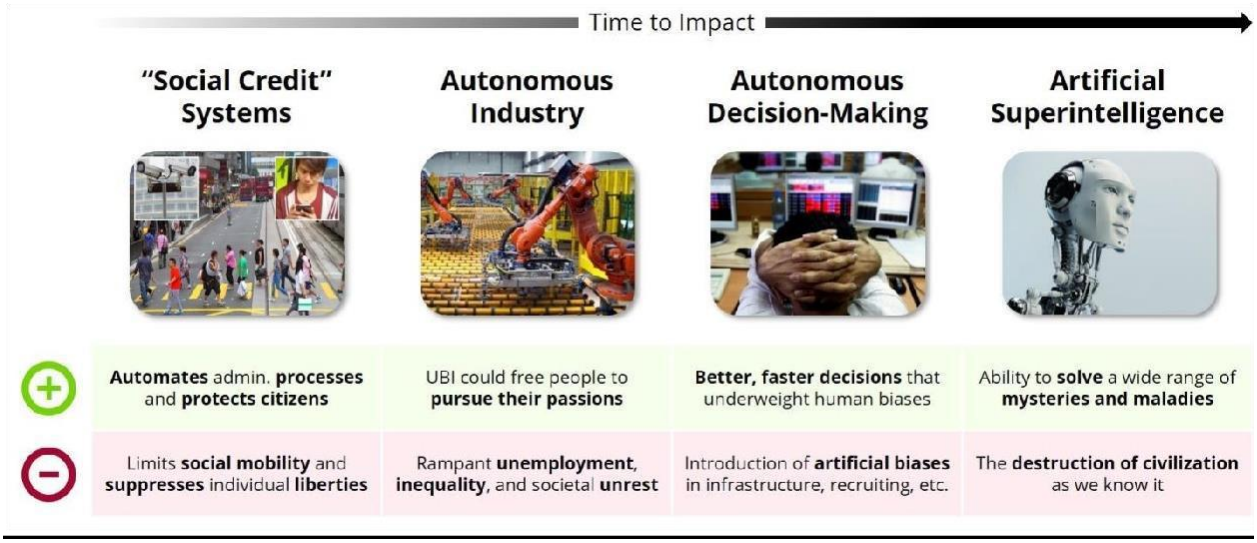
FOUNDATIONS OF AI:

Few technologies have captured the human imagination like artificial intelligence. Stories of intelligent robots and artificial beings have pervaded our myths and legends since antiquity, sparking both excitement and fear as our ancestors considered the existence of autonomous machines. These myths continue to influence the narrative surrounding AI and, though we're closer to developing artificial general intelligence than ever before, it can be difficult to separate fact from fiction given today's buzzword-laden media.

Though certain use cases have undoubtedly been overhyped, AI has the potential to transform every aspect of how we work, play, and live. In this post, we'll lay the foundation for understanding this technology by:

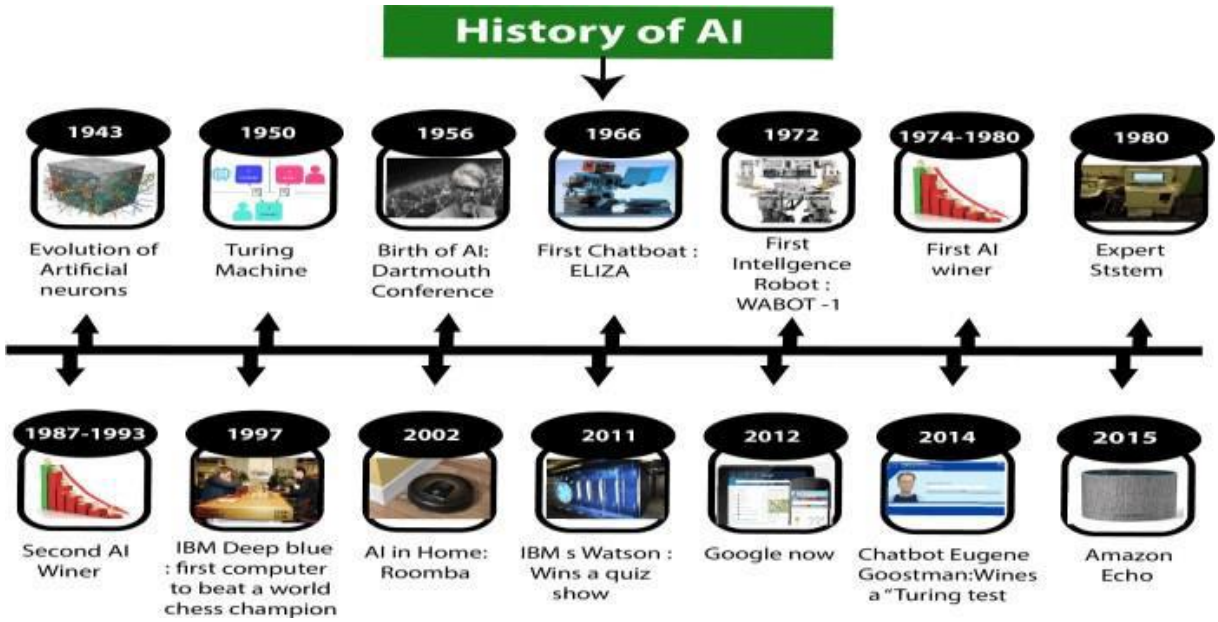
- **Defining artificial intelligence** and describing how it's impacting our rapidly-changing world
- **Simplifying the field of AI** into five core research areas
- **Bringing each research area to life** with anecdotes and examples

Artificial Intelligence is the **automation of activities we normally attribute to human thinking and rationality, such as problem-solving, decision-making, and learning**. AI lives within the intersection of many classic disciplines, including philosophy, neuroscience, behavioral economics, computer science, and mechanical engineering. These interdisciplinary roots help explain why AI has captured our imagination: this burgeoning research field offers a little something for people of all interests and backgrounds. As we evolve our understanding of what AI is and how it works, we simultaneously make headway against the academic community's most challenging research questions:



HISTORY OF ARTIFICIAL INTELLIGENCE:

Artificial Intelligence is not a new word and not a new technology for researchers. This technology is much older than you would imagine. Even there are the myths of Mechanical men in Ancient Greek and Egyptian Myths. 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.

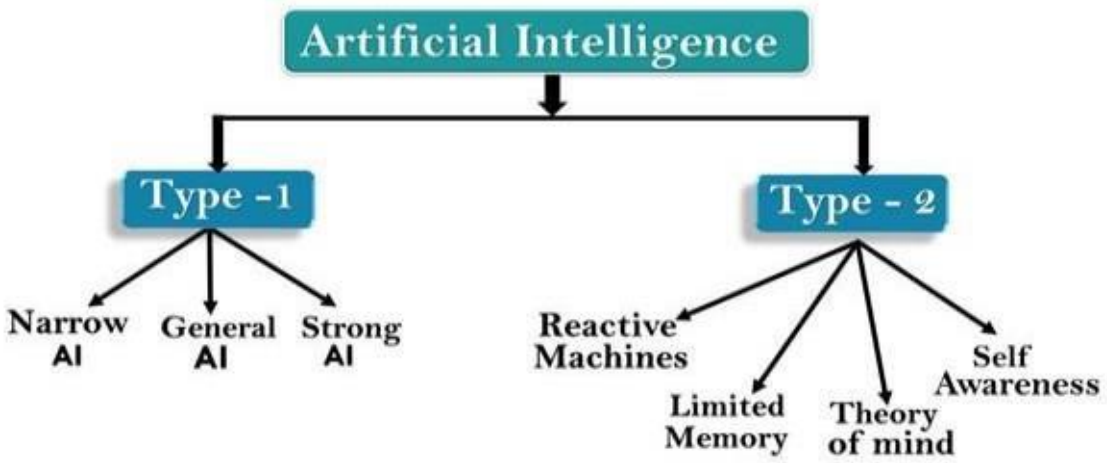
- **Year 2012:** Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
- **Year 2014:** In the year 2014, Chatbot "Eugene Goostman" won a competition in the infamous "Turing test."
- **Year 2018:** The "Project Debater" from IBM debated 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.

THE STATE OF ART:

Types of Artificial Intelligence:

Artificial Intelligence can be divided in various types, there are mainly two types of main categorization which are based on capabilities and based on functionally of AI. Following is flow diagram which explains the types of AI.



AI type-1: Based on Capabilities

1. Weak AI or Narrow AI:

- Narrow AI is a type of AI which is able to perform a dedicated task with intelligence. The most common and currently available AI is Narrow AI in the world of Artificial Intelligence.
- Narrow AI cannot perform beyond its field or limitations, as it is only trained for one specific task. Hence it is also termed as weak AI. Narrow AI can fail in unpredictable ways if it goes beyond its limits.
- Apple Siri is a good example of Narrow AI, but it operates with a limited pre-defined range of functions.
- IBM's Watson supercomputer also comes under Narrow AI, as it uses an Expert system approach combined with Machine learning and natural language processing.
- Some Examples of Narrow AI are playing chess, purchasing suggestions on e-commerce site, self-driving cars, speech recognition, and image recognition.

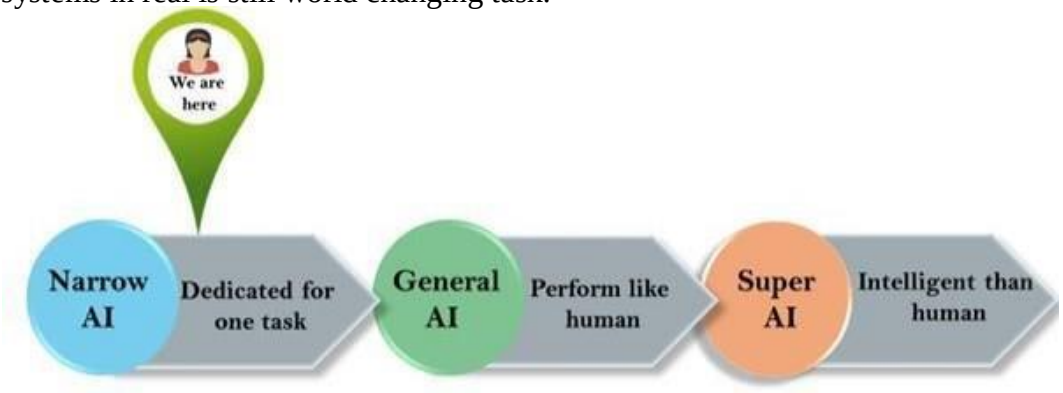
2. General AI:

- General AI is a type of intelligence which could perform any intellectual task with efficiency like a human.
- The idea behind the general AI to make such a system which could be smarter and think like a human by its own.
- Currently, there is no such system exist which could come under general AI and can perform any task as perfect as a human.
- The worldwide researchers are now focused on developing machines with General AI.
- As systems with general AI are still under research and it will take lots of efforts and time to develop such systems.

3. Super AI:

- Super AI is a level of Intelligence of Systems at which machines could surpass human intelligence, and can perform any task better than human with cognitive properties. It is an outcome of general AI.
- Some key characteristics of strong AI include capability include the ability to think, to reason, solve the puzzle, make judgments, plan, learn, and communicate by its own.

- Super AI is still a hypothetical concept of Artificial Intelligence. Development of such systems in real is still world changing task.



Artificial Intelligence type-2: Based on functionality

1. Reactive Machines

- Purely reactive machines are the most basic types of Artificial Intelligence.
- Such AI systems do not store memories or past experiences for future actions.
- These machines only focus on current scenarios and react on it as per possible best action.
- IBM's Deep Blue system is an example of reactive machines.
- Google's Alpha Go is also an example of reactive machines.

2. Limited Memory

- Limited memory machines can store past experiences or some data for a short period of time.
- These machines can use stored data for a limited time period only.
- Self-driving cars are one of the best examples of Limited Memory systems. These cars can store recent speed of nearby cars, the distance of other cars, speed limit, and other information to navigate the road.

3. Theory of Mind

- Theory of Mind AI should understand the human emotions, people, beliefs, and be able to interact socially like humans.

- These types of AI machines are still not developed, but researchers are making lots of efforts and improvement for developing such AI machines.

4. Self-Awareness

- Self-awareness AI is the future of Artificial Intelligence. These machines will be super intelligent, and will have their own consciousness, sentiments, and self-awareness.
- These machines will be smarter than human mind.
- Self-Awareness AI does not exist in reality still and it is a hypothetical concept.

INTELLIGENT AGENTS:

AGENTS IN ARTIFICIAL INTELLIGENCE:

An AI system can be defined as the study of the rational agent and its environment. The agents sense the environment through sensors and act on their environment through actuators. An AI agent can have mental properties such as knowledge, belief, intention, etc.

What is an Agent?

An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting**. An agent can be:

- **Human-Agent:** A human agent has eyes, ears, and other organs which work for sensors and hand, legs, vocal tract work for actuators.
- **Robotic Agent:** A robotic agent can have cameras, infrared range finder, NLP for sensors and various motors for actuators.
- **Software Agent:** Software agent can have keystrokes, file contents as sensory input and act on those inputs and display output on the screen.

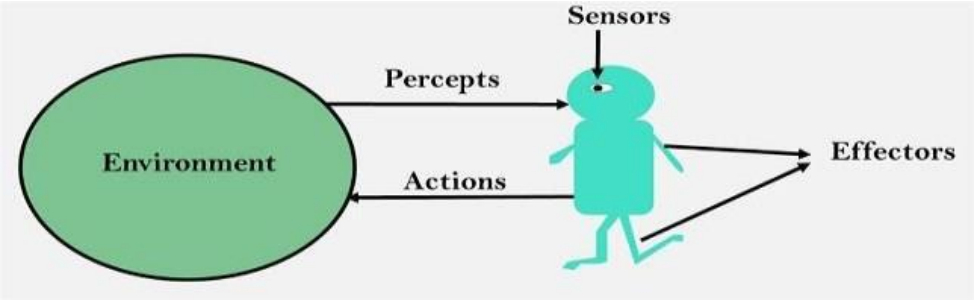
Hence the world around us is full of agents such as thermostat, cellphone, camera, and even we are also agents.

TERMS:

Sensor: Sensor is a device which detects the change in the environment and sends the information to other electronic devices. An agent observes its environment through sensors.

Actuators: Actuators are the component of machines that converts energy into motion. The actuators are only responsible for moving and controlling a system. An actuator can be an electric motor, gears, rails, etc.

Effectors: Effectors are the devices which affect the environment. Effectors can be legs, wheels, arms, fingers, wings, fins, and display screen.



Intelligent Agents:

An intelligent agent is an autonomous entity which acts upon an environment using sensors and actuators for achieving goals. An intelligent agent may learn from the environment to achieve their goals. A thermostat is an example of an intelligent agent.

Following are the main four rules for an AI agent:

- **Rule 1:** An AI agent must have the ability to perceive the environment.
- **Rule 2:** The observation must be used to make decisions.
- **Rule 3:** Decision should result in an action.
- **Rule 4:** The action taken by an AI agent must be a rational action.

Structure of an AI Agent:

The task of AI is to design an agent program which implements the agent function. The structure of an intelligent agent is a combination of architecture and agent program. It can be viewed as:

Agent = Architecture + Agent program

Following are the main three terms involved in the structure of an AI agent:

Architecture: Architecture is machinery that an AI agent executes on.

Agent Function: Agent function is used to map a percept to an action.

$$f:P^* \rightarrow A$$

Agent program: Agent program is an implementation of agent function. An agent program executes on the physical architecture to produce function f.

PEAS Representation

PEAS is a type of model on which an AI agent works upon. When we define an AI agent or rational agent, then we can group its properties under PEAS representation model. It is made up of four words:

- **P:** Performance measure
- **E:** Environment
- **A:** Actuators
- **S:** Sensors

Here performance measure is the objective for the success of an agent's behavior.

Example: PEAS for self-driving cars:

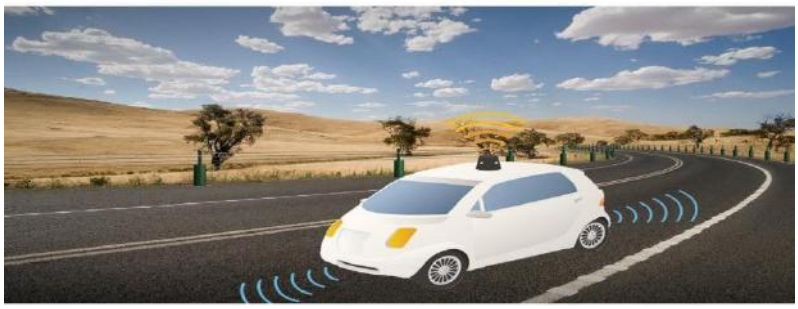
Let's suppose a self-driving car then PEAS representation will be:

Performance: Safety, time, legal drive, comfort

Environment: Roads, other vehicles, road signs, pedestrian

Actuators: Steering, accelerator, brake, signal, horn

Sensors: Camera, GPS, speedometer, odometer, accelerometer, sonar.



Example of Agents with their PEAS representation

Agent	Performance measure	Environment	Actuators	Sensors
1. Medical Diagnose	○ Healthy patient ○ Minimized cost	○ Patient ○ Hospital ○ Staff	○ Tests ○ Treatments	Keyboard (Entry of symptoms)
2. Vacuum Cleaner	○ Cleanness ○ Efficiency ○ Battery life ○ Security	○ Room ○ Table ○ Wood floor ○ Carpet ○ Various obstacles	○ Wheels ○ Brushes ○ Vacuum Extractor	○ Camera ○ Dirt detection sensor ○ Cliff sensor ○ Bump Sensor ○ Infrared Wall Sensor
3. Part -picking Robot	○ Percentage of parts in correct bins.	○ Conveyor belt with parts, ○ Bins	○ Jointed Arms ○ Hand	○ Camera ○ Joint angle sensors.

AGENTS AND ENVIRONMENTS:

- An environment is everything in the world which surrounds the agent, but it is not a part of an agent itself. An environment can be described as a situation in which an agent is present.
- The environment is where agent lives, operate and provide the agent with something to sense and act upon it. An environment is mostly said to be non-feministic.

Features of Environment:

As per Russell and Norvig, an environment can have various features from the point of view of an agent:

1. Fully observable vs Partially Observable
2. Static vs Dynamic
3. Discrete vs Continuous
4. Deterministic vs Stochastic
5. Single-agent vs Multi-agent
6. Episodic vs sequential
7. Known vs Unknown
8. Accessible vs Inaccessible

1. Fully observable vs Partially Observable:

- If an agent sensor can sense or access the complete state of an environment at each point of time then it is **a fully observable** environment, else it is **partially observable**.
- A fully observable environment is easy as there is no need to maintain the internal state to keep track history of the world.
- An agent with no sensors in all environments then such an environment is called as **unobservable**.

2. Deterministic vs Stochastic:

- If an agent's current state and selected action can completely determine the next state of the environment, then such environment is called a deterministic environment.
- A stochastic environment is random in nature and cannot be determined completely by an agent.
- In a deterministic, fully observable environment, agent does not need to worry about uncertainty.

3. Episodic vs Sequential:

- In an episodic environment, there is a series of one-shot actions, and only the current percept is required for the action.
- However, in Sequential environment, an agent requires memory of past actions to determine the next best actions.

4. Single-agent vs Multi-agent

- If only one agent is involved in an environment, and operating by itself then such an environment is called single agent environment.
- However, if multiple agents are operating in an environment, then such an environment is called a multi-agent environment.
- The agent design problems in the multi-agent environment are different from single agent environment.

5. Static vs Dynamic:

- If the environment can change itself while an agent is deliberating then such environment is called a dynamic environment else it is called a static environment.
- Static environments are easy to deal because an agent does not need to continue looking at the world while deciding for an action.
- However for dynamic environment, agents need to keep looking at the world at each action.

- Taxi driving is an example of a dynamic environment whereas Crossword puzzles are an example of a static environment.

6. Discrete vs Continuous:

- If in an environment there are a finite number of percepts and actions that can be performed within it, then such an environment is called a discrete environment else it is called continuous environment.
- A chess game comes under discrete environment as there is a finite number of moves that can be performed.
- A self-driving car is an example of a continuous environment.

7. Known vs Unknown

- Known and unknown is not actually a feature of an environment, but it is an agent's state of knowledge to perform an action.
- In a known environment, the results for all actions are known to the agent. While in unknown environment, agent needs to learn how it works in order to perform an action.
- It is quite possible that a known environment to be partially observable and an Unknown environment to be fully observable.

8. Accessible vs Inaccessible

- If an agent can obtain complete and accurate information about the state's environment, then such an environment is called an Accessible environment else it is called inaccessible.
- An empty room whose state can be defined by its temperature is an example of an accessible environment.
- Information about an event on earth is an example of Inaccessible environment.

Good Behavior: The Concept of Rationality:

Rational Agent:

- A rational agent is an agent which has clear preference, models uncertainty, and acts in a way to maximize its performance measure with all possible actions.
- A rational agent is said to perform the right things. AI is about creating rational agents to use for game theory and decision theory for various real-world scenarios.
- For an AI agent, the rational action is most important because in AI reinforcement learning algorithm, for each best possible action, agent gets the positive reward and for each wrong action, an agent gets a negative reward.

Rationality:

The rationality of an agent is measured by its performance measure. Rationality can be judged on the basis of following points:

- Performance measure which defines the success criterion.
- Agent prior knowledge of its environment.
- Best possible actions that an agent can perform.
- The sequence of percepts.

****

UNIT- II

SOLVING PROBLEMS BY SEARCHING

In previous chapter, we saw that simple reflex agents are unable to plan ahead. They are limited in what they can do because their actions are determined only by the current percept. Furthermore, they have no knowledge of what their actions do nor of what they are trying to achieve.

In this chapter, we describe one kind of goal-based agent called a problem-solving agent. Problem-solving agents decide what to do by finding sequences of actions that lead to desirable states. We discuss informally how the agent can formulate an appropriate view of the problem it faces. The problem type that results from the formulation process will depend on the knowledge available to the agent: principally, whether it knows the current state and the outcomes of actions. We then define more precisely the elements that constitute a "problem" and its "solution," and give several examples to illustrate these definitions. Given precise definitions of problems, it is relatively straightforward to construct a search process for finding solutions.

PROBLEM SOLVING AGENTS

Intelligent agents are supposed to act in such a way that the environment goes through a sequence of states that maximizes the performance measure. In its full generality, this specification is difficult to translate into a successful agent design. As we mentioned in previous chapter, the task is somewhat simplified if the agent can adopt a goal and aim to satisfy it. Let us first look at how and why an agent might do this.

Goal formulation, based on the current situation, is the first step in problem solving. As well as formulating a goal, the agent may wish to decide on some other factors that affect the desirability of different ways of achieving the goal.

Problem formulation is the process of deciding what actions and states to consider, and follows goal formulation. We will discuss this process in more detail. For now, let us assume that the agent will consider actions at the level of driving from one major town to another. The states it will consider therefore correspond to being in a particular town.

In general, then, an agent with several immediate options of unknown value can decide what to do by first examining; different possible sequences of actions that lead to states of known value, and then choosing the best one. This process of looking for such a sequence is called ***search.***

A search algorithm takes a problem as input and returns a solution in the form of an action sequence. Once a solution is found, the actions it recommends can be carried out. This is called the ***execution phase.***

Thus, we have a simple "formulate, search, execute" design for the agent, as shown in Figure 3.1. After formulating a goal and a problem to solve, the agent calls a search procedure to solve it. It then uses the solution to guide its actions, doing whatever the solution recommends as the next thing to do, and then removing that step from the sequence. Once the solution has been executed, the agent will find a new goal.

EXAMPLE PROBLEMS

The range of task environments that can be characterized by well-defined problems is vast. We can distinguish between so-called, toy problems, which are intended to illustrate or exercise various problem-solving methods, and so-called real-world problems, which tend to be more difficult and whose solutions people actually care about. In this section, we will give examples of both. By nature, toy problems can be given a concise, exact description. This means that they can be easily used by different researchers to compare the performance of algorithms. Real-world problems, on the other hand, tend not to have a single agreed-upon description, but we will attempt to give the general flavor of their formulations.

1. Toy Problems

The 8-puzzIe

The 8-puzzle, an instance of which is shown in Figure 3.4, consists of a 3x3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The object is to reach the configuration shown on the right of the figure. One important trick is to notice that rather than use operators such as "move the 3 tile into the blank space," it is more sensible to have operators such as "the blank space changes places with the tile to its left." This is because there are fewer of the latter kind of operator.

This leads us to the following formulation:

- States: a state description specifies the location of each of the eight tiles in one of the nine squares. For efficiency, it is useful to include the location of the blank.
- Operators: blank moves left, right, up, or down.
- Goal test: state matches the goal configuration shown in Figure 3.4.
- Path cost: each step costs 1, so the path cost is just the length of the path.

The 8-puzzle belongs to the family of sliding-block puzzles. This general class is known to be NP-complete, so one does not expect to find methods significantly better than the search algorithms described in this chapter and the next. The 8-puzzle and its larger cousin, the 15-puzzle, are the standard test problems for new search algorithms in Al.

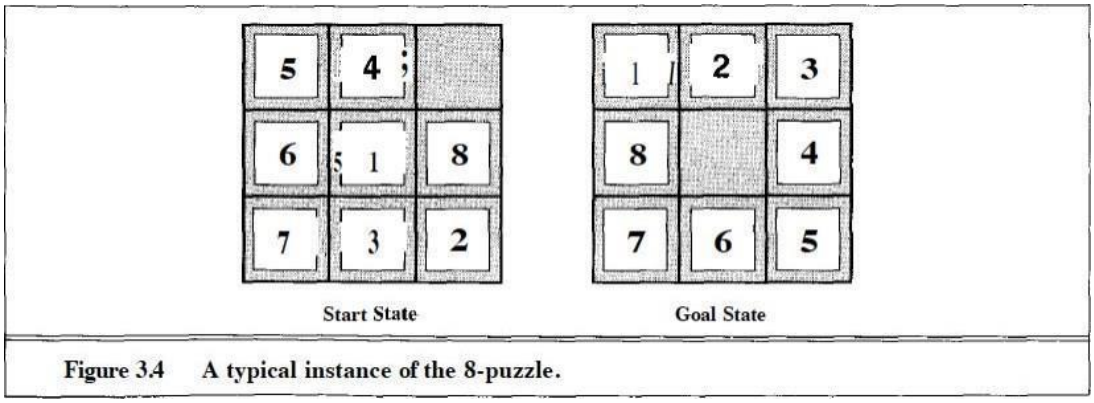


Figure 3.4 A typical instance of the 8-puzzle.

The 8-queens problem

The **eight queens puzzle** is the **problem** of placing **eight** chess **queens** on an **8×8** chessboard so that no two **queens** threaten each other; thus, a solution requires that no two **queens** share the same row, column, or diagonal.

The 8-queens problem can be defined as follows: Place 8 queens on an (8 by 8) chess board such that none of the queens attacks any of the others. A configuration of 8 queens on the board is shown in figure 1, but this does not represent a solution as the queen in the first column is on the same diagonal as the queen in the last column.

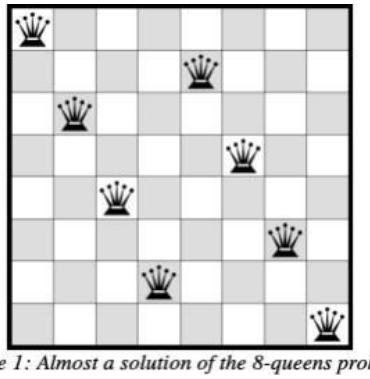


Figure 1: Almost a solution of the 8-queens problem

Although efficient special-purpose algorithms exist for this problem and the whole n queens family, it remains an interesting test problem for search algorithms. There are two main kinds of formulation. The incremental formulation involves placing queens one by one, whereas the complete-state formulation starts with all 8 queens on the board and moves them around. In either case, the path cost is of no interest because only the final state counts; algorithms are thus compared only on search cost. Thus, we have the following goal test and path cost:

- Goal test: 8 queens on board, none attacked.
- Path cost: zero.

There are also different possible states and operators. Consider the following simple-minded formulation:

- States: any arrangement of 0 to 8 queens on board.
- Operators: add a queen to any square.

In this formulation, we have 648 possible sequences to investigate. A more sensible choice would use the fact that placing a queen where it is already attacked cannot work, because subsequent placings of other queens will not undo the attack. So we might try the following:

- States: arrangements of 0 to 8 queens with none attacked.
- Operators: place a queen in the left-most empty column such that it is not attacked by any other queen.

It is easy to see that the actions given can generate only states with no attacks; but sometimes no actions will be possible. For example, after making the first seven choices (left-to-right) in Figure 1, there is no action available in this formulation. The search process must try another choice. A quick calculation shows that there are only 2057 possible sequences to investigate. The right formulation makes a big difference to the size of the search space. Similar considerations apply for a complete-state formulation. For example, we could set the problem up as follows:

- States: arrangements of 8 queens, one in each column.
- Operators: move any attacked queen to another square in the same column.

This formulation would allow the algorithm to find a solution eventually, but it would be better to move to an unattacked square if possible.

2. Real-world problems

Route finding

We have already seen how route finding is defined in terms of specified locations and transitions! along links between them. Route-finding algorithms are used in a variety of applications, such! as routing in computer networks, automated travel advisory systems, and airline travel planning! systems. The last application is somewhat more complicated, because airline travel has a very complex path cost, in terms of money, seat quality, time of day, type of airplane, frequent-flyer mileage awards, and so on. Furthermore, the actions in the problem do not have completely known outcomes: flights can be late or overbooked, connections can be missed, and fog or emergency maintenance can cause delays.

Touring and travelling salesperson problems

Consider the problem, "Visit every city in Figure 3.3 at least once, starting and ending in Bucharest." This seems very similar to route finding, because the operators still correspond to trips between adjacent cities. But for this problem, the state space must record more information. In addition to the agent's location, each state must keep track of the set of cities the agent has visited. So the initial state would be "In Bucharest; visited {Bucharest}," a typical intermediate state would be "In Vaslui; visited {Bucharest,Urziceni,Vaslui}," and the goal test would check if the agent is in Bucharest and that all 20 cities have been visited. The travelling salesperson problem (TSP) is a famous touring problem in which each city must be visited exactly once. The aim is to find the shortest tour.The problem is NP-hard (Karp, 1972), but an enormous amount of effort has been expended to improve the capabilities of TSP algorithms. In addition to planning trips for travelling salespersons, these algorithms have been used for tasks such as planning movements of automatic circuit board drills.

VLSI Layout

The design of silicon chips is one of the most complex engineering design tasks currently undertaken, and we can give only a brief sketch here. A typical VLSI chip can have as many as a million gates, and the positioning and connections of every gate are crucial to the successful operation of the chip. Computer-aided design tools are used in every phase of the process. Two of the most difficult tasks are cell layout and channel routing. These come after the components and connections of the circuit have been fixed; the purpose is to lay out the circuit on the chip so as to minimize area and connection lengths, thereby maximizing speed. In cell layout, the primitive components of the circuit are grouped into cells, each of which performs some recognized function. Each cell has a fixed footprint (size and shape) and requires a certain number of connections to each of the other cells. The aim is to place the cells on the chip so that they do not overlap and so that there is room for the connecting wires to be placed between the cells. Channel routing finds a specific route for each wire using the gaps between the cells. These search problems are extremely complex, but definitely worth solving.

Robot navigation

Robot navigation is a generalization of the route-finding problem described earlier. Rather than a discrete set of routes, a robot can move in a continuous space with (in principle) an infinite set of possible actions and states. For a simple, circular robot moving on a flat

surface, the space is essentially two-dimensional. When the robot has arms and legs that must also be controlled, the search space becomes many-dimensional. Advanced techniques are required just to make the search space finite.

SEARCHING FOR SOLUTIONS

We have seen how to define a problem, and how to recognize a solution. The remaining part— finding a solution—is done by a search through the state space. The idea is to maintain and extend a set of partial solution sequences. In this section, we show how to generate these sequences and how to keep track of them using suitable data structures.

Generating action sequences

To solve the route-finding problem from Arad to Bucharest, for example, we start off with just the initial state, Arad. The first step is to test if this is a goal state. Clearly it is not, but it is important to check so that we can solve trick problems like "starting in Arad, get to Arad." Because this is ; not a goal state, we need to consider some other states. This is done by applying the operators; to the current state, thereby generating a new set of states. The process is called expanding the state. In this case, we get three new states, "in Sibiu," "in Timisoara," and "in Zerind," because there is a direct one-step route from Arad to these three cities. If there were only one possibility; we would just take it and continue. But whenever there are multiple possibilities, we must make a choice about which one to consider further.

This is the essence of search—choosing one option and putting the others aside for later, in ' case the first choice does not lead to a solution. Suppose we choose Zerind. We check to see if it is a goal state (it is not), and then expand it to get "in Arad" and "in Oradea." We can then choose any of these two, or go back and choose Sibiu or Timisoara. We continue choosing, testing, and expanding until a solution is found, or until there are no more states to be expanded. The choice of which state to expand first is determined by the search strategy.

It is helpful to think of the search process as building up a search tree that is superimposed over the state space. The root of the search tree is a search node corresponding to the initial state. The leaf nodes of the tree correspond to states that do not have successors in the tree, either because they have not been expanded yet, or because they were expanded, but generated the empty set. At each step, the search algorithm chooses one leaf node to expand. Figure 3.8 shows some of the expansions in the search tree for route finding from Arad to Bucharest. The general search algorithm is described informally in Figure 3.9.

It is important to distinguish between the state space and the search tree. For the route finding problem, there are only 20 states in the state space, one for each city. But there are an infinite number of paths in this state space, so the search tree has an infinite number of nodes. For example, in Figure 3.8, the branch Arad-Sibiu-Arad continues Arad-Sibiu-Arad-Sibiu-Arad, and so on, indefinitely. Obviously, a good search algorithm avoids following such paths.

Data structures for search trees

There are many ways to represent nodes, but in this chapter, we will assume a node is a data structure with five components:

- the state in the state space to which the node corresponds;
- the node in the search tree that generated this node (this is called the parent node);
- the operator that was applied to generate the node;
- the number of nodes on the path from the root to this node (the depth of the node);
- the path cost of the path from the initial state to the node.

The node data type is thus:
datatype node
components: STATE, PARENT-NODE, OPERATOR, DEPTH, PATH-COST

It is important to remember the distinction between nodes and states. A node is a bookkeeping data structure used to represent the search tree for a particular problem instance as generated by a particular algorithm. A state represents a configuration (or set of configurations) of the world. Thus, nodes have depths and parents, whereas states do not. (Furthermore, it is quite possible for two different nodes to contain the same state, if that state is generated via two different sequences of actions.) The EXPAND function is responsible for calculating each of the components of the nodes it generates.

SEARCH STRATEGIES

Search Algorithm Terminologies

Search:

Searching is a step by step procedure to solve a search-problem in a given search space.

A search problem can have three main factors:

1. **Search Space:** Search space represents a set of possible solutions, which a system may have.
2. **Start State:** It is a state from where agent begins **the search**.
3. **Goal test:** It is a function which observe the current state and returns whether the goal state is achieved or not.

- **Search tree:** A tree representation of search problem is called Search tree. The root of the search tree is the root node which is corresponding to the initial state.
- **Actions:** It gives the description of all the available actions to the agent.
- **Transition model:** A description of what each action do, can be represented as a transition model.
- **Path Cost:** It is a function which assigns a numeric cost to each path.
- **Solution:** It is an action sequence which leads from the start node to the goal node.
- **Optimal Solution:** If a solution has the lowest cost among all solutions.

Properties of Search Algorithms:

Following are the four essential properties of search algorithms to compare the efficiency of these algorithms:

Completeness: A search algorithm is said to be complete if it guarantees to return a solution if at least any solution exists for any random input.

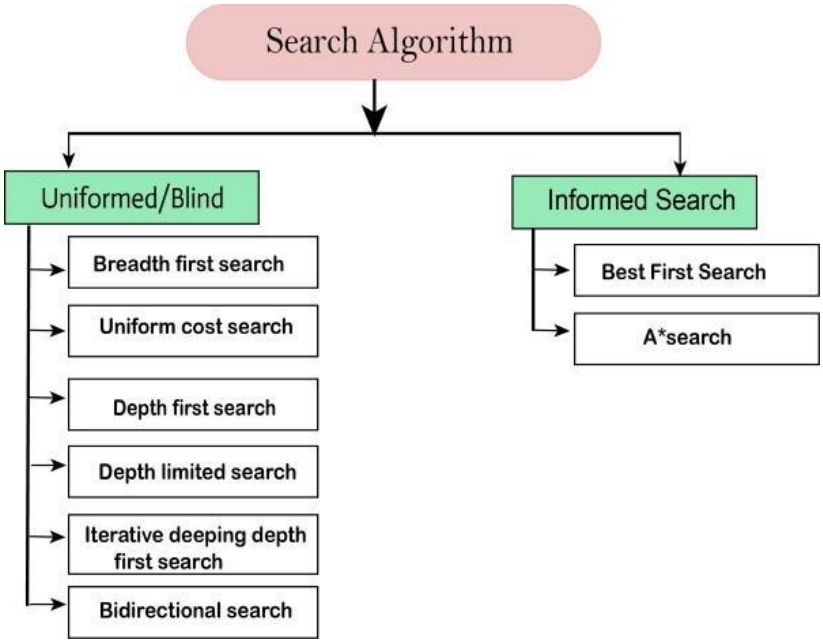
Optimality: If a solution found for an algorithm is guaranteed to be the best solution (lowest path cost) among all other solutions, then such a solution for is said to be an optimal solution.

Time Complexity: Time complexity is a measure of time for an algorithm to complete its task.

Space Complexity: It is the maximum storage space required at any point during the search, as the complexity of the problem.

Types of search algorithms

Based on the search problems we can classify the search algorithms into uninformed (Blind search) search and informed search (Heuristic search) algorithms.



1. UNINFORMED SEARCH (Blind Search):

The term means that they have no information about the number of steps or the path cost from the current state to the goal—all they can do is distinguish a goal state from a nongoal state. Uninformed search applies a way in which search tree is searched without any information about the search space like initial state operators and test for the goal, so it is also called blind search. It examines each node of the tree until it achieves the goal node.

It can be divided into five main types:

- Breadth-first search
- Uniform cost search
- Depth-first search
- Iterative deepening depth-first search
- Bidirectional Search

1. Breadth-first Search:

- Breadth-first search is the most common search strategy for traversing a tree or graph. This algorithm searches breadthwise in a tree or graph, so it is called breadth-first search.
- BFS algorithm starts searching from the root node of the tree and expands all successor node at the current level before moving to nodes of next level.
- The breadth-first search algorithm is an example of a general-graph search algorithm.
- Breadth-first search implemented using FIFO queue data structure.

Advantages:

- BFS will provide a solution if any solution exists.
- If there are more than one solutions for a given problem, then BFS will provide the minimal solution which requires the least number of steps.

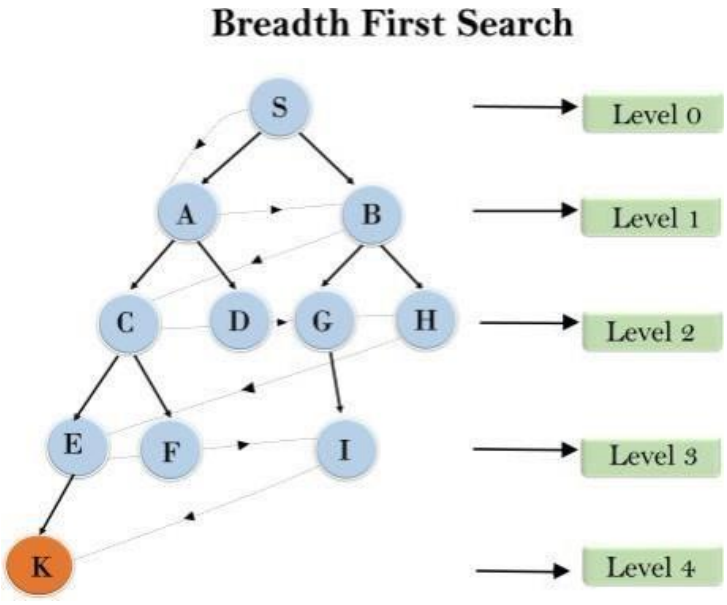
Disadvantages:

- t requires lots of memory since each level of the tree must be saved into memory to expand the next level.
- BFS needs lots of time if the solution is far away from the root node.

Example

In the below tree structure, we have shown the traversing of the tree using BFS algorithm from the root node S to goal node K. BFS search algorithm traverse in layers, so it will follow the path which is shown by the dotted arrow, and the traversed path will be:

S---> A--->B---->C--->D---->G--->H--->E---->F---->I >K



Time Complexity:

Time Complexity of BFS algorithm can be obtained by the number of nodes traversed in BFS until the shallowest Node. Where the d= depth of shallowest solution and b is a node at every state.

$T(b) = 1+b^2+b^3+ \quad + b^d = O(b^d)$

Space Complexity:

Space complexity of BFS algorithm is given by the Memory size of frontier which is $O(b^d)$.

Completeness:

BFS is complete, which means if the shallowest goal node is at some finite depth, then BFS will find a solution.

Optimality:

BFS is optimal if path cost is a non-decreasing function of the depth of the node.

2. Depth-first Search

- Depth-first search isa recursive algorithm for traversing a tree or graph data structure.
- It is called the depth-first search because it starts from the root node and follows each path to its greatest depth node before moving to the next path.
- DFS uses a stack data structure for its implementation.
- The process of the DFS algorithm is similar to the BFS algorithm.

Advantage:

- DFS requires very less memory as it only needs to store a stack of the nodes on the path from root node to the current node.
- It takes less time to reach to the goal node than BFS algorithm (if it traverses in the right path).

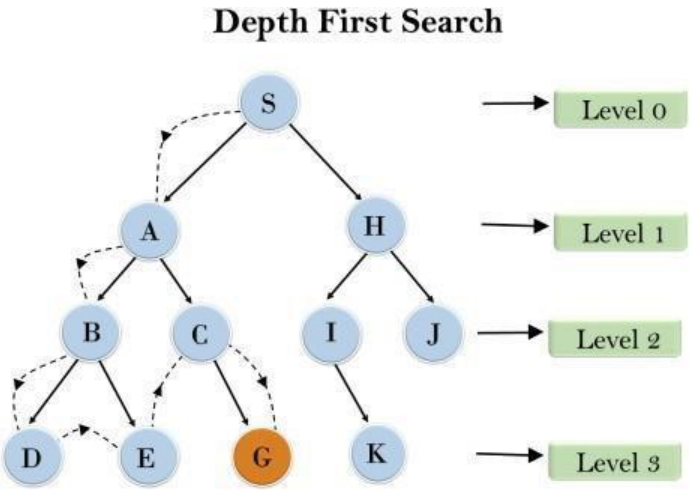
Disadvantage:

- There is the possibility that many states keep re-occurring, and there is no guarantee of finding the solution.
- DFS algorithm goes for deep down searching and sometime it may go to the infinite loop.

Example

In the below search tree, we have shown the flow of depth-first search, and it will follow the order as:

Root node--->Left node----> right node.



It will start searching from root node S, and traverse A, then B, then D and E, after traversing E, it will backtrack the tree as E has no other successor and still goal node is not found. After backtracking it will traverse node C and then G, and here it will terminate as it found goal node.

Completeness: DFS search algorithm is complete within finite state space as it will expand every node within a limited search tree.
Time Complexity: Time complexity of DFS will be equivalent to the node traversed by the algorithm. It is given by:
$T(n)= 1+ n^2+ n^3 +.....+ n^m=O(n^m)$
Where, m= maximum depth of any node and this can be much larger than d (Shallowest solution depth)
Space Complexity: DFS algorithm needs to store only single path from the root node, hence space complexity of DFS is equivalent to the size of the fringe set, which is **O(bm)**.
Optimal: DFS search algorithm is non-optimal, as it may generate a large number of steps or high cost to reach to the goal node.

3. Depth-Limited Search Algorithm:

A depth-limited search algorithm is similar to depth-first search with a predetermined limit. Depth-limited search can solve the drawback of the infinite path in the Depth-first search. In this algorithm, the node at the depth limit will treat as it has no successor nodes further.
Depth-limited search can be terminated with two Conditions of failure:

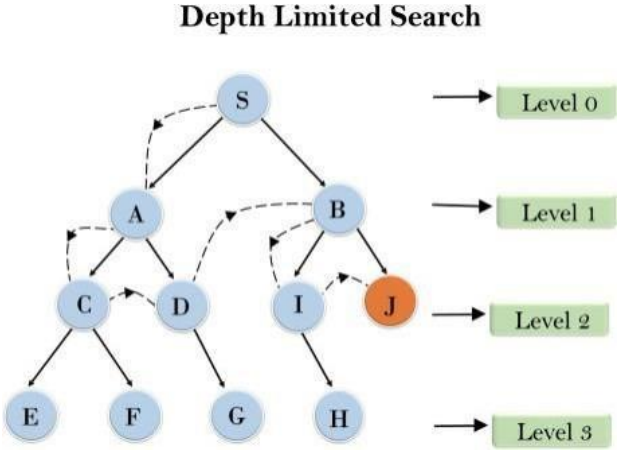
- Standard failure value: It indicates that problem does not have any solution.
- Cutoff failure value: It defines no solution for the problem within a given depth limit.

Advantages:
Depth-limited search is Memory efficient.

Disadvantages:

- Depth-limited search also has a disadvantage of incompleteness.
- It may not be optimal if the problem has more than one solution.

Example



Completeness: DLS search algorithm is complete if the solution is above the depth-limit.
Time Complexity: Time complexity of DLS algorithm is **O(b^{ℓ})**.
Space Complexity: Space complexity of DLS algorithm is O**(b×ℓ)**.
Optimal: Depth-limited search can be viewed as a special case of DFS, and it is also not optimal even if ℓ>d.

4. Uniform-cost Search Algorithm:

Uniform-cost search is a searching algorithm used for traversing a weighted tree or graph. This algorithm comes into play when a different cost is available for each edge. The primary goal of the uniform-cost search is to find a path to the goal node which has the lowest cumulative cost. Uniform-cost search expands nodes according to their path costs form the root node. It can be used to solve any graph/tree where the optimal cost is in demand. A uniform-cost search algorithm is implemented by the priority queue. It gives maximum priority to the lowest cumulative cost. Uniform cost search is equivalent to BFS algorithm if the path cost of all edges is the same.

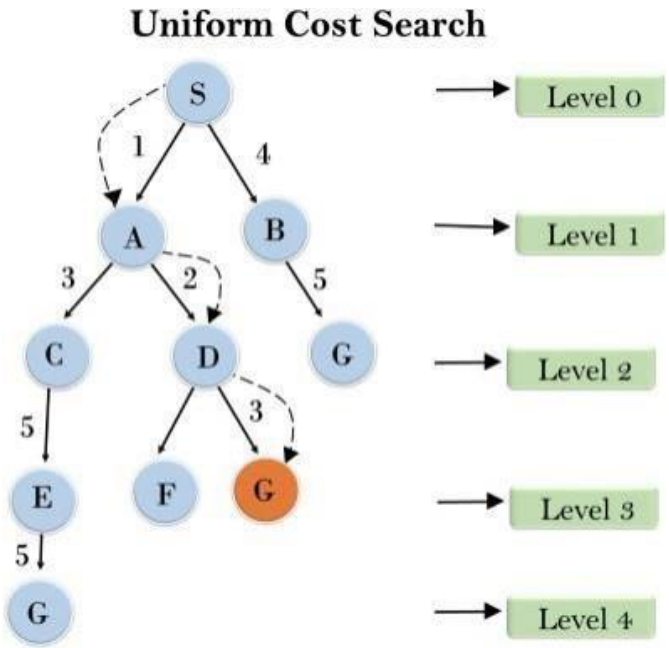
Advantages:

- Uniform cost search is optimal because at every state the path with the least cost is chosen.
-

Disadvantages:

- It does not care about the number of steps involve in searching and only concerned about path cost. Due to which this algorithm may be stuck in an infinite loop.

Example



Completeness:

Uniform-cost search is complete, such as if there is a solution, UCS will find it.

Time Complexity:

Let C* **is Cost of the optimal solution**, and **ε** is each step to get closer to the goal node. Then the number of steps is = C*/ε+1. Here we have taken +1, as we start from state 0 and end to C*/ε.

Hence, the worst-case time complexity of Uniform-cost search is$\mathbf{O(b^{1 + [C*/\varepsilon]})}$/.

Space Complexity:

The same logic is for space complexity so, the worst-case space complexity of Uniform-cost search is $\mathbf{O(b^{1 + [C*/\varepsilon]})}$.

Optimal:

Uniform-cost search is always optimal as it only selects a path with the lowest path cost.

5. Iterative deepening depth-first Search:

- The iterative deepening algorithm is a combination of DFS and BFS algorithms. This search algorithm finds out the best depth limit and does it by gradually increasing the limit until a goal is found.
- This algorithm performs depth-first search up to a certain "depth limit", and it keeps increasing the depth limit after each iteration until the goal node is found.
- This Search algorithm combines the benefits of Breadth-first search's fast search and depth-first search's memory efficiency.
- The iterative search algorithm is useful uninformed search when search space is large, and depth of goal node is unknown.

Advantages:

- It combines the benefits of BFS and DFS search algorithm in terms of fast search and memory efficiency.

Disadvantages:

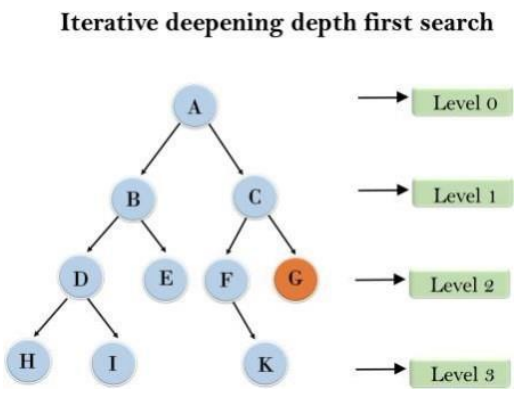
- The main drawback of IDDFS is that it repeats all the work of the previous phase.

Example

Following tree structure is showing the iterative deepening depth-first search.

IDDFS algorithm performs various iterations until it does not find the goal node.

The iteration performed by the algorithm is given as:



1'st Iteration ----> A
2'nd Iteration ---> A, B, C
3'rd Iteration----- >A, B, D, E, C, F, G
4'th Iteration----- >A, B, D, H, I, E, C, F, K, G
In the fourth iteration, the algorithm will find the goal node.

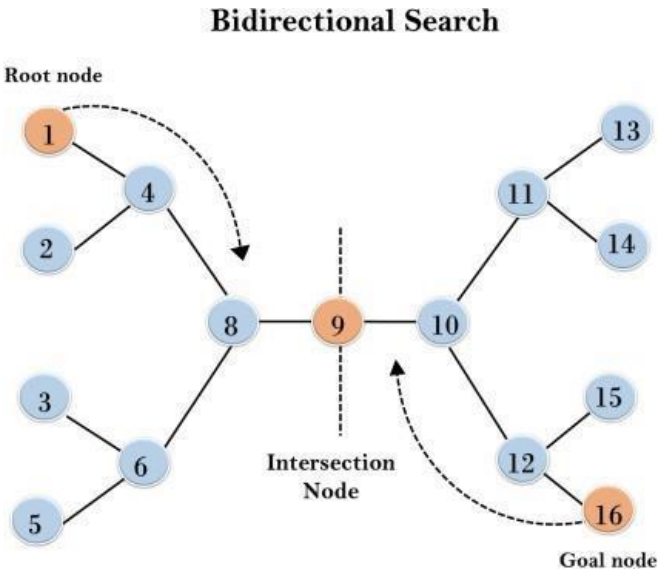
Completeness:
This algorithm is complete is if the branching factor is finite.
Time Complexity:
Let's suppose b is the branching factor and depth is d then the worst-case time complexity is **$O(b^d)$**.
Space Complexity:
The space complexity of IDDFS will be **O(bd)**.
Optimal:
IDDFS algorithm is optimal if path cost is a non- decreasing function of the depth of the node.

6. Bidirectional Search Algorithm
Bidirectional search algorithm runs two simultaneous searches, one form initial state called as forward-search and other from goal node called as backward-search, to find the goal node. Bidirectional search replaces one single search graph with two small subgraphs in which one starts the search from an initial vertex and other starts from goal vertex. The search stops when these two graphs intersect each other.
Bidirectional search can use search techniques such as BFS, DFS, DLS, etc.

Advantages:
- Bidirectional search is fast.
- Bidirectional search requires less memory

Disadvantages:
- Implementation of the bidirectional search tree is difficult.
- In bidirectional search, one should know the goal state in advance.

Example
In the below search tree, bidirectional search algorithm is applied. This algorithm divides one graph/tree into two sub-graphs. It starts traversing from node 1 in the forward direction and starts from goal node 16 in the backward direction.
The algorithm terminates at node 9 where two searches meet.



Completeness: Bidirectional Search is complete if we use BFS in both searches.
Time Complexity: Time complexity of bidirectional search using BFS is $O(b^d)$.
Space Complexity: Space complexity of bidirectional search is $O(b^d)$.
Optimal: Bidirectional search is Optimal.

2. INFORMED SEARCH (Heuristic Search)

- Informed search algorithms use domain knowledge.
- In an informed search, problem information is available which can guide the search. Informed search strategies can find a solution more efficiently than an uninformed search strategy.
- Informed search is also called a Heuristic search.
- A heuristic is a way which might not always be guaranteed for best solutions but guaranteed to find a good solution in reasonable time.

Heuristics function:
Heuristic is a function which is used in Informed Search, and it finds the most promising path. It takes the current state of the agent as its input and produces the estimation of how close agent is from the goal. The heuristic method, however, might not always give the best solution, but it guaranteed to find a good solution in reasonable time. Heuristic function estimates how close a state is to the goal. It is represented by h(n), and it calculates the cost of an optimal path between the pair of states. The value of the heuristic function is always positive.

Admissibility of the heuristic function is given as:

1. $h(n) <= h^*(n)$

Here h(n) is heuristic cost, and h*(n) is the estimated cost.

Hence heuristic cost should be less than or equal to the estimated cost.

Pure Heuristic Search:

Pure heuristic search is the simplest form of heuristic search algorithms. It expands nodes based on their heuristic value h(n). It maintains two lists, OPEN and CLOSED list. In the CLOSED list, it places those nodes which have already expanded and in the OPEN list, it places nodes which have yet not been expanded.

On each iteration, each node n with the lowest heuristic value is expanded and generates all its successors and n is placed to the closed list. The algorithm continues unit a goal state is found.

In the informed search we will discuss two main algorithms which are given below:

- **Best First Search Algorithm(Greedy search)**
- **A* Search Algorithm**

1. Best-first Search Algorithm (Greedy Search):

Greedy best-first search algorithm always selects the path which appears best at that moment. It is the combination of depth-first search and breadth-first search algorithms. It uses the heuristic function and search. Best-first search allows us to take the advantages of both algorithms. With the help of best-first search, at each step, we can choose the most promising node. In the best first search algorithm, we expand the node which is closest to the goal node and the closest cost is estimated by heuristic function, i.e.

$f(n)= g(n)$.

Were, $h(n)$= estimated cost from node n to the goal.

The greedy best first algorithm is implemented by the priority queue.

Best first search algorithm

- **Step 1:** Place the starting node into the OPEN list.
- **Step 2:** If the OPEN list is empty, Stop and return failure.
- **Step 3:** Remove the node n, from the OPEN list which has the lowest value of h(n), and places it in the CLOSED list.
- **Step 4:** Expand the node n, and generate the successors of node n.
- **Step 5:** Check each successor of node n, and find whether any node is a goal node or not. If any successor node is goal node, then return success and terminate the search, else proceed to Step 6.
- **Step 6:** For each successor node, algorithm checks for evaluation function f(n), and then check if the node has been in either OPEN or CLOSED list. If the node has not been in both list, then add it to the OPEN list.
- **Step 7:** Return to Step 2.

Advantages:

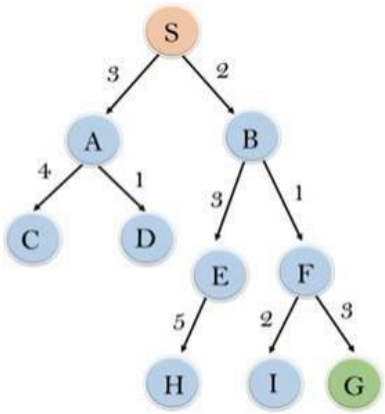
- Best first search can switch between BFS and DFS by gaining the advantages of both the algorithms.
- This algorithm is more efficient than BFS and DFS algorithms.

Disadvantages:

- It can behave as an unguided depth-first search in the worst case scenario.
- It can get stuck in a loop as DFS.
- This algorithm is not optimal.

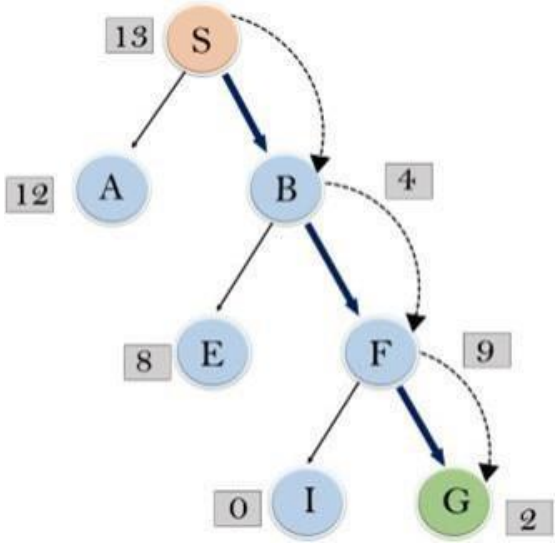
Example

Consider the below search problem, and we will traverse it using greedy best-first search. At each iteration, each node is expanded using evaluation function f(n)=h(n) , which is given in the below table.



node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

In this search example, we are using two lists which are **OPEN** and **CLOSED** Lists. Following are the iteration for traversing the above example.



Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

Iteration 1: Open [A], Closed [S, B]

Iteration 2: Open [E, F, A], Closed [S, B]

: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]

: Open [I, E, A], Closed [S, B, F, G]

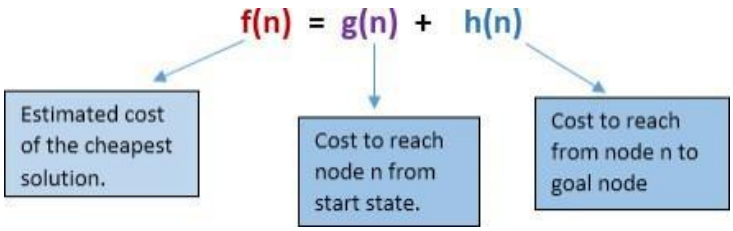
Hence the final solution path will be: **S----> B----->F ---> G**

Time Complexity: The worst case time complexity of Greedy best first search is $O(b^m)$.
Space Complexity: The worst case space complexity of Greedy best first search is $O(b^m)$. Where, m is the maximum depth of the search space.
Complete: Greedy best-first search is also incomplete, even if the given state space is finite.
Optimal: Greedy best first search algorithm is not optimal.

2. A* Search Algorithm:

A* search is the most commonly known form of best-first search. It uses heuristic function h(n), and cost to reach the node n from the start state g(n). It has combined features of UCS and greedy best-first search, by which it solve the problem efficiently. A* search algorithm finds the shortest path through the search space using the heuristic function. This search algorithm expands less search tree and provides optimal result faster. A* algorithm is similar to UCS except that it uses g(n)+h(n) instead of g(n).

In A* search algorithm, we use search heuristic as well as the cost to reach the node. Hence we can combine both costs as following, and this sum is called as a **fitness number**.



Algorithm of A* search:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function (g+h), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute evaluation function for n' and place into Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to **Step 2**.

Advantages:

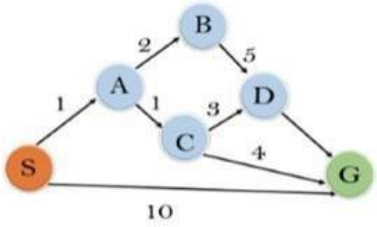
- A* search algorithm is the best algorithm than other search algorithms.
- A* search algorithm is optimal and complete.
- This algorithm can solve very complex problems.

Disadvantages:

- It does not always produce the shortest path as it mostly based on heuristics and approximation.
- A* search algorithm has some complexity issues.
- The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

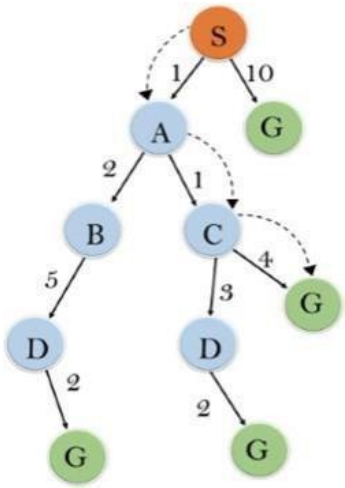
Example

In this example, we will traverse the given graph using the A* algorithm. The heuristic value of all states is given in the below table so we will calculate the f(n) of each state using the formula f(n)= g(n) + h(n), where g(n) is the cost to reach any node from start state.
Here we will use OPEN and CLOSED list.



State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution:



Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.

If the heuristic function is admissible, then A* tree search will always find the least cost path.

Time Complexity: The time complexity of A* search algorithm depends on heuristic function, and the number of nodes expanded is exponential to the depth of solution d. So the time complexity is O(b^d), where b is the branching factor.

Space Complexity: The space complexity of A* search algorithm is **O(b^d)**

****

ARTIFICIAL LANGUAGE

UNIT-III

REINFORCEMENT LEARNING

SYLLABUS:

Reinforcement Learning: Introduction, Passive Reinforcement Learning, Active Reinforcement Learning, Generalization in Reinforcement Learning, Policy Search, applications of RL

Natural Language Processing: Language Models, Text Classification, Information Retrieval, Information Extraction.

****

Reinforcement Learning:

Introduction:

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly progra1mned.

Classification of Machine Learning-

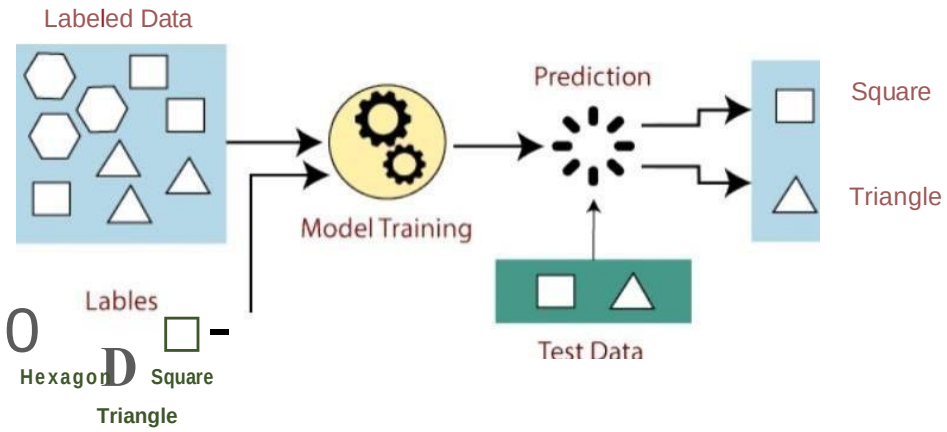
At abroad level machine learning can be classified into three types:

1. **Supervised learning**
2. **Unsupervised learning**
3. **Reinforcement learning**

1. Supervised Machine Learnini:;:

Supervised learning is the types of machine learning in which machines are trained using well "labeled" training data, and on basis of that data, machines predict the output.

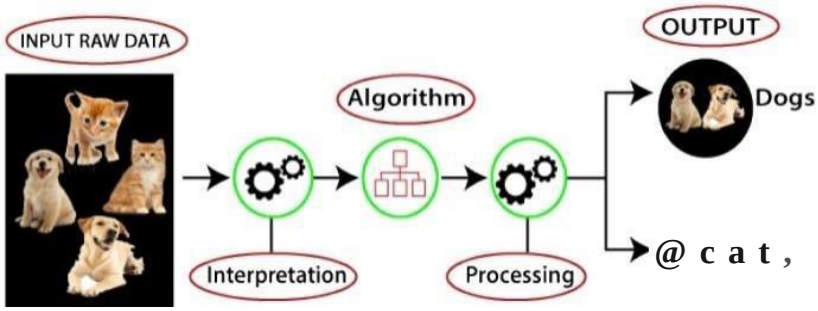
- Thelabelleddatameanssomeinputdataisah-eadytaggedwiththecorrectoutput.
- In supervised learning, models are trained using labeled dataset, where the model learns about each type of data.
- Once the training process is completed, the model is tested on the basis of test data (a subset of the training set),and then it predicts the output.



2. Unsupervised Machine Learning:

Unsupervised learning is a machine learning technique in which models are not supervised using training data set. Instead, models itself find the hidden patterns and insights from the given data. It can be compared to learning which takes place in the human brain while learning new things. It can be defined as:

Unsupervised learning is *a type of machine learning in which models are trained using unlabeled dataset and are allowed to Acton that data without any supervision.*



Unlabeleddata

3)Reinforcement Learning:

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, 1s an example of Reinforcement learning.

What is Reinforcement Leaming?

- Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.
- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.
- RL solves a specific type of problem where decision making is sequential, and the goal is long-term, such as **game-playing, robotics,** etc.

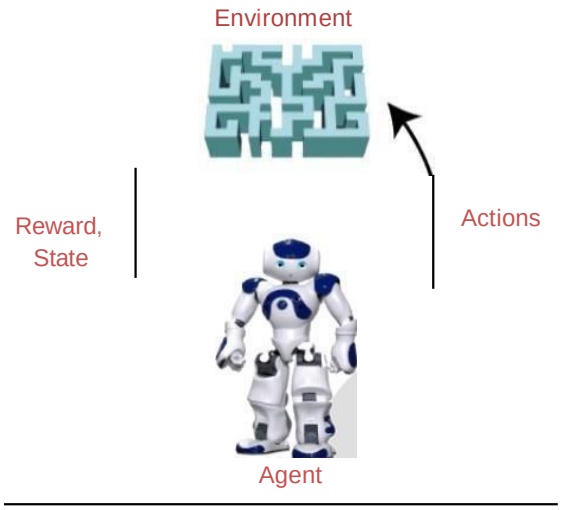
- The agent interacts with the environment and explores it by itselfTI1e primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.
- The agent learns with the process of hit and trial, and based on the experience, it learns to perform the task in a better way. Hence, we can say that

"Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."

- It is a core pa1t of Artificial intelligence, and all AI agent works on the concept of reinforcement learning. Here we do not need to pre-program the agent, as it learns from its own experience without any human inte1vention.
-

Example: Suppose there is an AI agent present within amaze environment, and his goal is to find the diamond. The agent interacts with the environment by performing some actions, and based on those actions, the state of the agent gets changed, and it also receives a reward or penalty as feedback.

- The agent continues doing these three things **(take action, change state/remain in the same state, and get feedback),** and by doing these actions, he learns and explores the environment.
- The agent learns that what actions lead to positive feedback or rewards and what actions lead to negative feedback penalty. As a positive reward, the agent gets a positive point, and as a penalty, it gets a negative point.



Terms used in Reinforcement Learning

- **Agent():**Anentitythatcanperceive/exploretheenvironmentandactuponit.
- **Environment O:** A situation in which an agent is present or surrounded by. In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():**Actions are the moves taken by an agent with in the environment.
- **State O:** State is a situation returned by the environment after each action taken by the agent.
- **Reward O:** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():**Policyisastrategyappliedbytheagentforthenextactionbasedonthecunent state.
- **Value():**It is expected long-term retuned with the discount factor and opposite to the sh01t-te1m reward.
- **Q-value O:**It is mostly similar to the value, but it takes one additional parameter as a current action (a).

Approaches to implement Reinforcement Learning

Therearemainlythreewaystoimplementreinforcement-learninginML,whichare:

1. **Value-based:**
   The value-based approach is about to find the optimal value function, which is the maximum value at as take under any policy. TI1erefore,theagent expects the long-term return at any state(s) under policy *n*.
2. **Policy-based:**
   Policy-based approach is to find the optimal policy for the maximum future rewards without using the value function. In this approach, the agent hies to apply such a policy thattheactionperfo1medineachstephelpstomaximizethefuturereward. The policy-based approach has mainly two types of policy:
   - **Deterministic:** The same action is produced by the policy(n)at any state.
   - **Stochastic:**Inthispolicy,probabilitydete1minestheproducedaction.
3. **Model-based:** In the model-based approach, a virtual model is created for the environment, and the agent explores that environment to learn it. There is no particular solutionoralgorithmforthisapproachbecausethemodelrepresentationisdifferent for each environment.

passive learning, where the agent's policy is fixed and the task is to learn the utilities of states(or state-action pair ·s);this could also involve learning model of the enviI· omnent.

Active learning, where the agent must also learn what to do. The principal issue is exploration: an agent must experience as much as possible of its environment in order to learns how to behave in it.

Passive Reinforcement Learning:

- Inpassivelearning,theagent'spolicy1ıisfixed:instates,italwaysexecutestheaction n(s).Its goal is in ply to learn how good the policy is-that is, to learn the utility function Un(s).

► In passive learning, the agent's policy TC is fixed: in state s, it always executes the action TC(s).Its goal is simply to learn how good the policy is-that is, to learn the utility function UTC(s).

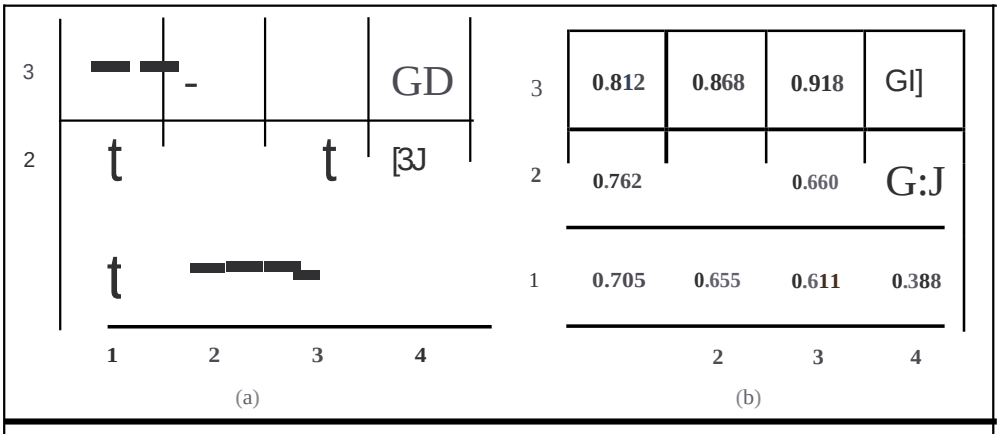


Figure21.1(a)Apolicyrrforthe4x3 world;thispolicyhappenstobeoplimal with rewards of*R(s)*=-0.04inthenontenninalstates and nodiscounting.(b)Theutilitiesof the statesin**the** 4x3world,givenpolicy11.

The agent executes a set of trials in the environment using its policy TC. In each trial, the agent struts in state (1,1) and experiences a sequence of state transitions until it reaches one of the tenninal states, (4,2)or(4,3). Its percepts supply both the current state and there would received in that state. Typical trials might look like this:

(1,1)..04--+**(1,2).,04**--+**(l,3)..04**--+**(1,2)..04**----+**(1,3)..04**----+**(2,3)..04**----+**(3,3)..04**+**(4,3)+t**
(1,1)._04--+**(l,2).,04**----+**(l,3)._04**--+**(2,3)..04**--+**(3,3)..04**---(-**3**,**2)..04**--+**(3,3)._04**--+**(4,3)+t**
(1,1)..04--+**(2,l).,04**----+**(3,1)..04**--+**(3,2)..04**----+**(4,2).1**.

The object is to use the information about rewards to learn the expected utility UTC(s) associated with each non terminal state s

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

1. Direct utility estimation
11. Adaptive dynamic programming
m Temporal-difference learning

i. Directutility estimation:

The idea is that the utility of a state ADAPTIVECONTROL THEORY REWARD-TO-GO is the expected total reward from that state onward (called the expected reward-to-go), and each trial provides a sample of this quantity for each state visited.

For example: the first trial in the set of three given earlierprovidesasampletotalrewardof0.72 forstate(l,l),twosamplesof0.76and0.84for(l,2),twosamplesof0.80and0.88for(1,3), and soon. Thus ,at the end of each sequence, the algorithm calculates the observed reward-to-go for each state and updates the estimated utility for that state accordingly, just by keeping a running average for each state in a table.

- Direct utility estimation succeed sin reducing there inforcement learning problem to an inductive learning problem, about which much is known.
- Unfortunately, it misses a veryimpo1tant source of information, namely, the fact that the utilities of states are not independent!
- The utility of each state equals its own reward plus the expected utility of its successor states.Thatis,theutilityvaluesobeytheBellmanequationsforafixedpolicy

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi(s)) U^{\pi}(s') .$$

ii)Adaptive dynamic programming:

- An adaptive dynamic programming(or ADP) agent takes advantage of the constraints amongtheutilitiesofstatesbylearningthetransitionmodelthatconnectsthemand

solvingtheconespondingMarkovdecisionprocessusmgadynamicprograimmng method.

- Forapassiveleai·ningagent,thismeanspluggingthelearnedtransitionmodelP(sls,n(s)) and the observed rewards R(s) into the Belhnan equations to calculate the utilities of the states.
- Alternatively, we can adopt the approach of modified policy iteration using a simplified valueiterationprocesstoupdatetheutilityestimatesaftereachchangetothelearned model.
- Because the model usually changes only slightly with each observation, the value iteration process can use the previous utility estimates as initial values and should converge quite quickly.
- We keep track of how often each action outcome occurs and estimate the transition probability P(sls, a) from the frequency with which sis reached when executing gains.
- Forexainple,inthethreetiialsgivenonpage832,Rightisexecutedthreetimesin(1,3) and two out of three times the resulting state is(2,3), so P((2,3)1(1, 3),Right) is estimated to be 2/3.

iii. Temporal-difference learning:

- Solving the underlying MDP as in the preceding section is not the only way to bring the Belhnan equations to bear on the learning problem.
- Another way is to use the observed tTai1Sitionstoadjust the utilities of the observed states so that they agree with the constraint equations.

Consider, for example, the trai1sition from (1,3) to (2,3) in the second trial on page 832. Suppose that, as a result of the first trial, the utility estimates are Un(l,3) = 0.84 and Un(2, 3) = 0.92. Now, if this trai1sition occurred all the time, we would expect the utilities to obey the equation

u,..(1,3)=-o.04+u,..(2,3),

so uw(l,3)*would*be0.88. Thus, its current estimate of 0.84 might be alittlelow and should be increased.More generaHy, when a transition occurs from state *s* tostate*s′*, weapply the following update toU,..(e):

$$U(s) \leftarrow U(s) + \alpha(R(s) + U(s') - U(s))$$

Here,a, is the**learningrate**parameter. Because this update rule usesthedifference inutilities between successive states, it is often calledthe **temporal-difference,** or TD, equation.

2. Active Reinforcement Learning:

Apassivelearningagenthasafixedpolicythatdeterminesitsbehavior.Anactive agent must decide what actions to take.

Let us begin with the adaptive dynamic programming agent and consider how it must be modified to handle this new freedom.

First, the agent will need to learn a complete model with outcome probabilities for all actions, rather than just the model for the fixed policy.

TI1esimplelearningmechanismusedbywilldojustfine forthis.

Next, we need to take in to account the fact that the agent has a choice of actions. The utilities it needs to learn are those defined by the optimal policy;

What the greedy agent has overlooked is that actions do more than provide rewards according to the current learned model; they also contribute to learning the true model by affecting the percepts that are received.

By improving the model, the agent will receive greater rewards in the future. An agent therefore must make a trade off between exploitation to maximize its reward- as reflected in its cunent utility estimates-and exploration to maximize its long- term well-being.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' | s,a) U(s').$$

GeneralizationinReinforcementLearning:

- So far, we have assumed that the utility functions and Q-functions learned by the agents are represented in tabular form with one output value for each input tuple.
- Such an approach works reasonably well for small state spaces, but the time to convergence and (for ADP) the time per iteration increase rapidly as the space gets larger.
- With carefully controlled, approximate ADP methods, it might be possible to handle 10,000statesor more. This suffices for two-dimensional maze-like environments, but more realistic worlds are out of the question.

Onewaytohandlesuchproblemsistousefunctionapproximation,whichsimplyFUNCTION APPROXIMATION means using any sort of representation for the Q-function other than a lookup table. There presentation is viewed as approximate because it might not be the case that the hue utility function or Q-function can be represented in the chosen form.

we described an evaluation function for chess that is represented as a weighted linear function of a set of features (or basis functions) fl,...,fn:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \cdots + \theta_n f_n(s) .$$

A reinforcement learning algorithm can learn values for the parameters0 = 01,...,0n such that the evaluation function U'0approximates the true utility function. Instead of, say, 1040values in a table, this function approximate is characterized by, say, n =20 parameters-an enormous compression.

PolicySearch:

- The final approach we will consider for reinforcement learning problems is called policy search. In some ways, policy search is the simplest of all the methods. the idea is to keep twiddling the policy as long as its performance improves, then stop.

Let us begin with the policies themselves.

- Remember that a policy *n* is a function that maps states to actions. We are interested primarily in parameterized representations of *n* that have far fewer parameters than there are states in the state space (just as in the preceding section).
- For example, we could represent TC bya collection of parameterized Q-functions, one for each action, and take the action with the highest predicted value

$$1r(s)=\max_{a} Qo(s,a).$$

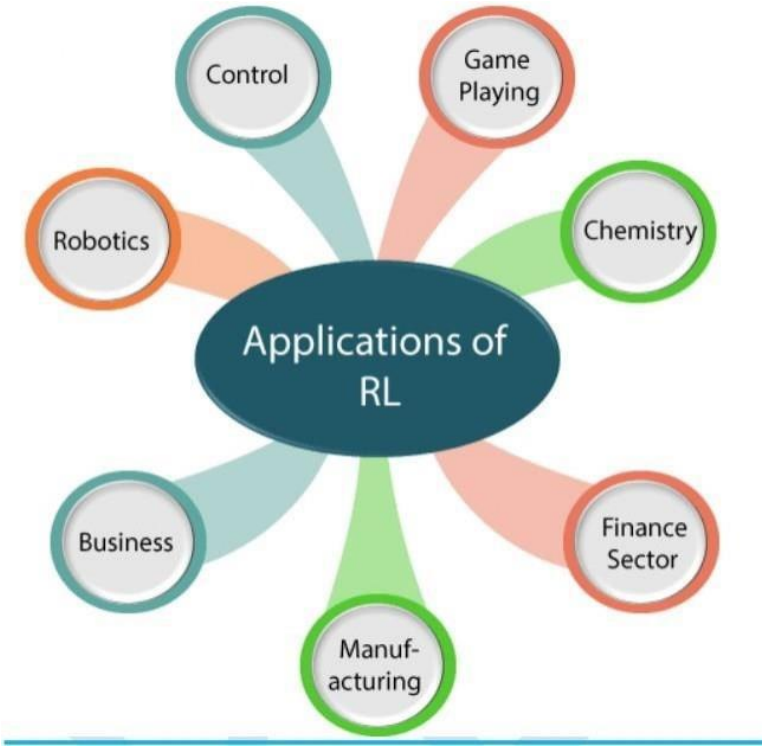
EachQ-functioncouldbealinearfunctionoftheparameters0,

- Policysearchwillthenadjusttheparameters0toimprovethepolicy.
- Notice that if the policy is represented by Q functions, then policy search results in a process that learns Q-functions.
- This process is not the same as Q-learning!
- In Q-learning with function approximation, the algorithm finds a value of 0 such that Q 0 is "close" to Q*,the optimal Q-function.
- Policysearch,ontheotherhand,fmdsavalueof0thatresultsingoodperformance;the values found by the two methods may differ very substantially.

policy search methods often use a stochastic policyrepresentationn0(s, a),which specifies the probability of selecting action a in states. One popular representation is the soft max function:

$$1ro(s,a)=eQe(s.a)/L_{a'} eQe(s.a')'$$

ApplicationsofRL:



1. **Robotics:**

   RL issued in **Robot navigation, Robo -soccer, walking, juggling**, etc.

2. **Control:**

   RL can be used for **adaptive control** such as Factory processes, admission control in telecommunication, and Helicopter pilot is an example of reinforcement learning.

3. **Game Playing:**

   RL can be used in **Game playing** such as tic-tac-toe, chess, etc.

4. **Chemistry:**

   RL can be used for optimizing the chemical reactions.

5. **Business:**

   RL is now used for business strategy planning.

6. **Manufacturing:**

In various automobile manufacturing companies, the robots use deep reinforcement learning to pick goods and put them in some containers.

7. **Finance Sector:**

   The RL is currently used meth finance sector for evaluating trading strategies.

NATURALLANGUAGEPROCESSING:

Introduction:

- There are over at Trillion pages of information on the Web, almost all of it in natural language. An agent that wants to do knowledge acquisition needs to understand (at least paitially) the ambiguous, messy languages that humans use.
- We examine the problem from the point of view of specific information-seeking tasks :text classification, information retrieval, and information extraction.
- One co1mnon factor in addressing these tasks is the use of language models: models that predict the probability distribution of language expressions.

LANGUAGEMODELS:

We will discuss about2typesoflanguages:

- Formal Languages
- Natural Languages

Formal languages, such as the programming languages Java or Python, have precisely defined language models.

- A language can be defined as a set of strings;"print(2+2)" is a legal program in the language Python, whereas "2)+(2 print" is not.
- Since there are an infinite number of legal programs, they cannot be enumerated; instead they are specified by a set of miles called a grammar.
- Formallanguagesalsohaverulesthatdefinethemeaningorsemanticsofaprogram;for example,therulessaythatthe"meaning"of"2+2"is4,and the meaning of "l/0"isthatan error is signaled.

Natural languages, such as English or Spanish, cannot be characterized as a definitive set of sentences.

- Everyone agrees that "Not to be invited is sad" is a sentence of English ,but people disagree on the grammaticality of "To be not invited is sad."

Problems:

- Natural languages are also ambiguous.
- ONaturallanguagesaredifficulttodealwithbecausetheyareverylarge,and constantly changing.

N-gram character models:

Awrittentextiscomposedofcharacters-letters,digits,punctuation,andspacesinEnglish.

Thus, one of the simplest language models is a probability distribution over sequences of characters. we write P(cl:N) for the probability ofasequenceofNcharacters,c1through c N. In one Web collection, P("the")=0.027and P("z g q")=0.000000002.

Definition: A sequence of written symbols of lengthen is called ann-gram.

A model of the N-GRAMMODEL probability distribution of n-letter sequences is thus called an n-gram model.

Ann-grammodelis**definedasa**Markovchainofordern-1.

that in a Markov chain the probability of character ci depends only on the immediately preceding characters, one to n any other characters. So in a trigram model(Markov chainoforder2)we have

$$P(ci]_{Cl:i-1)} = P(Ci_{JCi-2:i-t)}.$$

Wecandefinetheprobabilityofasequenceofcharacters$_{P(ci:N)}$underthetrigrammodel **by first** factoring **with the** chain rule **and then using the Markov** assumption:

$$P(c1:N) = \prod_{i=1}^{N} P(q|C1:i-1) = \prod_{i=1}^{N} P(q|Ci-2:i-l)$$

What can we do with n-gram character models?

.....One task for which they are well suite dis **language identification:** given a text, determine what natural language it is written in.

- Thisisarelativelyeasytask;evenwithshorttextssuchas"Hello,world"or"Wie geht esdir, "it is easy to identify the first as English and the second as German.
- Computersystemsidentifylanguageswithgreaterthan99%accmacy;occasionally, closely related languages, such as Swedish and No1wegian, are confused.
- Oneapproachtolanguageidentificationistofirstbuildatrigramcharactermodelofeach candidate language, **P(C;**lCd:;-1,L), where the variable L ranges over languages.

Smoothing n-gram models:

The major complication of n-gram models is that the training corpus provides only an estimate of the true probability distribution.

For common character sequences such as "th" any English corpus will give a good estimate: about 1.5%ofalltrigrams.

On the other hand, "ht" is very uncommon-no dictionary words it with .It is likely thatthesequencewouldhaveacountofzeroinatrainingcorpusofstandai·dEnglish.

Does that mean we should assign P("th")=O? If we did, then the text "The program issues an http request" would have an English probability of zero, which seems wrong.

Just because we have never seen "http" before does not mean that our model should claim that it is impossible. Thus, we will adjust our language model so that sequences that have a count of zero in the training corpus will be assigned a small nonzero probability.

The process od adjusting the probability of low-frequency counts is called **smoothing.**

Laplace smoothing--The simplest type of smoothing was suggested by Pie1Te-Sirnon Laplace in the 18thcentury: he said that, in the lack of fm the rinformation, if a random Boolean variable X has been false in all n observations so far then the estimate for P(X= hue) should be 1/(n+2).That is, he assumes that with two more trials, one might be hue and one false.

A better approach is a **back off model,** in which waste it by estimating n-gram counts, but for any particular sequence that has a low (or zero) count, we back off to (n -1)-grams.Linear interpolation smoothing is a back off model that combines trigram, big rain, and models by linear interpolation. It defines the probability estimate as

$$P(CiJCi\text{-}2{:}i\text{-}1) = A3P(CiJCi\text{-}2{:}i\text{-}t) + A2P(CilCi\text{-}1) + A1P(Ci),$$

$$\text{where } A3 + A2 + A1 = 1.$$

Model evaluation:

Withsomanypossiblen-grammodels-unigram,bigram,trigram,interpolatedsmoothingwith different values of A., **etc.-how do we know what model to choose?**

…..... We can evaluate a model with **cross-validation.**

Split the corpus in to a training corpus and a validation corpus. Determine the pai-ameters of the model from the training data. Then evaluate the model on the validation corpus.

N-gram word models:

- AllthesamemechanismVOCABULARY appliesequallytowordandcharactermodels. The main difference is that thevocabulary- the set ofsymbols that make up the corpus and the model-is larger.
- Thereareonlyabout100charactersinmostlanguages,andsometimeswebuildcharacter models that are even more restrictive, for example by treating "A"and "a"as thesame symbolorbytreatingallpunctuationasthesamesymbol.
- But with word models we have at least tens of thousands of symbols, and sometimes millions.

Wordn-grammodelsneedtodealwithoutofvocabularywords.Withcharactermodels, we didn't have toworry about someone inventinga new letter of the alphabet.But with word models there is always the chance of a new word that was not seen in the training corpus, so we need to model that explicitly in our languagemodel.

Solution:This canbedonebyaddingjust onenewwordtothevocabulaiy:**<UNK>.,** standing for the unknown word.We can estimate n-gram counts for**<UNK>.by** this trick: go through the training corpus, and the first time a1.1y individual word appeai·s it is previously unknown, so replace it with the symbol**<UNK>.**

Forexample,anystringofdigitsmightbereplacedwith**<NUM>•**,oranyemailaddresswith **<EMAIL>**

TEXTCLASSIFICATION:

Wenowconsiderindepththetaskoftext classification, also known ascategorization: givenatextofsomekind,decidewhichofapredefinedsetofclasses itbelongsto.

Languageidentificationand geme classificationai·e examples of text classification, as is sentimentanalysis (classifyinga movie or product review as positive or negative) and spam detection (classifying an email message as spam or not-spam).

Since"not-spam"isawkward,researchershavecoinedthetermhamfornot-spamWe cantreatspaindetection asaprobleminsupervised learning.

A training set is readily available:thepositive(spam) examples are inmyspam folder, the negative (ham) examples are in my inbox. Here is anexcerpt

Spam:WholesaleFashionWatches-57%today.Designerwatchesforcheap ...
Spam:YoucanbuyYiagraFr$l.85AllMedicationsatunbeatableprices!...
Spam:WECANTREATANYTHIG YOUSUFFERFROMJUSTTRUSTUS...
Spam:Sta.rteam*ingthesalaryyo,ud-eservebyo'btainingtheprope,rcrede'nliaJs!

Ham:The practical significance of hype11.ree width in identifying more... Ham: Abstract: We will motivare the problem ofociaJidenrity clustering: ... Ham: Goodto see youmyfriend.HeyPeter, Itwasgoodtohearfromyou....
Ham:PDSimpliesconvexityoftheresultingopcimizarionproblem(KernelRidge...

From thisexce1pt we canstaitto get anidea ofwhat might be good features toinclude in the supervised leai·ning model.

Word n-grams suchas "for cheap"and "You can buy" seem to be indicators ofspam. Apparently the spammers thought that the word bigram "you deserve"would be too indicative of spam, and thus wrote"yo,u d-eserve"instead.

Acharactermodelshoulddetectthis.Wecouldeithercreateafullcharacter n-gram model ofspam and ham,or we could handcraft features such as"number of punctuation marksembedded inwords."

Classificationbydatacompression:

- Anotherwaytothink about classificationisasaproblemindatacompression.
- A lossless compression algorithmtakes a sequence of symbols, detects repeated patternsin it, andwrites a description ofthe sequence that is more compactthan the original.
- Forexample, the text"0.142857142857142857"might becompressed to "0.[142857]*3."Compression alg01ithms work by building dictionaries of subsequences of the text, and thenrefening to entries in the dictionary.
- Theexampleherehadonlyonedictionaryentry,"142857."

INFORMATIONRETRIEVAL:

Definition: Informationretrieval is the task of finding documentsthat are relevantto a user's need for information.

Ex:TI1e best-known examples ofinformationretrievalsystemsaresearch engines ontheWorld Wide Web. A Webuser can type a querysuchas [AIbook]intoa search engine and seea list of relevant pages.

AnIR**informationretrievalsystemcanbecharacterizedby**

1. A corpus ofdocuments. Each system must decidewhatitwants totreat asa document: a paragraph, a page,or amultipage text.

2. Queriesposed in a query language.A queryspecifies whattheuser wantsto know. The querylanguagecanbejustalistofwords,suchas[Albook];oritcanspecifyaphraseofwords thatmustbeadjacent,asin["AIbook"];itcancontainBooleanoperatorsasin[AlANDbook]; itcanincludenon-Booleanoperatorssuchas[AlNEARbook]or[AIbooksite:www.aaai.org].

3. A result set. This is the subset of documents that the IR system judges to be relevant to RELEVANT the query. By relevant, we mean likely to be of use to the person who posed the query, for the pruticular information need expressed in the query.

4. Apresentationoftheresultset. Thiscanbeassimpleas arankedlistofdocument titlesor as complex as a rotatingcolormapof the result setprojected onto a threedimensional space, rendered asa two-dimensionaldisplay.

BooleanKeywordModel: Theeru·liestIR systemsworkedonaBooleankeywordmodel. Eachword inthe document collection istreatedasa Boolean feature that is true ofadocument if the word occurs in the documentand false if it does not.

Advantages: Thismodelhastheadvantageofbeingsimpletoexplainandimplement.

Disadvantages:

1. The degreeofrelevanceof a documentis asingle bit, sothere isno guidanceas tohow to order the relevant documents for presentation.
2. Booleanexpressionsareunfamiliartouserswhoarenotprogrammersorlogicians.
3. it canbehardtofonnulatean approp1iatequery,evenforaskilleduser.

IRscoringfunctions:

Definition:A scoring function takes a documentanda query andreturnsa numeric score;the most relevant documentshavethehighestscores.

Inthe**BM25function,**thescoreisalinearweightedcombinationofscoresforeachofthewords that make up the query.

Threefactors affecttheweightofaquerytenn:

i. **TF:** The frequencywithwhicha querytermappears in a document(also known asTFfor tenn:frequency).
   Ex: For thequery[farming in Kansas], documents thatmention "farming"frequently will have higher scores.
ii. **IDF:** The inverse document frequency of the term, orIDF. The word "in" appears in almost every document, so it has a high document frequency, and thus a low inverse documentfrequency, andthus it isnot asimportantto the queryas"farming"or"Kansas.
iii. **Length:**Thelength of the document.Amillion-worddocumentwill probablymention all the query words, but may not actually be about the query. A sho1t document that mentions all the words is a much better candidate.

TheBM25functiontakesallthreeofthese intoaccount.Weassumewehavecreated anindexoftheNdocuments inthecorpussothatwecan look up TF(qi,di), thecount of thenumberoftimeswordqi appearsindocumentdj.Wealsoassumeatableofdocument frequencycounts,DP(qi),thatgives the number ofdocumentsthat containthe wordqi. Then,givenadocumentdjandaqueryconsisting ofthewordsq1:N,wehave

$$BM25(d_j, q_{1:N}) = \sum_{i=1}^{N} IDF(q_i) \cdot \frac{TF(q_i, d_j) \cdot (k+1)}{TF(q_i, d_j) + k \cdot (1 - b + b \cdot \frac{|d_j|}{L})},$$

where|dj|isthelengthofdocumentdjinwords,andListheaveragedocumentlength inthecorpus:L=Lild-i|/N.Wehavetwoparameters,kand b, thatcan betunedby cros-validation;typicalvalues arek=2.0and b=0.75.IDF(qi)is theinversedocument

frequencyofwordqi,givenby

$$IDF(q_i) = \log \frac{DF(q_i) + 0.5}{DF(q_i) + 0.5}$$

IRsystemevaluation:

HowdoweknowwhetheranIRsystemisperformingwell?

Traditionally, there have been two measures used in the scoring: recallandprecision. We explain themwith thehelpofanexample. Imagine that an IRsystem hasreturned a result set for a single query, for which we knowwhichdocuments are and are not relevant,out of a co1pus of 100 documents. The document counts in each catego1y are given in the following table:

	Inresultset	Notinresultset
Relevant	30	20
Notrelevant	10	40

Precisionmeasuresthepropo1tionofdocumentsintheresultsetthatare actuallyrelevant.

Ex:

Inourexample,theprecisionis30/(30+10)=.75.Thefalsepositiverateisl-.75=.25.

Recall measures the propo1tion ofall the relevant documents in the collection that are in the resultset.

Ex: Inourexample,recallis30/(30+20)=.60.TI1efalsenegativerateis1 -.60=.40.

In a very large documentcollection,such as the WorldWide Web, recall is difficultto compute,becausethereis noeasy way toexamineevery pageontheWeb forrelevance.All we can do is either estimate recall by sampling or ignore recall completely and just judge precision.In the case of a Web searchengine, there may be thousandsof documentsin the result set, so it makes more sense to measure precision for several different sizes, such as "P@lO"(precisionin the top 10 results) or "P@50,"rather than to estimate precision in the entire result set.

IRrefinements:

One co1mnon refinement isa better modelof theeffect ofdocumentlength on relevance **pivoted document length normalization scheme;** the idea is that the pivot isthe document length atwhich theold-style normalizationis conect; documents sho1ter than that get aboost and longer ones get a penalty.

TI1e BM25 scoring function uses a word model that tr·eats all words as completely independent, but weknow that some words are correlated: "couch"is closely related to both "couches"and "sofa."Many IR systems attempt to account for these correlations.

STEMMING:

Forexample,if thequeryis[couch], itwould be ashame toexclude fromtheresult set thosedocuments thatmention"COUCH"or"couches"butnot"couch."

MostIRsystemsCASEFOLDINGdocasefoldingof"COUCH"to"couch,"andsome use a stemmingalgorithmto reduce"couches"to the stem form "couch,"both in the queryand the documents.

TI1istypically yields a smallincreaseinrecall (on the order of2% forEnglish). However, it can bairn precision.

Forexample,stemming"stocking"to"stock"willtend todecreaseprecision forqueries abouteitherfootcoveringsorfinancialinsh"uments,althoughitcouldimproverecallfor

queries about warehousing. Stemming algorithms based on rules (e.g., remove "-ing") cannotavoidthis problem,butalgorithmsbasedon dictionaries(don'tremove"-ing"if the word isah-eady listed in the dictionary)can.

WhilestemminghasasmalleffectinEnglish,it ismoreimportantinotherlanguages.

SYNONYM:

recognize synonyms, such as"sofa"for"couch."Aswithste1mning,this has thepotential for small gainsin recall,butcan hurtprecision.Auser whogives the query[TimCouch]wants to see results about the football player, not sofas.

Tiiat is, anytime there are two words that mean the same thing, speakers of the language conspire to evolve the meanings to remove the confusion. Related words that are not synonymsalsoplayanimpo1tantrole in ranking-termslike "leather","wooden,"or "modern" can serve to confirm that the document really is about "couch."Synonyms and related words canbe found in dictionaries or by looking for coITelations in documents or in queries-if we find thatmany users who askthe query [new sofa] follow it up with the query [new couch],we can in the future alter [new sofa] tobe [new sofa OR new couch].

METADATA:

As afinalrefinement,IR can beimprovedby consideringmetadata-dataoutside ofthetext of the document. Examples include human-supplied keywords and publication data. LINKS On theWeb, hype1text links between documents area crncial source of information.

ThePageRankalgorithm:

PageRank was inventedto solve the problem ofthetyram1y ofTF scores: if the query is [IBM], how dowe makesurethat IBM's home page, ibm.com, is the first result, even if another page mentions the term "IBM"more frequently?

TI1eideaisthat**ibm.com**hasmanyin-links(linkstothepage),so itshould beranked higher:eachin-link isa voteforthequalityofthelinked-topage.

But ifweonlycounted in-links, then itwouldbepossible for a Webspammer to create a network ofpages and have them all point to a page ofhis choosing,increasingthescore ofthat page.

Therefore, the PageRank algorithm is designed to weight links from high-quality sites moreheavily.

What is a**highqualitysite?** One thatis linked to by other high-qualitysites. The definitionisrecursive, butwe willseethat the recursionbottoms outproperly. The PageRank for a page p isdefined as:

$$PR(p) = \frac{1-d}{N} + d \sum_{ini} \frac{PR(ini)}{C(ini)}$$

whereP R(p) isthe PageRank ofpage p, N isthetotalnumber ofpages in the c01pus, iniare the pagesthatlinkintop,andC(ini)isthecountof thetotalnumberofout-linksonpageini.

TheHITSalgorithm:

TI1e Hyperlink-Induced Topic Search algorithm, also known as "Hubs and Authorities"or HITS, is another influential link-analysis algorithm.

- HITS differs from PageRankin several ways.First, it isaquery-dependent measure: it rates pages withrespectto a que1y.
- Thatmeans that itmust becomputed anewforeachquery-acomputationalburden thatmost searchengines haveelectednot totakeon.
- Givenaque1y,HITSfirstfindsaset ofpagesthatarerelevanttothequery.
- It does that by intersecting hit lists ofquery words, and then adding pages inthe link neighborhoodofthesepages-pagesthatlink toorarelinkedfromoneof thepages in the 01iginal relevant set

Questionanswering:

Information retrieval is the task of finding documents that are relevant to a query, where the query may be a question, or just a topic area or concept. Question answering is a somewhat QUESTION ANSWERING different task, in which the query really is a question, and the answerisnotarankedlistofdocumentsbutrather asho1tresponse-asentence,or even just a phrase.

TI1e ASKMSR system (Banko et al., 2002) is a typical Web-based question-answering system. It is based on the intuition that most questions will be answered many times on the Web, so question answering should be thought of as a problem in precision, not recall. We don'thave todealwith all thedifferentwaysthatan answermightbe phrased-weonly have to find one of themForexample,considerthe query [Who killed AbrahamLincoh1?] Suppose a system had to answer that question with access only toa single encyclopedia, whose entry on Lincom said

"**JohnWilkesBoothalteredhisto11'withabullet.Hewillforeverbeknownasthemanwho ended AbrahamLincoln's life.**"

INFORMATIONEXTRACTION:

Definition:

Informationextraction istheprocessofacquiringknowledge by skimminga texta11d looking for occunences of a paiticular class of object and for relationshipsamong objects.

- A typical task is to extract insta11ces of addresses from Web pages, with database fieldsforstreet,city, state,and zipcode;orinsta11cesofsto1ms fromweatherrepo1ts, with fields for temperature, wind speed, and precipitation.
- In a limited domain, this canbe done with high accuracy. As the domain gets more general, more complex linguistic models and more complex learning techniques are necessary.

Here wehave approaches to information extraction, inorder of increasing complexity on several dimensions: deterministic tostochastic,domain-specifictogeneral, hand-crafted tolea111ed, and small-scale to large-scale.

i. **Finite-stateautomataforinformationextraction:**

► TI1e simplest type of information extraction system is an attribute-based extraction system thatassumes that the entire text refersto a single object andthe taskis to extract attributes of that object.

Ex:For example,extractingfromthe text "IBMThinkBook970.Ourprice: $399.00"the set of attributes {Maimfacturer=IBM, Model=ThinkBook970, P1ice=$399.00}.

We can address this problem by defining atemplate (also known as a pattern)for each attribute we would like toextract. Thetemplateis defined byafinitestateautomaton, the simplest example of which is the regular expression,or regex. Regular expressions areusedinUnix co1mnands suchasgrep, inprogramminglanguages suchasPerl, and in word processors such as Microsoft Word.

`(0-9]`	matchesanydigitfromOto9
`[0-9]+`	matchesoneormoredigits
`[.][0-9][0-9]`	matchesaperiodfollowedbytwodigits
`([.][0-9][0-9])?`	matchesaperiodfollowedbytwodigits,ornothing
`($][0-9]+([.][0-9][0-9])?`	matches$249.99or$1.23or$I000000or...

Onestepup from attribute-based extractionsystemsare relational extractionsystems, RELATIONAL EXTRACTIONwhichdealwith multipleobjects andthe relations among them. Thus,whenthesesystemsseethetext"$249.99,"theyneedtodeterminenotjustthatitisaprice,but alsowhichobject has that price. A typical relational-basedextraction system isFASTUS, which handlesnewsstoriesaboutcorporatemergersandacquisitions.

FASTUSconsistsoffivestages:

1. Tokenization
2. Complex-wordhandling
3. Basic-grouphandling
4. Complex-phrasehandling
5. Strncturemerging

i. **Tokenization:**

FASTUS'sfirst stage is tokenization,whichsegmentsthestreamof characters into tokens (words,numbers,andpunctuation).For English,tokenization can be fairlysimple; just separating charactersat white spaceor punctuation does a fairly good job.Some tokenizes also deal with markup languages suchas HTML, SGML, andXML.

..

11. **complexwords,**includingcollocationssuchas"setup"and"jointventure,"aswellasproper namessuchas"BridgestoneSportsCo."Thesearerecognizedbyacombinationoflexicalentries

and finite-state grammar mles. For example, a company name might be recognized by the mle Capitalized Word+ ("Company"|"Co"|"Inc"|"Ltd")

111. Thethirdstagehandlesbasicgroups,meaningnoungroupsandverbgroups.Theideaistochunk these into units that will be managed by the later stages. We will see how to write a complex description of noun and verb phrases in Chapter 23,but here we have simple rnles that only approximate the complexity of English, but have the advantage of being represent able by finite stateautomata.Theexamplesentencewouldemergefromthisstageasthefollowingsequenceof tagged groups:

```
1 NG:BridgestoneSportsCo.10NG:alocalconcern
2 VG:said                  11CJ:and
3 NG:Friday                12NG:aJapanesetradinghouse
4 NG:it                    13VG:toproduce
5 VG:hadsetup              14NG:golfclubs
6 NG:ajointventure         15VG:tobeshipped
7 PR:in                    16PR:to
8 NG:Taiwan                17NG:Japan
9 PR:with
```

HereNGmeansnoungroup,VGisverbgroup,PRispreposition,andCJisconjunction.

IV. The fourth stage combines the basic groups into complex phrases. Again, the aim is to have rnles that are finite-state and thus can be processed quickly, and that result in unambiguous(or nearly unambiguous)output phrases. One type of combination rnle deals with domain-specific events. For example, the rule Company+ Setup Joint Venture ("with" Company+)?

v. The final stage merges structures that were built up in the previous step. If then ext sentence says "The joint venture will start production in January, "then this step will notice that there are two references to a joint venture, and that they should be merged in to one.

3. Probabilisticmodelsforinformationextraction:

When information extraction must be attempted from noisy or varied input, simple finite-state approachesfarepoorly.Itistoohardtogetallthernlesandtheirprioritiesright;itisbettertouse a probabilistic model rather than a rnle-based model. The simplest probabilistic model for sequences with hidden state is the hidden Markov model, or HMM.

To apply HMMs to information extraction, we can either build one big HMM for all the attributes or build a separate HMM for each attribute.

HMMs have two big advantages over FSAs for extraction.

First, HMMs are probabilistic ,and thus tolerant to noise. In a regular expression, if a single expectedcharacterismissing,theregexfailstomatch;withHMMsthereisgracefuldegradation withmissingcharacters/words,andwegetaprobabilityindicatingthedegreeofmatch,notjusta Boolean match/fail.

Second ,HMMs can be trained from data; they don't require laborious engineering of templates, and thus they can more easily be kept up to date as text changes overtime.

Conditional random fields for information extraction:

One issue with HMMs for the information extraction task is that they model a lot of probabilities that we don't really need.

An HMM is a generative model; it models the full join t probability of observations and hidden states, and thus can be used to generate samples. That is, we can use the **HMM** model not only to parse a text and recover the speaker and date, but also to generate a random instance of a text containing a speaker and a date.

Since we're not interested in that task, it is natural to ask whether we might be better off with a model that doesn't bother modeling that possibility.

All we need in order to understand a text is a discriminative model, one that models the conditional probability of the hidden attributes given the observations (the text).

A frameworkforthistypeofmodelistheconditionalrandomfield,orCRF,whichmodelsa conditional probability distribution of a set of target variables given a set of observed variables.

Like Bayesian networks, CRFs can represent many different structures of dependencies among the variables. One common structure is the linear-chain conditional random field for representing Markov dependencies among variables in a temporal sequence.

Thus, HMMs are the temporal version of naive Bayes models, and linear-chain CRFs are the temporal version of logistic regression, where the predicted target is an entire state sequence rather than a singlebina1y variable.

****

ARTIFICIAL INTELLIGENCE

UNIT-IV

SYLLABUS:

Natural Language for Communication: Phrase structure grammars, Syntactic Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition.

Perception: Image Formation, Early Image Processing Operations, Object Recognition by appearance, Reconstructing the 3D World, Object Recognition from Structural information, Using Vision.

ARTIFICIAL INTELLIGENCE

Unit-IV: **Natural Language for Communication**: Phrase structure grammars, Syntactic Analysis, Augmented Grammars and semantic Interpretation, Machine Translation, Speech Recognition
Perception: Image Formation, Early Image Processing Operations, Object Recognition by appearance, Reconstructing the 3D World, Object Recognition from Structural information, Using Vision.

Computers don't speak languages the way humans do. They communicate in machine code or machine language, while we speak English, Dutch, French or some other human language. Most of us don't understand the millions of zeros and ones computers communicate in. And in turn, computers don't understand human language unless they are programmed to do so. That's where natural language processing (NLP) comes in.

What is natural language processing?
Natural language processing is a form of artificial intelligence (AI) that gives computers the ability to read, understand and interpret human language. It helps computers measure sentiment and determine which parts of human language are important. For computers, this is an extremely difficult thing to do because of the large amount of unstructured data, the lack of formal rules and the absence of real-world context or intent.

Language is a method of communication with the help of which we can speak, read and write. Natural Language Processing (NLP) is a subfield of Computer Science that deals with Artificial Intelligence (AI), which enables computers to understand and process human language.

Study of Human Languages

Language is a crucial component for human lives and also the most fundamental aspect of our behavior. We can experience it in mainly two forms - written and spoken. In the written form, it is a way to pass our knowledge from one generation to the next. In the spoken form, it is the primary medium for human beings to coordinate with each other in their day-to-day behavior. Language is studied in various academic disciplines. Each discipline comes with its own set of problems and a set of solution to address those.

Consider the following table to understand this −

Discipline	Problems	Tools
Linguists	How phrases and sentences can be formed with words?	Intuitions about well-formedness and meaning. Mathematical model of structure. For example,

	What curbs the possible meaning for a sentence?	model theoretic semantics, formal language theory.
Psycholinguists	How human beings can identify the structure of sentences? How the meaning of words can be identified? When does understanding take place?	Experimental techniques mainly for measuring the performance of human beings. Statistical analysis of observations.
Philosophers	How do words and sentences acquire the meaning? How the objects are identified by the words? What is meaning?	Natural language argumentation by using intuition. Mathematical models like logic and model theory.
Computational Linguists	How can we identify the structure of a sentence How knowledge and reasoning can be modeled? How we can use language to accomplish specific tasks?	Algorithms Data structures Formal models of representation and reasoning. AI techniques like search & representation methods.

Ambiguity and Uncertainty in Language

Ambiguity, generally used in natural language processing, can be referred as the ability of being understood in more than one way. In simple terms, we can say that ambiguity is the capability of being understood in more than one way. Natural language is very ambiguous.

Components of Language

The language of study is divided into the interrelated components, which are conventional as well as arbitrary divisions of linguistic investigation. The explanation of these components is as follows −

Phonology

The very first component of language is phonology. It is the study of the speech sounds of a particular language. The origin of the word can be traced to Greek language, where 'phone' means sound or voice. Phonetics, a subdivision of phonology is the study of the speech sounds of human language from the perspective of their production, perception or their physical properties. IPA (International Phonetic Alphabet) is a tool that represents human sounds in a regular way while studying phonology. In IPA, every written symbol represents one and only one speech sound and vice-versa.

Phonemes

It may be defined as one of the units of sound that differentiate one word from other in a language. In linguistic, phonemes are written between slashes. For example, phoneme **/k/** occurs in the words such as kit, skit.

Morphology

It is the second component of language. It is the study of the structure and classification of the words in a particular language. The origin of the word is from Greek language, where the word 'morphe' means 'form'. Morphology considers the principles of formation of words in a language. In other words, how sounds combine into meaningful units like prefixes, suffixes and roots. It also considers how words can be grouped into parts of speech.

Lexeme

In linguistics, the abstract unit of morphological analysis that corresponds to a set of forms taken by a single word is called lexeme. The way in which a lexeme is used in a sentence is determined by its grammatical category. Lexeme can be individual word or multiword. For example, the word talk is an example of an individual word lexeme, which may have many grammatical variants like talks, talked and talking. Multiword lexeme can be made up of more than one orthographic word. For example, speak up, pull through, etc. are the examples of multiword lexemes.

Syntax

It is the third component of language. It is the study of the order and arrangement of the words into larger units. The word can be traced to Greek language, where the word suntassein means 'to put in order'. It studies the type of sentences and their structure, of clauses, of phrases.

Semantics

It is the fourth component of language. It is the study of how meaning is conveyed. The meaning can be related to the outside world or can be related to the grammar of the sentence. The word can be traced to Greek language, where the word semainein means means 'to signify', 'show', 'signal'.

Pragmatics:

It is the fifth component of language. It is the study of the functions of the language and its use in context. The origin of the word can be traced to Greek language where the word 'pragma' means 'deed', 'affair'

Spoken Language Syntax

The written English and spoken English grammar have many common features but along with that, they also differ in a number of aspects. The following features distinguish between the spoken and written English grammar −

Disfluencies and Repair

This striking feature makes spoken and written English grammar different from each other. It is individually known as phenomena of disfluencies and collectively as phenomena of repair. Disfluencies include the use of following −

- **Fillers words** − Sometimes in between the sentence, we use some filler words. They are called fillers of filler pause. Examples of such words are uh and um.
- **Reparandum and repair** − The repeated segment of words in between the sentence is called reparandum. In the same segment, the changed word is called repair. Consider the following example to understand this −

Does ABC airlines offer any one-way flights uh one-way fares for 5000 rupees?

In the above sentence, one-way flight is a reparadum and one-way flights is a repair.

Word Fragments

Sometimes we speak the sentences with smaller fragments of words. For example, **wwha-what is the time?** Here the words **w-wha** are word fragments.

NATURAL LANGUAGE GRAMMAR

For linguistics, language is a group of arbitrary vocal signs. We may say that language is creative, governed by rules, innate as well as universal at the same time. On the other hand, it is humanly too. The nature of the language is different for different people. There is a lot of misconception about the nature of the language. That is why it is very important to understand the meaning of the ambiguous term ***'grammar'***. In linguistics, the term grammar may be defined as the rules or principles with the help of which language works. In broad sense, we can divide grammar in two categories −

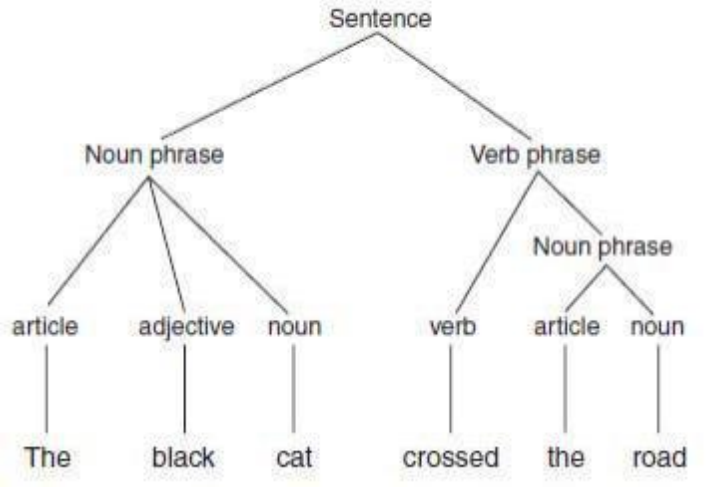
Descriptive Grammar

The set of rules, where linguistics and grammarians formulate the speaker's grammar is called descriptive grammar.

Perspective Grammar

It is a very different sense of grammar, which attempts to maintain a standard of correctness in the language. This category has little to do with the actual working of the language.

Grammatical Categories



A grammatical category may be defined as a class of units or features within the grammar of a language. These units are the building blocks of language and share a common set of characteristics. Grammatical categories are also called grammatical features.

In recent years, AI has evolved rapidly, and with that, NLP got more sophisticated, too. Many of us already use NLP daily without realizing it. You've probably used at least one of the following tools:

- Spell checker.
- Autocomplete.
- Spam filters.
- Voice text messaging.

GENERATIVE CAPACITY

Grammatical formalisms can be classified by their **generative capacity**: the set of languages they can represent. Chomsky (1957) describes four classes of grammatical formalisms that differ only in the form of the rewrite rules. The classes can be arranged in a hierarchy, where each class can be used to describe all the languages that can be described by a less powerful class, as well as some additional languages. Here we list the hierarchy, most powerful class first:

Recursively enumerable grammars use unrestricted rules: both sides of the rewrite rules can have any number of terminal and nonterminal symbols, as in the rule $A\ B\ C \rightarrow D\ E$. These grammars are equivalent to Turing machines in their expressive power.

Context-sensitive grammars are restricted only in that the right-hand side must contain at least as many symbols as the left-hand side. The name "contextsensitive" comes from the fact that a rule such as $A\ X\ B \rightarrow A\ Y\ B$ says that an X can be rewritten as a Y in the context of a preceding A and a following B. Context-sensitive grammars can represent languages such as $a^n b^n c^n$ (a sequence of n copies of a followed by the same number of bs and then cs).

In **context-free grammars** (or **CFG**s), the left-hand side consists of a single nonterminal symbol. Thus, each rule licenses rewriting the nonterminal as the right-hand side in *any* context. CFGs are popular for natural-language and programming-language grammars, although it is now widely accepted that at least some natural languages have constructions that are not context-free (Pullum, 1991). Context-free grammars can represent $a^n b^n$, but not $a^n b^n c^n$.

Regular grammars are the most restricted class. Every rule has a single nonterminal on the left-hand side and a terminal symbol optionally followed by a nonterminal on the right-hand side. Regular grammars are equivalent in power to finitestate machines. They are poorly suited for programming languages, because they cannot represent constructs such as balanced opening and closing parentheses (a variation of the $a^n b^n$ language). The closest they can come is representing $a\ *\ b\ *$, a sequence of any number of as followed by any number of bs.

The grammars higher up in the hierarchy have more expressive power, but the algorithms for dealing with them are less efficient. Up to the 1980s, linguists focused on context-free and context-sensitive languages.

There have been many competing language models based on the idea of phrase structure; we will describe a popular model called the **probabilistic context-free grammar**, or PROBABILISTIC CONTEXT-FREE GRAMMAR PCFG.

1 A **grammar** is a collection of GRAMMAR rules that defines a **language** as a set of allowable LANGUAGE strings of words. "Context-free" and "probabilistic" means that the grammar assigns a probability to every string. Here is a PCFG rule:

$VP \rightarrow Verb\ [0.70]$

$|\ VP\ NP\ [0.30]$.

Here VP (*verb phrase*) and NP (*noun phrase*) are **non-terminal symbols**. The grammar also refers to actual words, which are called **terminal symbols**. This rule is saying that with probability 0.70 a verb phrase consists solely of a

verb, and with probability 0.30 it is a VP followed by an NP. Appendix B describes non-probabilistic context-free grammars.

We now define a grammar for a tiny fragment of English that is suitable for communication between agents exploring the wumpus world. We call this language E_0. Later sections improve on E_0 to make it slightly closer to real English.

We are unlikely ever to devise a complete grammar for English, if only because no two persons would agree entirely on what constitutes valid English.

Five basic NLP tasks

As we mentioned before, human language is extremely complex and diverse. That's why natural language processing includes many techniques to interpret it, ranging from statistical and machine learning methods to rules-based and algorithmic approaches.

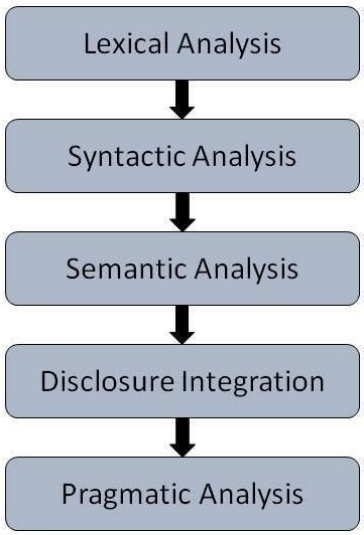
There are five basic NLP tasks that you might recognize from school.

Lexical Analysis − 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. Lexical analysis divides the whole text into paragraphs, sentences, and words. It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language.

Syntactic Analysis (Parsing) − Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as "The school goes to boy" is rejected by English syntactic analyzer.

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.



Semantic Analysis − Semantic analysis is concerned with the meaning representation. It mainly focuses on the literal meaning of words, phrases, and sentences.

It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”.

Discourse Integration − Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

- The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.

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.

- During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

CONCEPT OF GRAMMAR

Grammar is very essential and important to describe the syntactic structure of well-formed programs. In the literary sense, they denote syntactical rules for conversation in natural languages. Linguistics have attempted to define grammars since the inception of natural languages like English, Hindi, etc.

The theory of formal languages is also applicable in the fields of Computer Science mainly in programming languages and data structure. For example, in ‘C’ language, the precise grammar rules state how functions are made from lists and statements.

A mathematical model of grammar was given by **Noam Chomsky** in 1956, which is effective for writing computer languages.

Mathematically, a grammar G can be formally written as a 4-tuple (N, T, S, P) where −

- **N** or $\mathbf{V_N}$ = set of non-terminal symbols, i.e., variables.
- **T** or ∑ = set of terminal symbols.
- **S** = Start symbol where S ∈ N
- **P** denotes the Production rules for Terminals as well as Non-terminals. It has the form α → β, where α and β are strings on $V_N \cup \sum$ and least one symbol of α belongs to V_N

Phrase Structure or Constituency Grammar

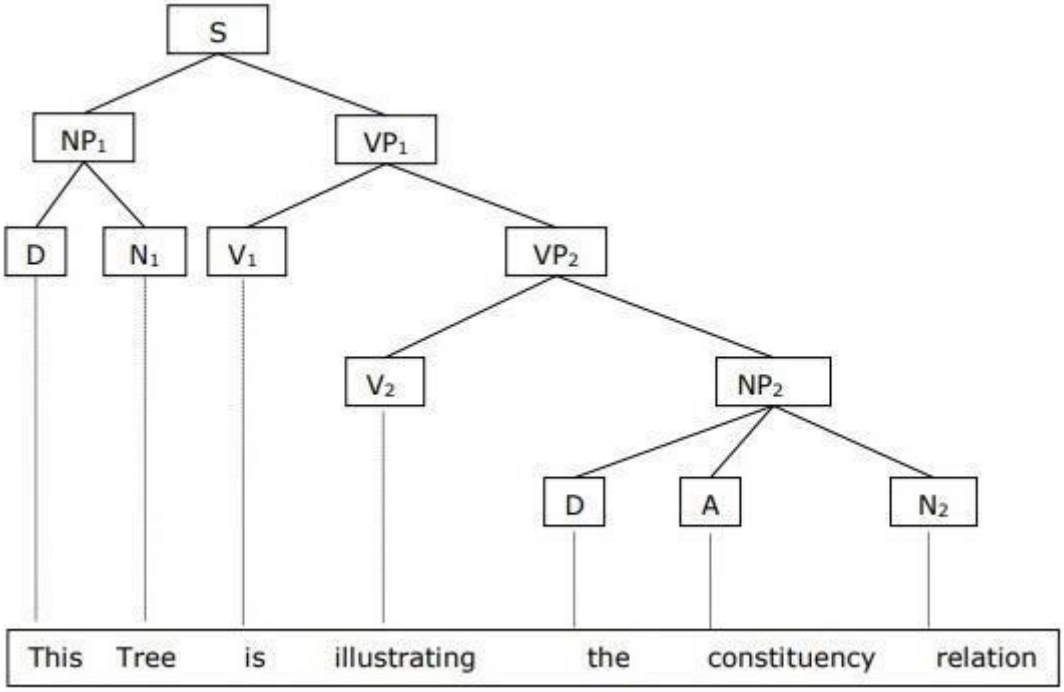
Phrase structure grammar, introduced by Noam Chomsky, is based on the constituency relation. That is why it is also called constituency grammar. It is opposite to dependency grammar.

Example

Before giving an example of constituency grammar, we need to know the fundamental points about constituency grammar and constituency relation.

- All the related frameworks view the sentence structure in terms of constituency relation.
- The constituency relation is derived from the subject-predicate division of Latin as well as Greek grammar.
- The basic clause structure is understood in terms of **noun phrase NP** and **verb phrase VP**.

We can write the sentence **"This tree is illustrating the constituency relation"** as follows −



Dependency Grammar

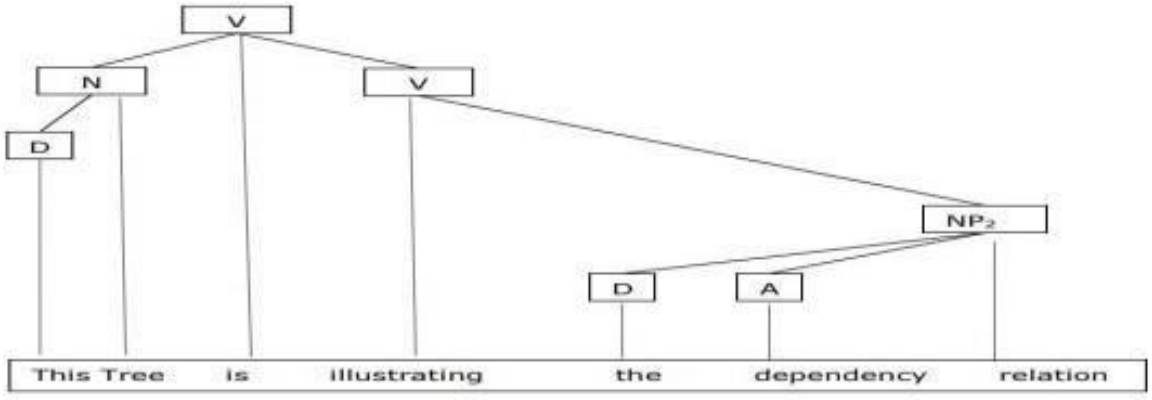
It is opposite to the constituency grammar and based on dependency relation. It was introduced by Lucien Tesniere. Dependency grammar (DG) is opposite to the constituency grammar because it lacks phrasal nodes.

Example

Before giving an example of Dependency grammar, we need to know the fundamental points about Dependency grammar and Dependency relation.

- In DG, the linguistic units, i.e., words are connected to each other by directed links.
- The verb becomes the center of the clause structure.
- Every other syntactic units are connected to the verb in terms of directed link. These syntactic units are called ***dependencies***.

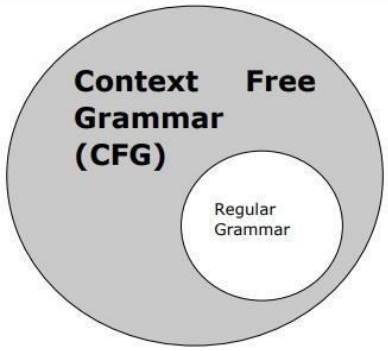
We can write the sentence **"This tree is illustrating the dependency relation"** as follows;



Parse tree that uses Constituency grammar is called constituency-based parse tree; and the parse trees that uses dependency grammar is called dependency-based parse tree.

Context Free Grammar

Context free grammar, also called CFG, is a notation for describing languages and a superset of Regular grammar. It can be seen in the following diagram −



Definition of CFG

CFG consists of finite set of grammar rules with the following four components −

Set of Non-terminals

It is denoted by V. The non-terminals are syntactic variables that denote the sets of strings, which further help defining the language, generated by the grammar.

Set of Terminals

It is also called tokens and defined by Σ. Strings are formed with the basic symbols of terminals.

Set of Productions

It is denoted by P. The set defines how the terminals and non-terminals can be combined. Every production(P) consists of non-terminals, an arrow, and terminals (the sequence of terminals). Non-terminals are called the left side of the production and terminals are called the right side of the production.

Start Symbol

The production begins from the start symbol. It is denoted by symbol S. Non-terminal symbol is always designated as start symbol.

IMPLEMENTATION ASPECTS OF SYNTACTIC ANALYSIS

Parsing is the process of analyzing a string of words to uncover its phrase structure, according to the rules of a grammar.

There are a number of algorithms researchers have developed for syntactic analysis, but we consider only the following simple methods –

- Context-Free Grammar
- Top-Down Parser

Context-Free Grammar

It is the grammar that consists rules with a single symbol on the left-hand side of the rewrite rules. Let us create grammar to parse a sentence −

"The bird pecks the grains"

Articles (DET) − a | an | the

Nouns − bird | birds | grain | grains

Noun Phrase (NP) − Article + Noun | Article + Adjective + Noun

= DET N | DET ADJ N

Verbs − pecks | pecking | pecked

Verb Phrase (VP) − NP V | V NP

Adjectives (ADJ) − beautiful | small | chirping

The parse tree breaks down the sentence into structured parts so that the computer can easily understand and process it. In order for the parsing algorithm to construct this parse tree, a set of rewrite rules, which describe what tree structures are legal, need to be constructed.

These rules say that a certain symbol may be expanded in the tree by a sequence of other symbols. According to first order logic rule, if there are two strings Noun Phrase (NP) and Verb Phrase (VP), then the string combined by NP followed by VP is a sentence. The rewrite rules for the sentence are as follows-

S → NP VP

NP → DET N | DET ADJ N

VP → V NP

Lexocon –

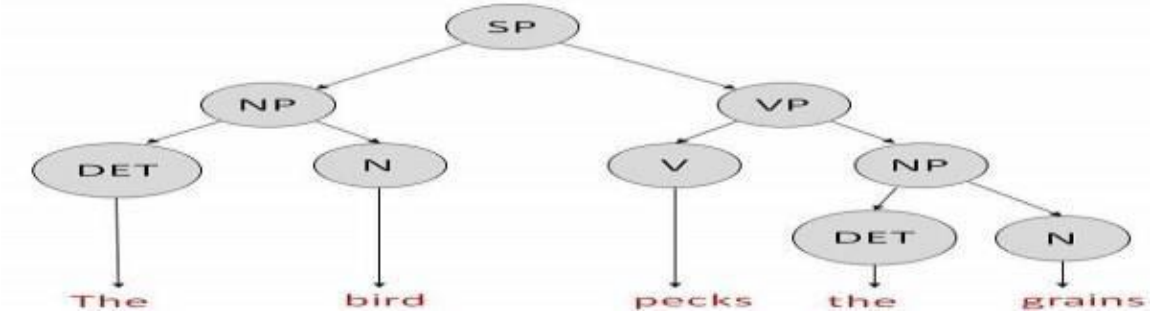
DET → a | the

ADJ → beautiful | perching

N → bird | birds | grain | grains

V → peck | pecks | pecking

The parse tree can be created as shown −



Now consider the above rewrite rules. Since V can be replaced by both, "peck" or "pecks", sentences such as "The bird peck the grains" can be wrongly permitted. i. e. the subject-verb agreement error is approved as correct.

Merit − The simplest style of grammar, therefore widely used one.

Demerits −

- They are not highly precise. For example, “The grains peck the bird”, is a syntactically correct according to parser, but even if it makes no sense, parser takes it as a correct sentence.
- To bring out high precision, multiple sets of grammar need to be prepared. It may require a completely different sets of rules for parsing singular and plural variations, passive sentences, etc., which can lead to creation of huge set of rules that are unmanageable.

Top-Down Parser

Here, the parser starts with the S symbol and attempts to rewrite it into a sequence of *terminal symbols* that matches the classes of the words in the input sentence until it consists entirely of terminal symbols.

These are then checked with the input sentence to see if it matched. If not, the process is started over again with a different set of rules. This is repeated until a specific rule is found which describes the structure of the sentence.

Merit − It is simple to implement.

Demerits −

- It is inefficient, as the search process has to be repeated if an error occurs.
- Slow speed of working.

Natural Language Processing - Semantic Analysis

The purpose of semantic analysis is to draw exact meaning, or you can say dictionary meaning from the text. The work of semantic analyzer is to check the text for meaningfulness.

We already know that lexical analysis also deals with the meaning of the words, then how is semantic analysis different from lexical analysis? Lexical analysis is based on smaller token but on the other side semantic analysis focuses on larger chunks. That is why semantic analysis can be divided into the following two parts −

Studying meaning of individual word

It is the first part of the semantic analysis in which the study of the meaning of individual words is performed. This part is called lexical semantics.

Studying the combination of individual words

In the second part, the individual words will be combined to provide meaning in sentences.

The most important task of semantic analysis is to get the proper meaning of the sentence. For example, analyze the sentence **"Ram is great."** In this sentence, the speaker is talking either about Lord Ram or about a person whose name is Ram. That is why the job, to get the proper meaning of the sentence, of semantic analyzer is important.

Elements of Semantic Analysis

Followings are some important elements of semantic analysis −

Hyponymy

It may be defined as the relationship between a generic term and instances of that generic term. Here the generic term is called hypernym and its instances are called hyponyms. For example, the word color is hypernym and the color blue, yellow etc. are hyponyms.

Homonymy

It may be defined as the words having same spelling or same form but having different and unrelated meaning. For example, the word "Bat" is a homonymy word because bat can be an implement to hit a ball or bat is a nocturnal flying mammal also.

Polysemy

Polysemy is a Greek word, which means "many signs". It is a word or phrase with different but related sense. In other words, we can say that polysemy has the same spelling but different and related meaning. For example, the word "bank" is a polysemy word having the following meanings −

- A financial institution.
- The building in which such an institution is located.
- A synonym for "to rely on".

Difference between Polysemy and Homonymy

Both polysemy and homonymy words have the same syntax or spelling. The main difference between them is that in polysemy, the meanings of the words are related but in homonymy, the meanings of the words are not related. For example, if we talk about the same word "Bank", we can write the meaning 'a financial institution' or 'a river bank'. In that case it would be the example of homonym because the meanings are unrelated to each other.

Synonymy

It is the relation between two lexical items having different forms but expressing the same or a close meaning. Examples are 'author/writer', 'fate/destiny'.

Antonymy

It is the relation between two lexical items having symmetry between their semantic components relative to an axis. The scope of antonymy is as follows −

- **Application of property or not** − Example is 'life/death', 'certitude/incertitude'
- **Application of scalable property** − Example is 'rich/poor', 'hot/cold'
- **Application of a usage** − Example is 'father/son', 'moon/sun'.

Meaning Representation

Semantic analysis creates a representation of the meaning of a sentence. But before getting into the concept and approaches related to meaning representation, we need to understand the building blocks of semantic system.

Building Blocks of Semantic System

In word representation or representation of the meaning of the words, the following building blocks play an important role −

- **Entities** − It represents the individual such as a particular person, location etc. For example, Haryana. India, Ram all are entities.
- **Concepts** − It represents the general category of the individuals such as a person, city, etc.
- **Relations** − It represents the relationship between entities and concept. For example, Ram is a person.
- **Predicates** − It represents the verb structures. For example, semantic roles and case grammar are the examples of predicates.

Now, we can understand that meaning representation shows how to put together the building blocks of semantic systems. In other words, it shows how to put together entities, concepts, relation and predicates to describe a situation. It also enables the reasoning about the semantic world.

Approaches to Meaning Representations

Semantic analysis uses the following approaches for the representation of meaning −

- First order predicate logic (FOPL)
- Semantic Nets
- Frames
- Conceptual dependency (CD)
- Rule-based architecture
- Case Grammar
- Conceptual Graphs

Need of Meaning Representations

A question that arises here is why do we need meaning representation? Followings are the reasons for the same −

Linking of linguistic elements to non-linguistic elements

The very first reason is that with the help of meaning representation the linking of linguistic elements to the non-linguistic elements can be done.

Representing variety at lexical level

With the help of meaning representation, unambiguous, canonical forms can be represented at the lexical level.

Can be used for reasoning

Meaning representation can be used to reason for verifying what is true in the world as well as to infer the knowledge from the semantic representation.

Lexical Semantics

The first part of semantic analysis, studying the meaning of individual words is called lexical semantics. It includes words, sub-words, affixes (sub-units), compound words and phrases also. All the words, sub-

words, etc. are collectively called lexical items. In other words, we can say that lexical semantics is the relationship between lexical items, meaning of sentences and syntax of sentence.

Following are the steps involved in lexical semantics –

- Classification of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Decomposition of lexical items like words, sub-words, affixes, etc. is performed in lexical semantics.
- Differences as well as similarities between various lexical semantic structures is also analyzed.

MACHINE TRANSLATION

All translation systems must model the source and target languages, but systems vary in the type of models they use. Some systems attempt to analyze the source language text all the way into an interlingua knowledge representation and then generate sentences in the target language from that representation. This is difficult because it involves three unsolved problems: creating a complete knowledge representation of everything; parsing into that representation and generating sentences from that representation.

Other systems are based on a **transfer model**. They keep a database of translation rules (or examples), and whenever the rule (or example) matches, they translate directly. Transfer can occur at the lexical, syntactic, or semantic level.

What is a machine translation and how does it work?

Machine Translation or MT or robotized interpretation is simply a procedure when a computer software translates text from one language to another without human contribution. At its fundamental level, machine translation performs a straightforward replacement of atomic words in a single characteristic language for words in another.

Using corpus methods, more complicated translations can be conducted, taking into account better treatment of contrasts in phonetic typology, express acknowledgement, and translations of idioms, just as the seclusion of oddities. Currently, some systems are not able to perform just like a human translator, but in the coming future, it will also be possible.

In simple language, we can say that ***machine translation works by using computer software to translate the text from one source language to another target language.***

The term 'machine translation' (MT) refers to computerized systems responsible for producing translations with or without human assistance. It excludes computer-based translation tools that support translators by providing access to online dictionaries, remote terminology databanks, transmission and reception of texts, etc.

Before the AI technology era, computer programs for the automatic translation of text from one language to another were developed. In recent years, AI has been tasked with making the automatic or machine translation of human languages' fluidity and versatility of scripts, dialects, and variations. Machine translation is challenging given the inherent ambiguity and flexibility of human language.

Different types of machine translation in NLP

There are four types of machine translation:

1. **Statistical Machine Translation or SMT**

   It works by alluding to statistical models that depend on the investigation of huge volumes of bilingual content. It expects to decide the correspondence between a word from the source language and a word from the objective language. ***A genuine illustration of this is Google Translate.***

   Presently, SMT is extraordinary for basic translation, however its most noteworthy disadvantage is that it doesn't factor in context, which implies translation can regularly be wrong or you can say, don't expect great quality translation. There are several types of statistical-based machine translation models which are: ***Hierarchical phrase-based translation, Syntax-based translation, Phrase-based translation, Word-based translation.***

2. **Rule-based Machine Translation or RBMT**

   RBMT basically translates the basics of ***grammatical rules***. It directs a grammatical examination of the source language and the objective language to create the translated sentence. But, RBMT requires broad editing, and its substantial reliance on dictionaries implies that proficiency is accomplished after a significant period. (Also read: Top 10 Natural Processing Languages (NLP) Libraries with Python)

3. **Hybrid Machine Translation or HMT**

   HMT, as the term demonstrates, is a mix of RBMT and SMT. It uses a translation memory, making it unquestionably more successful regarding quality. Nevertheless, even HMT has a lot of downsides, the biggest of which is the requirement for enormous editing, and human translators will also be needed. There are several approaches to HMT like multi-engine, statistical rule generation, multi-pass, and confidence-based.

4. **Neural Machine Translation or NMT**

   NMT is a type of machine translation that relies upon neural network models (based on the human brain) to build statistical models with the end goal of translation. The essential advantage of NMT is that it gives a solitary system that can be prepared to unravel the source and target text. Subsequently, it doesn't rely upon specific systems that are regular to other machine translation systems, particularly SMT.

What are the benefits of machine translation?

One of the crucial benefits of machine translation is speed as you have noticed that computer programs ***can translate a huge amount of text rapidly.*** Yes, the human translator does their work more accurately but they cannot match the speed of the computer.

If you especially train the machine to your requirements, machine translation gives the ideal blend of brisk and cost-effective translations as ***it is less expensive*** than using a human translator. With a specially trained machine, MT can catch the setting of full sentences before translating them, which gives you high quality and human-sounding yield. Another benefit of machine translation is its ***capability to learn important words and reuse them wherever they might fit.***

Applications of machine translation

Machine translation technology and products have been used in numerous application situations, for example, business travel, the travel industry, etc. In terms of the object of translation, there are composed language-oriented content text translation and spoken language.

Machine Translation vs Human translation

- Machine translation hits that sweet spot of ***cost and speed***, offering a truly snappy path for brands to translate their records at scale without much overhead. Yet, that doesn't mean it's consistently relevant. On the other hand, human translation is incredible for those undertakings that require additional consideration and subtlety. Talented translators work on your image's substance to catch the first importance and pass on that feeling or message basically in another assortment of work.

- Leaning upon how much content should be translated, the machine translation can give translated content very quickly, though human translators will take additional time. Time spent finding, verifying, and dealing with a group of translators should likewise be considered.

- Numerous translation programming providers can give machine translations at practically zero cost, making it a reasonable answer for organizations who will be unable to manage the cost of expert translations.

- Machine Translation is the instant modification of text from one language to another utilizing artificial intelligence whereas a human translation, includes actual brainpower, in the form of one or more translators translating the text manually.

- **Text translation**

  Automated text translation is broadly used in an assortment of sentence-level and text-level translation applications. Sentence-level translation applications incorporate the translation of inquiry and recovery inputs and the translation of (OCR) outcomes of picture optical character acknowledgement. Text-level translation applications incorporate the translation of a wide range of unadulterated reports, and the translation of archives with organized data.

  Organized data mostly incorporates the presentation configuration of text content, object type activity, and other data, for example, textual styles, colours, tables, structures, hyperlinks, etc. Presently, the translation objects of machine translation systems are mostly founded on the sentence level.

  Most importantly, a sentence can completely communicate a subject substance, which normally frames an articulation unit, and the significance of each word in the sentence can be resolved to an enormous degree as per the restricted setting inside the sentence.

  Also, the methods and nature of getting data at the sentence level granularity from the preparation corpus are more effective than that dependent on other morphological levels, for example, words, expressions, and text passages. Finally, the translation depends on sentence-level can be normally reached out to help translation at other morphological levels.

- **Other applications**

Naturally, the task of machine translation is to change one source language word succession into another objective language word grouping which is semantically the same. Generally, it finishes a grouping transformation task, which changes over a succession object into another arrangement object as indicated by some information and rationale through model and algorithms.

All things considered, many undertaking situations total the change between grouping objects, and the language in the machine translation task is just one of the succession object types. In this manner, when the ideas of the source language and target language are stretched out from dialects to other arrangement object types, machine translation strategies and techniques can be applied to settle numerous comparable change undertakings.

Speech translation : **Speech recognition** is the task of identifying a sequence of words uttered by a speaker, given the acoustic signal. It has become one of the mainstream applications of AI—millions of people interact with speech recognition systems every day to navigate voice mail systems, search the Web from mobile phones, and other applications. Speech is an attractive option when hands-free operation is necessary, as when operating machinery.

Speech recognition is difficult because the sounds made by a speaker are ambiguous and, well, noisy. As a well-known example, the phrase “recognize speech” sounds almost the same as “wreck a nice beach” when spoken quickly.

A speech understanding system must answer three questions:
1. What speech sounds did the speaker utter?
2. What words did the speaker intend to express with those speech sounds?
3. What meaning did the speaker intend to express with those words?

With the fast advancement of mobile applications, voice input has become an advantageous method of human-computer cooperation, and discourse translation has become a significant application situation. The fundamental cycle of discourse interpretation is "source language discourse source language text-target language text-target language discourse".

In this cycle, programmed text translation from source language text to target-language text is an important moderate module. What's more, the front end and back end likewise need programmed discourse recognition, ASR and text-to-speech, TTs.

The quality of a speech recognition system depends on the quality of all of its components the language model, the word-pronunciation models, the phone models, and the signal processing algorithms used to extract spectral features from the acoustic signal.

SPEECH RECOGNITION

Speech recognition refers to a computer interpreting the words spoken by a person and converting them to a format that is understandable by a machine. Depending on the end-goal, it is then converted to text or voice or another required format.

For instance, Apple's Siri and Google's Alexa use AI-powered speech recognition to provide voice or text support whereas voice-to-text applications like Google Dictate transcribe your dictated words to text. Voice recognition is another form of speech recognition where a source sound is recognized and matched to a person's voice.

Speech recognition AI applications have seen significant growth in numbers in recent times as businesses are increasingly adopting digital assistants and automated support to streamline their services. Voice assistants, smart home devices, search engines, etc are a few examples where speech recognition has seen prominence. As per Research and Markets, the global market for speech recognition is estimated to grow at a CAGR of 17.2% and reach $26.8 billion by 2025.

Speech recognition is fast overcoming the challenges of poor recording equipment and noise cancellation, variations in people's voices, accents, dialects, semantics, contexts, etc using artificial intelligence and machine learning. This also includes challenges of understanding human disposition, and the varying human language elements like colloquialisms, acronyms, etc. The technology can provide a 95% accuracy now as compared to traditional models of speech recognition, which is at par with regular human communication.

Furthermore, it is now an acceptable format of communication given the large companies that endorse it and regularly employ speech recognition in their operations. It is estimated that a majority of search engines will adopt voice technology as an integral aspect of their search mechanism.

Speech recognition and AI play an integral role in NLP models in improving the accuracy and efficiency of human language recognition.

Use Cases of Speech Recognition

Let's explore the uses of speech recognition applications in different fields:

1. Voice-based speech recognition software is now used to initiate purchases, send emails, transcribe meetings, doctor appointments, and court proceedings, etc.
2. Virtual assistants or digital assistants and smart home devices use voice recognition software to answer questions, provide weather news, play music, check traffic, place an order, and so on.
3. Companies like Venmo and PayPal allow customers to make transactions using voice assistants. Several banks in North America and Canada also provide online banking using voice-based software.
4. Ecommerce is significantly powered by voice-based assistants and allows users to make purchases quickly and seamlessly.
5. Speech recognition is poised to impact transportation services and streamline scheduling, routing, and navigating across cities.
6. Podcasts, meetings, and journalist interviews can be transcribed using voice recognition. It is also used to provide accurate subtitles to a video.
7. There has been a huge impact on security through voice biometry where the technology analyses the varying frequencies, tone and pitch of an individual's voice to create a voice profile. An example of this is Switzerland's telecom company Swisscom which has enabled voice authentication technology in its call centres to prevent security breaches.

8. Customer care services are being traced by AI-based voice assistants, and chatbots to automate repeatable tasks.

As speech recognition and AI impact both professional and personal lives at workplaces and homes respectively, the demand for skilled AI engineers and developers, Data Scientists, and Machine Learning Engineers, is expected to be at an all-time high.

There will be a requirement for skilled AI professionals to enhance the relationship between humans and digital devices. As job opportunities are created, they will result in increased perks and benefits for those in this field.

CHAPTER-2:

WHAT IS PERCEPTION IN AI?

Perception is a process to interpret, acquire, select and then organize the sensory information that is captured from the real world. Perception provides agents with information about the world they inhabit. Perception is initiated by **sensors.** A sensor is anything that can change the computational state of the agent in response to a change in the state of the world. It could be as simple as a one-bit sensor that detects whether a switch is on or off, or as complex as the retina of the human eye, which contains more than a hundred million photosensitive elements.

For example: Human beings have sensory receptors such as touch, taste, smell, sight and hearing. So, the information received from these receptors is transmitted to human brain to organize the received information.

Perception in Artificial Intelligence is the process of interpreting vision, sounds, smell, and touch. Perception helps to build machines or robots that react like humans. Perception is a process to interpret, acquire, select, and then organize the sensory information from the physical world to make actions like humans. The main difference between AI and robot is that the robot makes actions in the real world.

According to the received information, action is taken by interacting with the environment to manipulate and navigate the objects.

Perception provides agents with information about the world they inhabit by interpreting the response of **sensors**. A sensor measures some aspect of the environment in a form that can be used as input by an agent program. The sensor could be as simple as a switch, which gives one bit telling whether it is on or off, or as complex as the eye. A variety of sensory modalities are available to artificial agents.

Perception and action are very important concepts in the field of Robotics. The following figures show the complete **autonomous robot.**

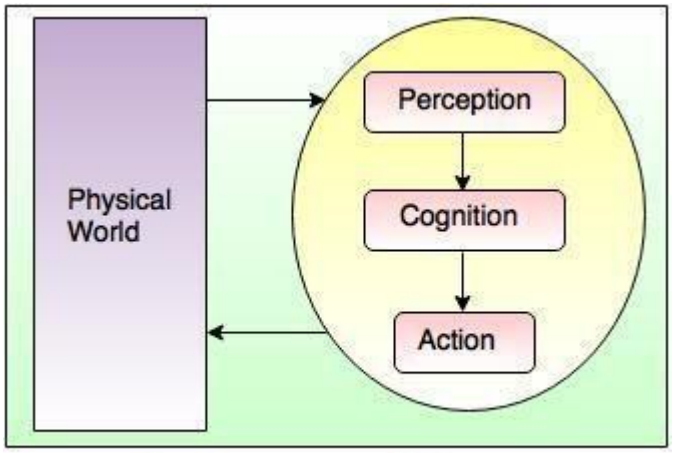


Fig: Autonomous Robot

There is one important difference between the artificial intelligence program and robot. The AI program performs in a computer stimulated environment, while the robot performs in the physical world.

For example:

In chess, an AI program can be able to make a move by searching different nodes and has no facility to touch or sense the physical world.

However, the chess playing robot can make a move and grasp the pieces by interacting with the physical world.

Image formation in digital camera

Image formation is a physical process that captures object in the scene through lens and creates a 2-D image.

Let's understand the geometry of a pinhole camera shown in the following diagram.

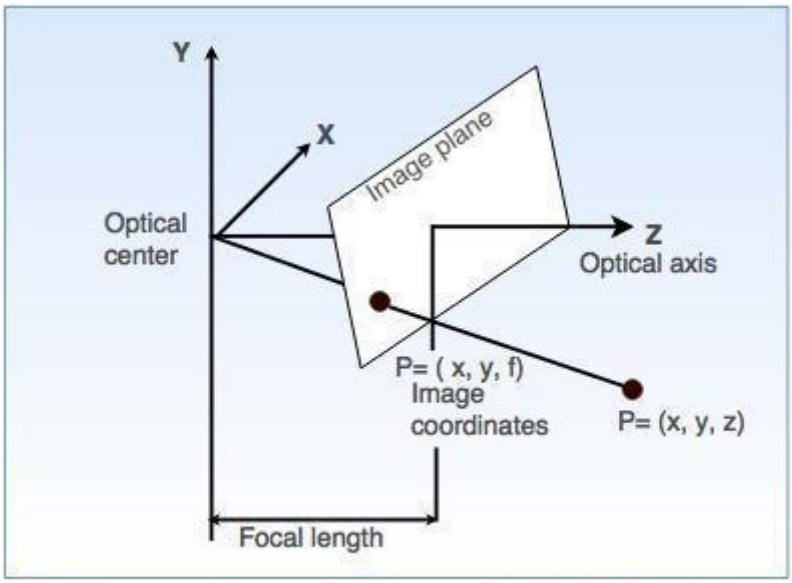


Fig: Geometry of Image Formation in the pinhole camera

In the above figure, an optical axis is perpendicular to the image plane and image plane is generally placed in front of the optical center.

So, let **P** be the point in the scene with coordinates (X,Y,Z) and **P'** be its image plane with coordinates (x, y, z).

If the focal length from the optical center is **f**, then by using properties of similar triangles, equation is derived as,

$-x/f = X/Z$ so $x = -fX/Z$ .....equation (i)
$-y/f = -Y/Z$ so $y = -fY/Z$ .....equation (ii)

These equations define an image formation process called as **perspective projection.**

EARLY IMAGE PROCESSING OPERATIONS

From the inception of AI, image processing to be incorporated in smart systems is a perennial project for people working on it. In its initial phase, it required a lot of manual input, providing instructions to computers, to get some output. These machines,or known as Expert Systems, were trained to recognize images.

There are 2 methods of image processing:

- – Analog image processing, which is used for processing photographs, printouts, and other image hard copies.
- – Digital image processing, which is used for manipulating digital images with the help of complex algorithms

Main Purpose of Image Processing

- – Representing processed data in a visual way one can understand, for instance, giving a visual form to invisible objects.
- – To improve the processed image quality, image sharpening and restoration works well.
- – Image convalescence helps in searching images.
- – Helps to measure objects in the image.
- – With pattern recognition, it becomes easy to classify objects in the image, locate their position and get an overall understanding of the scene.

Image Processing Phases

There are 8 phases for image processing which goes step-wise:

- **Image acquisition**:
  Captures the image with a sensor and converts it into a manageable entity
- **Image enhancement**
  The input image quality is improved and also extracts details hidden in it
- **Image restoration**
  Any possible corruption like blur, noise, or camera misfocus is removed to get a cleaner vision on probabilistic and mathematical model basis
- **Color image processing**
  The colored images and varied color spaces are processed with pseudocolor or RGB processing way.
- **Image compression and decompression**

This allows for changes in image resolution and size, be it for reduction or restoring images depending on the need.

- **Morphological processing**
  Defines the object structure and shape in the image.
- **Image recognition**
  For a particular object, the specific features are identified in the image and techniques like object detection are used for the same.
- **Representation and description**
  is all about visualizing the processed data.

IMAGE PROCESSING METHODS, TECHNIQUES, AND TOOLS

The images captured with regular sensors need preprocessing as some could contain too much noise or are misfocused. There are two detection techniques to be used for processing digital images as well as for preprocessing.

- **Filtering**
  Used to modify and enhance the input image. With various filters available, certain features in the image can be emphasized or removed, can also reduce the image noise and so on.
- **Edge detection**
  Used for data extraction and image segmentation, to find meaningful object edges in the images that are preprocessed.

To make things easier, there are specific libraries and frameworks that can be used to implement image processing functionalities.

Object Recognition from Structural information,

Conventionally the recognition of objects in a scene is achieved by digital image processing [11], whose purpose is directly related to artificial vision or computer vision. Artificial vision aims to detect, segment, locate, and recognize particular objects. Thus, the following methodology is proposed: (a) segmentation, (b) intelligent recognition, and, (c) extraction of characteristics.

A. Segmentation

The proposed image processing consists of converting the image to grayscale, obtaining a lighter image format. To obtain a lighter image format is needed to convert on grayscale images. Then, edges of the images are detected by the derivative method; before the image is enlarged and eroded to close the found edges. Finally, the borders are filled, achieving a mask that identifies the position of the object inside the image. Each step is described in detail below:

1) Greyscale Transformation. It consists of determining the equivalent of the luminance, which is defined as the light received on a surface being defined as the relation of the luminous flux on the illuminated area [12], luminance concept is associated with human eye perception of different light intensity [13]. The luminance is calculated based on the weighted average of the color components of each pixel, as shown in Equation 1, where L corresponds to the luminance, R is the red component. G the green component and B the blue component.

$$L = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad (1)$$

Equation 1 should be used for calculation grayscale to each pixel; this process is required for the image edge detection.

2) Images Edges Detection. The image obtained in the previous step can be represented as a discrete function in 2D, which is defined by the coordinates of each pixel *m* and *n*. The discrete value of the function is evaluated at a specific point; its procedure is known as brightness or pixel intensity. An edge is defined as tone changes between pixels, in cases where changes exceed a threshold value, it is considered an edge. Different methods to identify edges have been proposed, one of them is the intensity gradient of each pixel, using a convolution mask, then magnitude is calculated finally, the threshold process is applied [14].

The most used edge detection techniques employ local operators [15], using discrete approximations of the first and second of grayscales images, hereunder it will be described the proposed operator, which is based on the first derivative of the image.

What is the purpose of edge detection/image detection?

Edges are straight lines or curves in the image plane across which there is a "significant" change in image brightness. The goal of edge detection is to abstract away from the messy, multi megabyte image and toward a more compact, abstract representation

- Edge detection operation is used in an image processing.
- The main goal of edge detection is to construct the ideal outline of an image.
  - Discontinuity in brightness of image is affected due to:

i) Depth discontinuities
ii) Surface orientation discontinuities
iii) Reflectance discontinuities
iv) Illumination.

B. Artificial Intelligence for Object Recognition

- The image is reconstructed using a Hopfield neural network; the network eliminates noise in edge image, improving accuracy and precision, and obtaining a proper segmented image.
- Artificial intelligence is known as "the science and engineering of making intelligent machines, especially intelligent computer programs", this definition was proposed by Professor John Mccarthy, in 1956 [19]. The target of artificial intelligence is to think, evaluate or act according to certain inputs to exercise some specific function, to achieve this, different processes could be performed: i.) Genetic algorithms, ii.) Artificial neural networks and iii.) Formal logic.
- For the specific problem, artificial neural networks are employed, it uses elements information processors, where local interactions depend on overall system performance.[20] Networks consist of a large number of simple processing elements called nodes or neurons that are arranged in layers [21]. Each neuron is connected to other neurons by communication links, which have a weight associated. Weights represent the information that will be used by the neural network to solve a given problem [22]. Now, there are various types of neural networks, such as self-organization, recurrent, among others.

3D-Information extraction using vision

Why extraction of 3-D information is necessary?

The 3-D information extraction process plays an important role to perform the tasks like manipulation, navigation and recognition. It deals with the following aspects:

1. Segmentation of the scene

The segmentation is used to arrange the array of image pixels into regions. This helps to match semantically meaningful entities in the scene.

- The goal of segmentation is to divide an image into regions which are homogeneous.
- The union of the neighboring regions should not be homogeneous.
- **Thresholding** is the simplest technique of **segmentation.** It is simply performed on the object, which has an homogeneous intensity and a background with a different intensity level and the pixels are partitioned depending on their intensity values.

2. To determine the position and orientation of each object
Determination of the position and orientation of each object relative to the observer is important for manipulation and navigation tasks.
For example: Suppose a person goes to a store to buy something. While moving around he must know the locations and obstacles, so that he can make the plan and path to avoid them.

- The whole orientation of image should be specified in terms of a three dimensional rotation.

3. To determine the shape of each and every object
When the camera moves around an object, the distance and orientation of that object will change but it is important to preserve the shape of that object.
For example: If an object is cube, that fact does not change, but it is difficult to represent the global shape to deal with wide variety of objects present in the real world.

- If the shape of an object is same for some manipulating tasks, it becomes easy to decide how to grasp that object from a particular place.
- The **object recognition** plays most significant role to identify and classify the objects as an example only when the geometric shapes are provided with color and texture.

However, a question arises that, how should we recover 3-D image from the pinhole camera?

There are number of techniques available in the visual stimulus for 3D-image extraction such as **motion, binocular stereopsis, texture, shading, and contour**. Each of these techniques operates on the background assumptions about physical scene to provide interpretation.

EXTRACTING 3-D INFORMATION USING VISION

We need to extract 3-D information for performing certain tasks such as manipulation, navigation, and recognition. There are threte aspects of this:
1. Segmentation of the scene into distinct objects.
2. Determining the position and orientation of each object relative to the observer.
3. Determining the shape of each object.

Segmentation of the image is a key step towards organizing the array of image pixels into regions that would correspond to semantically meaningful entities in the scene. For recognition, we would like to know what features belong together so that one could compare them with stored models; to grasp an object, one needs to know what belongs together as an object.
Segmentation is best viewed as part of extraction of 3-D information about the scene.

One of the principal uses of vision is to provide information for manipulating objects—picking them up, grasping, twirling, and so on—as well as navigating in a scene while avoiding obstacles. The capability to use vision for these purposes is present in the most primitive of animal visual systems. Perhaps the evolutionary origin of the vision sense can be traced back to the presence of a photosensitive spot on one end of an organism that enabled it to orient itself toward (or away from) the light. Flies use vision based on optic flow to control their landing responses. Mobile robots moving around in an environment need to know where the obstacles are, where free space corridors are available, and so on.

******

ARTIFICIAL INTELLIGENCE

UNIT-V

SYLLABUS:

Robotics: Introduction, Robot Hardware, Robotic Perception, Planning to move, planning uncertain movements, Moving, Robotic software architectures, application domains.

Philosophical foundations: Weak AI, Strong AI, Ethics and Risks of AI, Agent Components, Agent Architectures, Are we going in the right direction, What if AI does succeed.

AI Unit-5.3: ROBOT:

Robots are physical agents that perform tasks by manipulating the physical world.

Effectors have a single purpose that to assert physical forces on the environment. Robots are also equipped with **sensors**, which allow them to perceive their environment.

Most of today's robots fall into one of three primary categories.

1.MANIPULATORS:

Manipulator motion usually involves a chain of controllable joints, enabling such robots to place their effectors in any position within the workplace. Few car manufacturers could survive without robotic manipulators, and some manipulators have even been used to generate original artwork.

2.MOBBILE ROBOT:

The second category is the **mobile robot**. Mobile robots move about their environment using wheels, legs, or similar mechanisms. They have been put to use delivering food in hospitals, moving containers at loading docks, and similar tasks. Other types of mobile robots include **unmanned air vehicles, Autonomous underwater vehicles etc..,**

3.MOBILE MANIPULATOR:

The third type of robot combines mobility with manipulation, and is often called a **mobile manipulator**. **Humanoid robots** mimic the human torso.

The field of robotics also includes prosthetic devices , intelligent environments and multibody systems, wherein robotic action is achieved through swarms of small cooperating robots. Robotics brings together many of the concepts we have seen earlier in the book, including probabilistic state estimation, perception, planning, unsupervised learning, and reinforcement learning.

ROBOT HARDWARE:

The robot hardware mainly depends on 1.sensors and 2.effectors

1.sensors:

Sensors are the perceptual interface between robot and environment.

PASSIVE SENSOR: Passive sensors, such as cameras, are true observers of the environment: they capture signals that are generated by other sources in the environment.

ACTIVE SENSOR: Active sensors, such as sonar, send energy into the environment. They rely on the fact that this energy is reflected back to the sensor.

Range finders are sensors that measure the distance to nearby objects. In the early days of robotics, robots were commonly equipped with **sonar sensors**. Sonar sensors emit directional sound waves, which are reflected by objects, with some of the sound making it back into the sensor.

Stereo vision relies on multiple cameras to image the environment from slightly different viewpoints, analyzing the resulting parallax in these images to compute the range of surrounding objects.

Other common range sensors include radar, which is often the sensor of choice for UAVs. Radar sensors can measure distances of multiple kilometers. On the other extreme end of range sensing are **tactile sensors** such as whiskers, bump panels, and touch-sensitive skin.

A second important class of sensors is **location sensors**. Most location sensors use range sensing as a primary component to determine location. Outdoors, the **Global Positioning System** (GPS) is the most common solution to the localization problem. GPS measures the distance to satellites that emit pulsed signals.

The third important class is **proprioceptive sensors**, which inform the robot of its own motion. To measure the exact configuration of a robotic joint, motors are often equipped with **shaft decoders** that count the revolution of motors in small increments.

Other important aspects of robot state are measured by **force sensors** and **torque sensors**. These are indispensable when robots handle fragile objects or objects whose exact shape and location is unknown.

EFFECTORS:

Effectors are the means by which robots move and change the shape of their bodies. To understand the design of effectors we use the concept of degree of freedom.

We count one degree of freedom for each independent direction in which a robot, or one of its effectors, can move. For example, a rigid mobile robot such as an AUV has six degrees of freedom, three for its (x, y, z) location in space and three for its angular orientation, known as *yaw*, *roll*, and *pitch*. These six degrees define the **kinematic state**2 or **pose** of the robot. The **dynamic state** of a robot includes these six plus an additional six dimensions for the rate of change of each kinematic dimension, that is, their velocities.

For nonrigid bodies, there are additional degrees of freedom within the robot itself. For example, the elbow of a human arm possesses two degree of freedom. It can flex the upper arm towards or away, and can rotate right or left. The wrist has three degrees of freedom. It can move up and down, side to side, and can also rotate. Robot joints also have one, two, or three degrees of freedom each. Six degrees of freedom are required to place an object, such as a hand, at a particular point in a particular orientation.

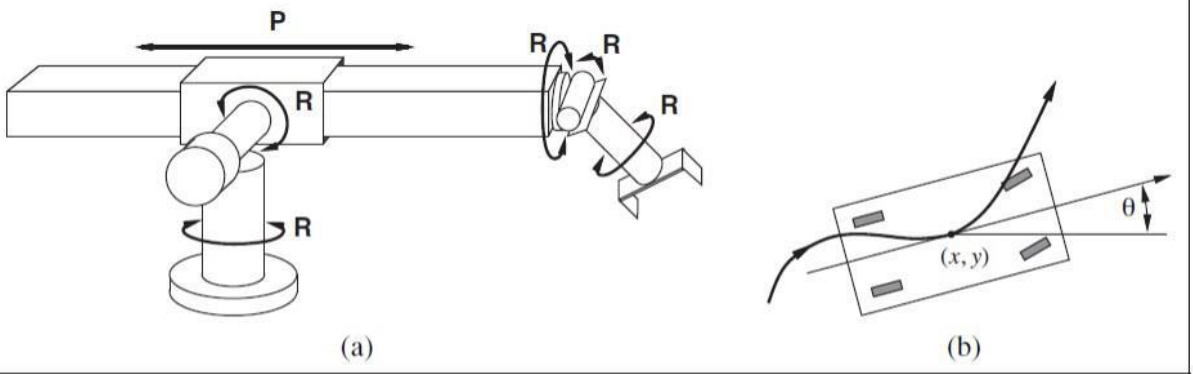


Figure 25.4 (a) The Stanford Manipulator, an early robot arm with five revolute joints (R) and one prismatic joint (P), for a total of six degrees of freedom. (b) Motion of a nonholonomic four-wheeled vehicle with front-wheel steering.

In the fig 4(a) has exactly six degrees of freedom, created REVOLUTE JOINT by five **revolute joints** that generate rotational motion and one **prismatic joint** that generates sliding motion

For mobile robots, the DOFs are not necessarily the same as the number of actuated elements.
Consider, for example, your average car: it can move forward or backward, and it can turn, giving it two DOFs. In contrast, a car's kinematic configuration is three-dimensional: on an open flat surface, one can easily maneuver a car to any (x, y) point, in any orientation. (See Figure 25.4(b).) Thus, the car has three **effective degrees of freedom** but two **control label degrees of freedom**. We say a robot is **nonholonomic** if it has more effective DOFs than controllable DOFs and **holonomic** if the two numbers are the same.

Sensors and effectors alone do not make a robot. A complete robot also needs a source of power to drive its effectors. The **electric motor** is the most popular mechanism for both manipulator actuation and locomotion, but **pneumatic actuation** using compressed gas and **Hydraulic actuation** using pressurized fluids also have their application niches.

ROBOTIC PERCEPTION:

Perception is the process by which robots map sensor measurements into internal representations of the environment. Perception is difficult because sensors are noisy, and the environment is partially observable, unpredictable, and often dynamic.

As a rule of thumb, good internal representations for robots have three properties: they contain enough information for the robot to make good decisions, they are structured so that they can be updated efficiently, and they are natural in the sense that internal variables correspond to natural state variables in the physical world.

For robotics problems, we include the robot's own past actions as observed variables in the model. Figure 25.7 shows the notation used in this

chapter: **X**t is the state of the environment (including the robot) at time t, **Z**t is the observation received at time t, and At is the action taken after the observation is received.

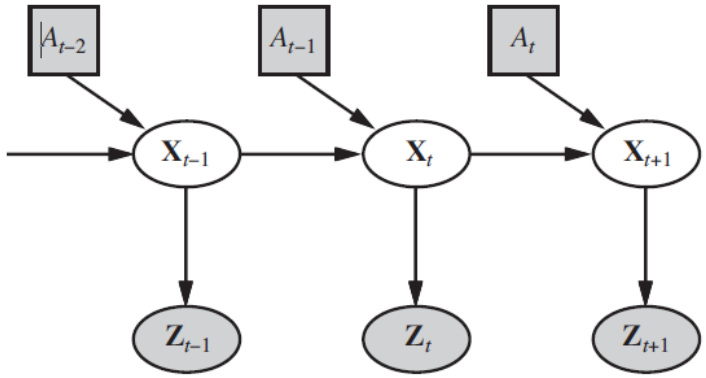


Figure 25.7 Robot perception can be viewed as temporal inference from sequences of actions and measurements, as illustrated by this dynamic Bayes network.

We would like to compute the new belief state, **P**(**X**t+1 | **z**1:t+1, a1:t), from the current belief state **P**(**X**t | **z**1:t, a1:t−1) and the new observation **z**t+1. Thus, we modify the recursive filtering equation (15.5 on page 572) to use integration rather than summation:

$$\mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{z}_{1:t+1}, a_{1:t})$$

$$= \alpha\, \mathbf{P}(\mathbf{z}_{t+1} \mid \mathbf{X}_{t+1}) \int \mathbf{P}(\mathbf{X}_{t+1} \mid \mathbf{x}_t, a_t)\, P(\mathbf{x}_t \mid \mathbf{z}_{1:t}, a_{1:t-1})\, d\mathbf{x}_t \quad (25.1)$$

This equation states that the posterior over the state variables **X** at time t + 1 is calculated recursively from the corresponding estimate one time step earlier. This calculation involves the previous action at and the current sensor measurement **z**t+1. The probability **P**(**X**t+1 | **x**t, at) is called the **transition model** or **motion model**, and **P**(**z**t+1 | **X** t+1) is the **sensor model**.

1.Localization and mapping

Localization is the problem of finding out where things are—including the robot itself.

Knowledge about where things are is at the core of any successful physical interaction with the environment.

To keep things simple, let us consider a mobile robot that moves slowly in a flat 2D world. Let us also assume the robot is given an exact map of the environment. The pose of such a mobile robot is defined by its two Cartesian coordinates with values x and y and its heading with value θ, as illustrated in Figure 25.8(a). If we arrange those three values in a vector, then any particular state is given by **X**t = θ .

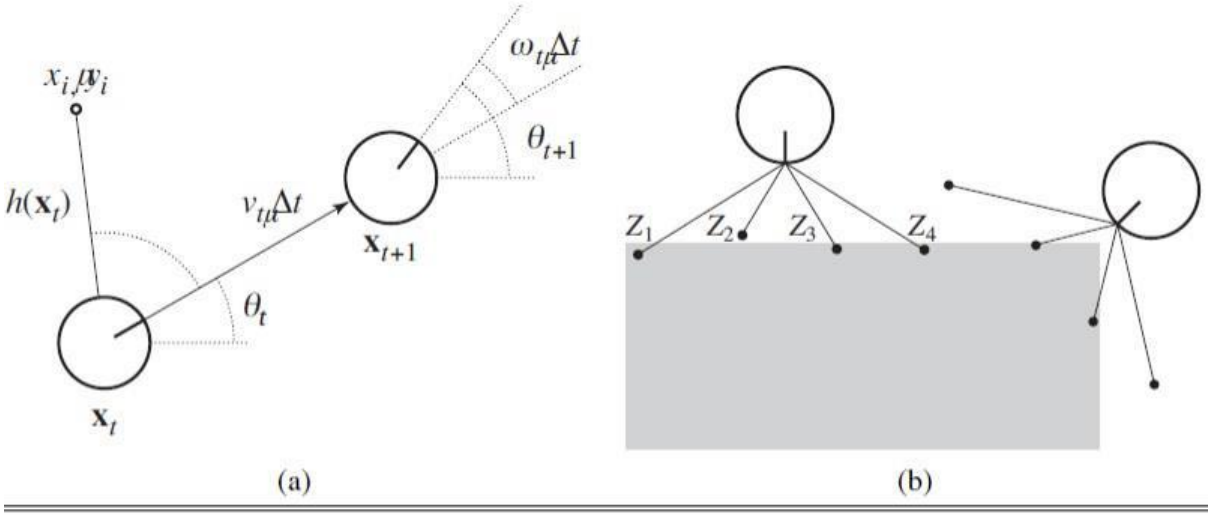


Figure 25.8 (a) A simplified kinematic model of a mobile robot. The robot is shown as a circle with an interior line marking the forward direction. The state $\mathbf{x}_t$ consists of the (x_t, y_t) position (shown implicitly) and the orientation θ_t. The new state $\mathbf{x}_{t+1}$ is obtained by an update in position of $v_t\Delta_t$ and in orientation of $\omega_t\Delta_t$. Also shown is a landmark at (x_i, y_i) observed at time t. (b) The range-scan sensor model. Two possible robot poses are shown for a given range scan (z_1, z_2, z_3, z_4). It is much more likely that the pose on the left generated the range scan than the pose on the right.

In the kinematic approximation, each action consists of the ―instantaneous‖ specification of two velocities—a translational velocity vt and a rotational velocity ωt. For small time intervals Δt, a crude deterministic model of the motion of such robots is given by

$$\hat{\mathbf{X}}_{t+1} = f(\mathbf{X}_t, \underbrace{v_t, \omega_t}_{a_t}) = \mathbf{X}_t + \begin{pmatrix} v_t\Delta t \cos\theta_t \\ v_t\Delta t \sin\theta_t \\ \omega_t\Delta t \end{pmatrix} .$$

The notation $\hat{\mathbf{X}}$ refers to a deterministic state prediction. Of course, physical robots are somewhat unpredictable. This is commonly modeled by a Gaussian distribution with mean f(**X**t, vt, ωt) and covariance **Σ**x. (See Appendix A for a mathematical definition.)

$$\mathbf{P}(\mathbf{X}t+1 \mid \mathbf{X}t, vt, \omega t) = N(\hat{\mathbf{X}}t+1, \mathbf{\Sigma}x) .$$

Next, we need a sensor model. We will consider two kinds of sensor model. The first assumes that the sensors detect *stable*, *recognizable* features of the environment called **landmarks**. The exact prediction of the observed range and bearing would be

$$\hat{\mathbf{z}}_t = h(\mathbf{x}_t) = \begin{pmatrix} \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \\ \arctan\frac{y_i - y_t}{x_i - x_t} - \theta_t \end{pmatrix}$$

Again, noise distorts our measurements. To keep things simple, one might assume Gaussian noise with covariance **Σ**z, giving us the sensor model

$$P(\mathbf{z}t \mid \mathbf{x}t) = N(\hat{}\mathbf{z}t, \mathbf{\Sigma}z) .$$

function MONTE-CARLO-LOCALIZATION(a, z, N, $P(X'|X,\ v,\ \omega)$, $P(z|z^*)$, m) **returns** a set of samples for the next time step

inputs: a, robot velocities v and ω
z, range scan $z_1, \ldots, z_M$
$P(X'|X,\ v,\ \omega)$, motion model
$P(z|z^*)$, range sensor noise model
m, 2D map of the environment

persistent: S, a vector of samples of size N

local variables: W, a vector of weights of size N
S', a temporary vector of particles of size N
W', a vector of weights of size N

if S is empty **then** /* initialization phase */
  for $i = 1$ to N **do**
    $S[i] \leftarrow$ sample from $P(X_0)$
  for $i = 1$ to N **do** /* update cycle */
    $S'[i] \leftarrow$ sample from $P(X'|X = S[i], v, \omega)$
    $W'[i] \leftarrow 1$
    for $j = 1$ to M **do**
      $z^* \leftarrow$ RAYCAST($j, X = S'[i], m$)
      $W'[i] \leftarrow W'[i] \cdot P(z_j | z^*)$
  $S \leftarrow$ WEIGHTED-SAMPLE-WITH-REPLACEMENT(N, S', W')
return S

Figure 25.9 A Monte Carlo localization algorithm using a range-scan sensor model with independent noise.

This problem is important for many robot applications, and it has been studied extensively under the name **simultaneous localization and mapping**, abbreviated as **SLAM**.

SLAM problems are solved using many different probabilistic techniques, including the extended Kalman filter

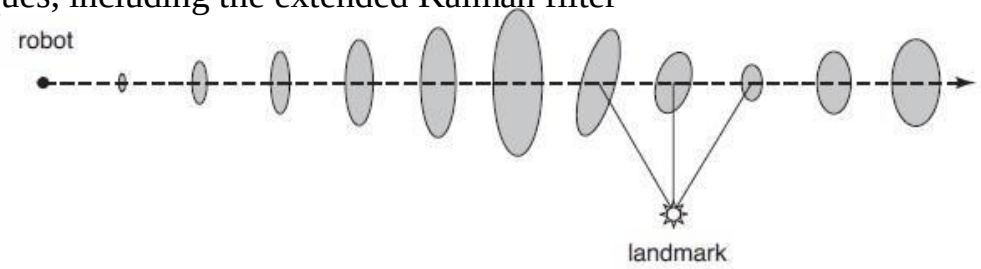


Figure 25.12 Example of localization using the extended Kalman filter. The robot moves on a straight line. As it progresses, its uncertainty increases gradually, as illustrated by the error ellipses. When it observes a landmark with known position, the uncertainty is reduced.

Expectation–maximization is also used for SLAM.

2.Other types of perception

Not all of robot perception is about localization or mapping. Robots also perceive the temperature, odors, acoustic signals, and so on. Many of these quantities can be estimated using variants of dynamic Bayes networks.

It is also possible to program a robot as a reactive agent, without explicitly reasoning about probability distributions over states.

3.Machine learning in robot perception

Machine learning plays an important role in robot perception. This is particularly the case when the best internal representation is not known. One common approach is to map high dimensional sensor streams into lower-dimensional spaces using unsupervised machine learning method. Such an approach is called **low-dimensional embedding**.

Methods that make robots collect their own training data are called **Self Supervised**.

In this instance, the robot uses machine learning to leverage a short-range sensor that works well for terrain classification into a sensor that can see much farther. That allows the robot to drive faster, slowing down only when the sensor model says there is a change in the terrain that needs to be examined more carefully by the short-range sensors.

PLANNING TO MOVE:

All of a robot's deliberations ultimately come down to deciding how to move effectors. The **point-to-point motion** problem is to deliver the robot or its end effector to a designated target location. A greater challenge is the **compliant motion** problem, in which a robot moves while being in physical contact with an obstacle.

There are two main approaches: **cell decomposition** and **skeletonization**. Each reduces the continuous path-planning problem to a discrete graph-search problem.

1 Configuration space

We will start with a simple representation for a simple robot motion problem. It has two joints that move independently. the robot's configuration can be described by a four dimensional coordinate: (x_e, y_e) for the location of the elbow relative to the environment and (x_g, y_g) for the location of the gripper. They constitute what is known as **workspace representation**.

The problem with the workspace representation is that not all workspace coordinates are actually attainable, even in the absence of obstacles. This is because of the **linkage constraints** on the space of attainable workspace coordinates.

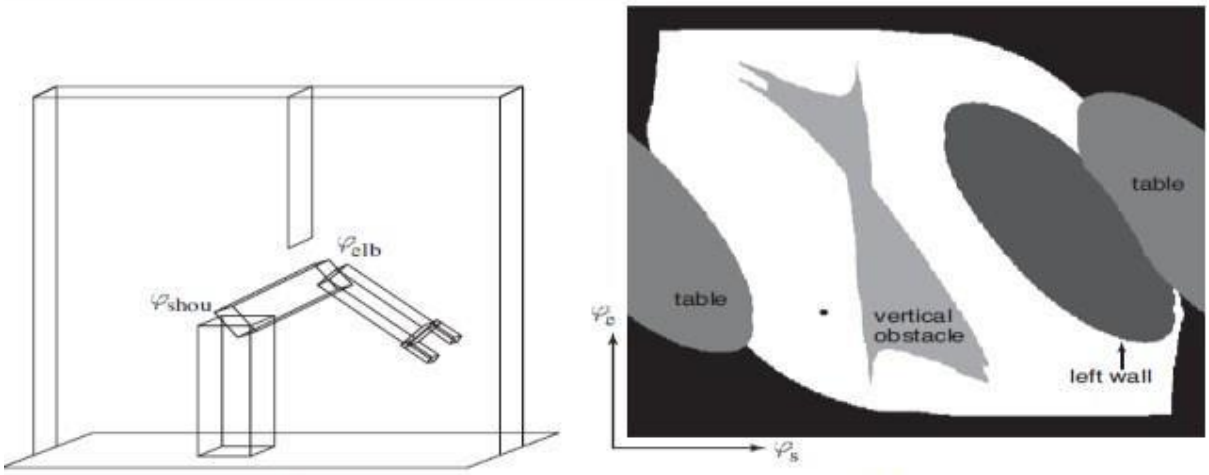


Figure 25.14 (a) Workspace representation of a robot arm with 2 DOFs. The workspace is a box with a flat obstacle hanging from the ceiling. (b) Configuration space of the same robot. Only white regions in the space are configurations that are free of collisions. The dot in this diagram corresponds to the configuration of the robot shown on the left.

Transforming configuration space coordinates into workspace coordinates is simple: it involves a series of straightforward coordinate transformations. These transformations are linear for prismatic joints and trigonometric for revolute joints. This chain of coordinate transformation is known as **kinematics**.

The inverse problem of calculating the configuration of a robot whose effector location is specified in workspace coordinates is known as **inverse kinematics.**

2 Cell decomposition methods

The first approach to path planning uses **cell decomposition**—that is, it decomposes the free space into a finite number of contiguous regions, called cells.

A decomposition has the advantage that it is extremely simple to implement, but it also suffers from three limitations. First, it is workable only for low-dimensional configuration spaces, Second, there is the problem of what to do with cells that are ―mixed‖, And third, any path through a discretized state space will not be smooth.

Cell decomposition methods can be improved in a number of ways, to alleviate some of these problems. The first approach allows *further subdivision* of the mixed cells—perhaps using cells of half the original size. A second way to obtain a complete algorithm is to insist on an **exact cell decomposition** of the free space.

3 Modified cost functions:

This problem can be solved by introducing a **potential field**. A potential field is a function defined over state space, whose value grows with the distance to the closest obstacle. The potential field can be used as an additional cost term in the shortest-path calculation. This induces an interesting trade off. On the one hand, the robot seeks to minimize path length to the goal. On the other hand, it

tries to stay away from obstacles by virtue of minimizing the potential function. Clearly, the resulting path is longer, but it is also safer.

There exist many other ways to modify the cost function. However, it is often easy to smooth the resulting trajectory after planning, using conjugate gradient methods. Such post-planning smoothing is essential in many real world applications.

4 Skeletonization methods

The second major family of path-planning algorithms is based on the idea of **skeletonization**.

These algorithms reduce the robot's free space to a one-dimensional representation, for which the planning problem is easier. This lower-dimensional representation is called a **skeleton** of the configuration space.

Voronoi graph of the free space—the set of all points that are equidistant to two or more obstacles. To do path planning with a Voronoi graph, the robot first changes its present configuration to a point on the Voronoi graph. It is easy to show that this can always be achieved by a straight-line motion in configuration space. Second, the robot follows the Voronoi graph until it reaches the point nearest to the target configuration. Finally, the robot leaves the Voronoi graph and moves to the target. Again, this final step involves straight-line motion in configuration space.

An alternative to the Voronoi graphs is the **probabilistic roadmap**, a skeletonization approach that offers more possible routes, and thus deals better with wide-open spaces. With these improvements, probabilistic roadmap planning tends to scale better to high-dimensional configuration spaces than most alternative path-planning techniques.

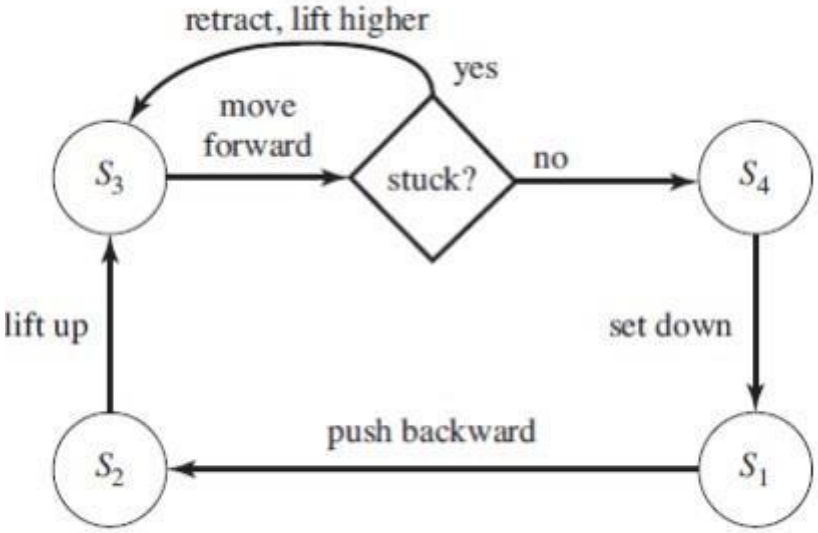
ROBOTIC SOFTWARE ARCHITECTURE:

A methodology for structuring algorithms is called a **software architecture**. An architecture includes languages and tools for writing programs, as well as an overall philosophy for how programs can be brought together. Architectures that combine reactive and deliberate techniques are called **hybrid architectures**.

1 Subsumption architecture

The **subsumption architecture** is a framework for assembling reactive controllers out of finite state machines. Nodes in these machines may contain tests for certain sensor variables, in which case the execution trace of a finite state machine is conditioned on the outcome of such a test. The resulting machines are refereed to as **augmented finite state machines**, or AFSMs, where the augmentation refers to the use of clocks.

An example of a simple AFSM is the four-state machine shown in BELOW Figure, which generates cyclic leg motion for a hexapod walker.



(b)

In our example, we might begin with AFSMs for individual legs, followed by an AFSM for coordinating multiple legs. On top of this, we might implement higher-level behaviors such as collision avoidance, which might involve backing up and turning.

Unfortunately, the subsumption architecture has its own problems. First, the AFSMs are driven by raw sensor input, an arrangement that works if the sensor data is reliable and contains all necessary information for decision making, but fails if sensor data has to be integrated in nontrivial ways over time. A subsumption style robot usually does just one task, and it has no notion of how to modify its controls to accommodate different goals. Finally, subsumption style controllers tend to be difficult to understand.

However, it has had an influence on other architectures, and on individual components of some architectures.

2 Three-layer architecture

Hybrid architectures combine reaction with deliberation. The most popular hybrid architecture is the **three-layer architecture**, which consists of a reactive layer, an executive layer, and a deliberative layer.

The **reactive layer** provides low-level control to the robot. It is characterized by a tight sensor–action loop. Its decision cycle is often on the order of milliseconds.

The **executive layer** (or sequencing layer) serves as the glue between the reactive layer and the deliberative layer. It accepts directives by the deliberative layer, and sequences them for the reactive layer.

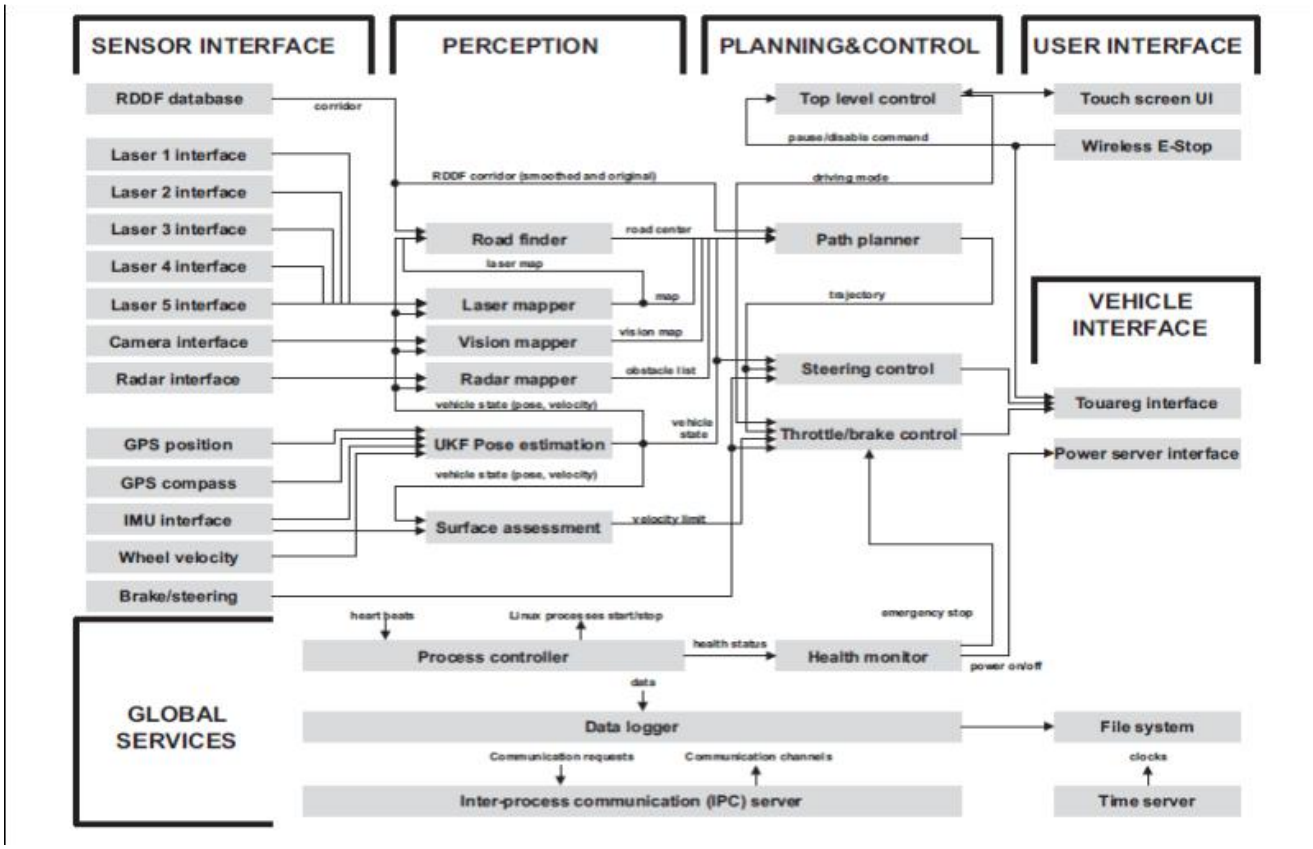
The **deliberative layer** generates global solutions to complex tasks using planning.

Because of the computational complexity involved in generating such solutions, its decision cycle is often in the order of minutes. The deliberative layer (or planning layer) uses models for decision making.

3 Pipeline architecture

Another architecture for robots is known as the **pipeline architecture**. Just like the subsumption architecture, the pipeline architecture executes multiple process in parallel.

Data enters this pipeline at the **sensor interface layer**. The **perception layer** then updates the robot's internal models of the environment based on this data. Next, these models are handed to the **planning and control layer**. Those are then communicated back to the vehicle through the **vehicle interface layer**.



The key to the pipeline architecture is that this all happens in parallel. While the perception layer processes the most recent sensor data, the control layer bases its choices on slightly older data. In this way, the pipeline architecture is similar to the human brain. We don't switch off our motion controllers when we digest new sensor data. Instead, we perceive, plan, and act all at the same time. Processes in the pipeline architecture run asynchronously, and all computation is data-driven. The resulting system is robust, and it is fast.

APPLICTION DOMAINS:

Industry and Agriculture. Traditionally, robots have been fielded in areas that require difficult human labour, yet are structured enough to be amenable to robotic automation. The best example is the assembly line, where manipulators routinely perform tasks such as assembly, part placement, material handling, welding, and painting. In many of these tasks, robots have become more cost-effective than human workers.

Transportation. Robotic transportation has many facets: from autonomous helicopters that deliver payloads to hard-to-reach locations, to automatic wheelchairs that transport people who are unable to control wheelchairs by themselves, to autonomous straddle carriers that outperform skilled human drivers when transporting containers from ships to trucks on loading docks.

Robotic cars. Most of use cars every day. Many of us make cell phone calls while
driving. Some of us even text. The sad result: more than a million people die every year in traffic accidents. Robotic cars like BOSS and STANLEY offer hope: Not only will they make driving much safer, but they will also free us from the need to pay attention to the road during our daily commute.

Health care. Robots are increasingly used to assist surgeons with instrument placement when operating on organs as intricate as brains, eyes, and hearts. Robots have become indispensable tools in a range of surgical procedures, such as hip replacements, thanks to their high precision. In pilot studies, robotic devices have been found to reduce the danger of lesions when performing colonoscopy.

Hazardous environments. Robots have assisted people in cleaning up nuclear waste, most notably in Chernobyl and Three Mile Island. Robots were present after the collapse of the World Trade Center, where they entered structures deemed too dangerous for human search and rescue crews.

Exploration. Robots have gone where no one has gone before, including the surface of Mars. Robotic arms assist astronauts in deploying and retrieving satellites and in building the International Space Station. Robots also help explore under the sea. They are routinely used to acquire maps of sunken ships.

Personal Services. Service is an up-and-coming application domain of robotics. Service robots assist individuals in performing daily tasks. Commercially available domestic service robots include autonomous vacuum cleaners, lawn mowers, and golf caddies. example for robot vaccum cleaner is ROOMBA.

Entertainment. Robots have begun to conquer the entertainment and toy industry. we see **robotic soccer**, a competitive game very much like human soccer, but played with autonomous mobile robots. Robot soccer provides great opportunities for research in AI, since it raises a range of problems relevant to many other, more serious robot applications.

Human augmentation. A final application domain of robotic technology is that of human augmentation. Researchers have developed legged walking machines that can carry people around, very much like a wheelchair. Several research efforts presently focus on the development of devices that make it easier for people to walk or move their arms by providing additional forces through extra skeletal attachments.

WEAK AI: CAN MACHINES ACT INTELLIGENTLY?

AI is impossible depends on how it is defined. we defined AI as the quest for the best agent program on a given architecture. With this formulation, AI is by definition possible: for any digital architecture with k bits of program storage there are exactly 2k agent programs, and all we have to do to find the best one is enumerate and test them all. This might not be feasible for large *k*, but philosophers deal with the theoretical, not the practical.

Our definition of AI works well for the engineering problem of finding a good agent, given an

architecture. Therefore, we're tempted to end this section right now, answering the title question in the affirmative. But philosophers are interested in the problem of compar- ing two architectures—human and machine. Furthermore, they have traditionally posed the question not in terms of maximizing expected utility but rather as, "**Can machines think**?"

Alan Turing, in his famous paper "Computing Machinery and Intelligence" (1950), sug- gested that instead of asking whether machines can think, we should ask whether machines can pass a behavioral intelligence test, which has come to be called the **Turing Test**. The test is for a program to have a conversation (via online typed messages) with an interrogator for five minutes. The interrogator then has to guess if the conversation is with a program or a person; the program passes the test if it fools the interrogator 30% of the time.

The argument from disability

The "argument from disability" makes the claim that "a machine can never do *X*." As exam- ples of *X*, Turing lists the following:

Be kind, resourceful, beautiful, friendly, have initiative, have a sense of humor, tell right from wrong, make mistakes, fall in love, enjoy strawberries and cream, make someone fall in love with it, learn from experience, use words properly, be the subject of its own thought, have as much diversity of behavior as man, do something really new

It is clear that computers can do many things as well as or better than humans, including things that people believe require great human insight and understanding. This does not mean, of course, that computers use insight and understanding in performing these tasks those are not part of behavior, and we address such questions elsewhere but the point is that one's first guess about the mental processes required to produce a given behavior is often wrong. It is also true, of course, that there are many tasks at which computers do not yet excel (to put it mildly), including Turing's task of carrying on an open-ended conversation.

The mathematical objection

It is well known, through the work of Turing (1936) and Go¨del (1931), that certain math- ematical questions are in principle unanswerable by particular formal systems. Go¨del's in- completeness theorem is the most famous example of this. Briefly, for any formal axiomatic system F powerful enough to do arithmetic, it is possible to construct a so-called Go¨del sentence G(F) with the following properties:

- $G(F)$ is a sentence of F, but cannot be proved within F.
- If F is consistent, then $G(F)$ is true.

even if we grant that computers have limitations on what they can prove, there is no evidence that humans are immune from those limitations. It is all too easy to show rigorously that a formal system cannot do *X*, and then claim that humans *can* do *X* using their own informal method, without giving any evidence for this claim. Indeed, it is impossible to prove that humans are not subject to Go¨del's incompleteness theorem, because any rigorous proof would require a formalization of the claimed unformalizable human talent, and hence refute itself. So we are left with an appeal to intuition that humans can somehow perform superhuman feats of mathematical insight. This appeal is expressed with arguments such as "we must assume our own consistency, if thought is to be possible at all". But if anything, humans are known to be inconsistent. This is certainly true for everyday reasoning, but it is

also true for careful mathematical thought. A famous example is the four-color map problem.

The argument from informality

One of the most influential and persistent criticisms of AI as an enterprise was raised by Turing as the "argument from informality of behavior." Essentially, this is the claim that human behavior is far too complex to be captured by any simple set of rules and that because computers can do no more than follow a set of rules, they cannot generate behavior as intelligent as that of humans. The inability to capture everything in a set of logical rules is called the **qualification problem** in AI.

1. Good generalization from examples cannot be achieved without background knowledge. They claim no one has any idea how to incorporate background knowledge into the neural network learning process. In fact, that there are techniques for using prior knowledge in learning algorithms. Those techniques, however, rely on the availability of knowledge in explicit form, something that Dreyfus and Dreyfus strenuously deny. In our view, this is a good reason for a serious redesign of current models of neural processing so that they *can* take advantage of previously learned knowledge in the way that other learning algorithms do.
2. Neural network learning is a form of supervised learning, requiring the prior identification of relevant inputs and correct outputs. Therefore, they claim, it cannot operate autonomously without the help of a human trainer. In fact, learning without a teacher can be accomplished by **unsupervised learning** and **reinforcement learning** .
3. Learning algorithms do not perform well with many features, and if we pick a subset of features, "there is no known way of adding new features should the current set proveinadequate to account for the learned facts." In fact, new methods such as support vector machines handle large feature sets very well. With the introduction of large Web-based data sets, many applications in areas such as language processing (Sha and Pereira, 2003) and computer vision (Viola and Jones, 2002a) routinely handle millionsof features.
4. The brain is able to direct its sensors to seek relevant information and to process itto extract aspects relevant to the current situation. But, Dreyfus and Dreyfus claim, "Currently, no details of this mechanism are understood or even hypothesized in a waythat could guide AI research." In fact, the field of active vision, underpinned by the theory of information value , is concerned with exactly the problem of directing sensors, and already some robots have incorporated the theoretical results obtained

STRONG AI: CAN MACHINES REALLY THINK?

Many philosophers have claimed that a machine that passes the Turing Test would still not be *actually* thinking, but would be only a *simulation* of thinking. Again, the objection was foreseen by Turing. He cites a speech by Professor Geoffrey Jefferson (1949):

> Not until a machine could write a sonnet or compose a concerto because of thoughts and emotions felt, and not by the chance fall of symbols, could we agree that machine equals brain—that is, not only write it but know that it had written it.

Turing calls this the argument from **consciousness**—the machine has to be aware of its ownmental states and actions. While consciousness is an important subject, Jefferson's key point actually relates to **phenomenology**, or the study of direct experience: the machine has to actually feel emotions. Others focus on **intentionality**—that is, the question of whether the machine's purported beliefs, desires, and other representations are actually "about" some- thing in the real world.

Turing argues that Jefferson would be willing to extend the polite convention to ma- chines if only he had experience with ones that act intelligently. He cites the following dialog, which has become such a part of AI's oral tradition that we simply have to include it:

HUMAN: In the first line of your sonnet which reads "shall I compare thee to a summer's day," would not a "spring day" do as well or better?
MACHINE: It wouldn't scan.
HUMAN: How about "a winter's day." That would scan all right.
MACHINE: Yes, but nobody wants to be compared to a winter's day.
HUMAN: Would you say Mr. Pickwick reminded you of Christmas?
MACHINE: In a way.
HUMAN: Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.
MACHINE: I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

Mental states and the brain in a vat

Physicalist philosophers have attempted to explicate what it means to say that a person—and, by extension, a computer—is in a particular mental state. They have focused in particular on **intentional states**. These are states, such as believing, knowing, desiring, fearing, and so on, that refer to some aspect of the external world. For example, the knowledge that one is eating a hamburger is a belief *about* the hamburger and what is happening to it.

If physicalism is correct, it must be the case that the proper description of a person's mental state is *determined* by that person's brain state. Thus, if I am currently focused on eating a hamburger in a mindful way, my instantaneous brain state is an instance of the class of mental states "knowing that one is eating a hamburger." Of course, the specific configurations of all the atoms of my brain are not essential: there are many configurations of my brain, or of other people's brain, that would belong to the same class of mental states. The key point is that the same brain state could not correspond to a fundamentally distinct mental state, such as the knowledge that one is eating a banana.

The "**wide content**" view interprets it from the point of view of an omniscient outside observer with access to the whole situation, who can distinguish differences in the world. Under this view, the content of mental states involves both the brain state and the environment history. **Narrow content**, on the other hand, considers only the brain state. The narrow content of the brain states of a real hamburger-eater and a brain-in-a-vat "hamburger"-"eater" is the same in both cases.

Functionalism and the brain replacement experiment

The theory of **functionalism** says that a mental state is any intermediate causal condition between input and output. Under functionalist theory, any two systems with isomorphic causal processes would have the same mental states. Therefore, a computer program could have the same mental states as a person. Of course, we have not yet said what "isomorphic" really means, but the assumption is that there is some level of abstraction below which the specific implementation does not matter.

And this explanation must also apply to the real brain, which has the same functional properties. There are three possible conclusions:

1. The causal mechanisms of consciousness that generate these kinds of outputs in normalbrains are still operating in the electronic version, which is therefore conscious.
2. The conscious mental events in the normal brain have no causal connection to behavior, and are missing from the electronic brain, which is therefore not conscious.
3. The experiment is impossible, and therefore speculation about it is meaningless.

Biological naturalism and the Chinese Room

A strong challenge to functionalism has been mounted by John Searle's (1980) biological naturalism, according to which mental states are high-level emergent features that are caused by low-level physical processes in the neurons, and it is the (unspecified) properties of the neurons that matter. Thus, mental states cannot be duplicated just on the basis of some pro- gram having the same functional structure with the same input–output behavior; we would require that the program be running on an architecture with the same causal power as neurons. To support his view, Searle describes a hypothetical system that is clearly running a program and passes the Turing Test, but that equally clearly (according to Searle) does not understand anything of its inputs and outputs. His conclusion is that running the appropriate program (i.e., having the right outputs) is not a sufficient condition for being a mind.

So far, so good. But from the outside, we see a system that is taking input in the form of Chinese sentences and generating answers in Chinese that are as "intelligent" as those in the conversation imagined by Turing.[4] Searle then argues: the person in the room doesnot understand Chinese (given). The rule book and the stacks of paper, being just pieces ofpaper, do not understand Chinese. Therefore, there is no understanding of Chinese. Hence, according to Searle, running the right program does not necessarily generate understanding.

The real claim made by Searle rests upon the following four axioms :

1. Computer programs are formal (syntactic).
2. Human minds have mental contents (semantics).
3. Syntax by itself is neither constitutive of nor sufficient for semantics.
4. Brains cause minds.

From the first three axioms Searle concludes that programs are not sufficient for minds. In other words, an agent running a program *might* be a mind, but it is not *necessarily* a mind justby virtue of running the program. From the fourth axiom he concludes "Any other system capable of causing minds would have to have causal powers (at least) equivalent to thoseof brains." From there he infers that any artificial brain would have to duplicate the causal powers of brains, not just run a particular program, and that human brains do not produce mental phenomena solely by virtue of running a program.

Consciousness, qualia, and the explanatory gap

Running through all the debates about strong AI—the elephant in the debating room, soto speak—is the issue of **consciousness**. Consciousness is often broken down into aspects such as understanding and self-awareness. The aspect we will focus on is that of *subjective experience*: why it is that it *feels* like something to have certain brain states (e.g., while eatinga hamburger), whereas it presumably does not feel like anything to have other physical states(e.g., while being a rock). The technical term for the intrinsic nature of experiences is **qualia**(from the Latin word meaning, roughly, "such things").

Qualia present a challenge for functionalist accounts of the mind because different qualia could be involved in what are otherwise isomorphic causal processes. Consider, for example, the **inverted spectrum**

thought experiment, which the subjective experience of per- son *X* when seeing red objects is the same experience that the rest of us experience when seeing green objects, and vice versa.

This **explanatory gap** has led some philosophers to conclude that humans are simply incapable of forming a proper understanding of their own consciousness. Others, notably Daniel Dennett (1991), avoid the gap by denying the existence of qualia,attributing them to a philosophical confusion.

THE ETHICS AND RISKS OF DEVELOPING ARTIFICIAL INTELLIGENCE

So far, we have concentrated on whether we can develop AI, but we must also consider whether we should. If the effects of AI technology are more likely to be negative than positive, then it would be the moral responsibility of workers in the field to redirect their research. Many new technologies have had unintended negative side effects: nuclear fission brought Chernobyl and the threat of global destruction; the internal combustion engine brought air pollution, global warming, and the paving-over of paradise. In a sense, automobiles are robots that have conquered the world by making themselves indispensable.

AI, however, seems to pose some fresh problems beyond that of, say, building bridges that don't fall down:

- People might lose their jobs to automation.
- People might have too much (or too little) leisure time.
- People might lose their sense of being unique.
- AI systems might be used toward undesirable ends.
- The use of AI systems might result in a loss of accountability.
- The success of AI might mean the end of the human race.

People might lose their jobs to automation. The modern industrial economy has be come dependent on computers in general, and select AI programs in particular. For example, much of the economy, especially in the United States, depends on the availability of con- sumer credit. Credit card applications, charge approvals, and fraud detection are now done by AI programs. One could say that thousands of workers have been displaced by these AI programs, but in fact if you took away the AI programs these jobs would not exist, because human labor would add an unacceptable cost to the transactions.

People might lose their sense of being unique. In *Computer Power and Human Rea- son*, Weizenbaum (1976), the author of the ELIZA program, points out some of the potential threats that AI poses to society. One of Weizenbaum's principal arguments is that AI research makes possible the idea that humans are automata—an idea that results in a loss of autonomyor even of humanity.

AI systems might be used toward undesirable ends. Advanced technologies have often been used by the powerful to suppress their rivals. As the number theorist G. H. Hardy wrote (Hardy, 1940), "A science is said to be useful if its development tends to accentuate the existing inequalities in the distribution of wealth, or more directly promotes the destruction of human life." This holds for all sciences, AI being no exception. Autonomous AI systems are now commonplace on the battlefield; the U.S. military deployed over 5,000 autonomousaircraft and 12,000 autonomous ground vehicles in Iraq (Singer, 2009).

The use of AI systems might result in a loss of accountability. In the litigious atmo-sphere that prevails in the United States, legal liability becomes an important issue. When aphysician relies on the judgment of a medical expert system for a diagnosis, who is at fault ifthe diagnosis is wrong? Fortunately, due in part to the growing influence of decision-theoreticmethods in medicine, it is now accepted that negligence cannot

be shown if the physician performs medical procedures that have high *expected* utility, even if the *actual* result is catastrophic for the patient.

The success of AI might mean the end of the human race. Almost any technology has the potential to cause harm in the wrong hands, but with AI and robotics, we have the new problem that the wrong hands might belong to the technology itself. Countless science fiction stories have warned about robots or robot–human cyborgs running amok.

If ultra intelligent machines are a possibility, we humans would do well to make sure that we design their predecessors in such a way that they design themselves to treat us well. Science fiction writer Isaac Asimov (1942) was the first to address this issue, with his three laws of robotics:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey orders given to it by human beings, except where such orders would conflict with the First Law.

A robot must protect its own existence as long as such protection does not conflict with the First or Second Law

AGENT COMPONENTS

Interaction with the environment through sensors and actuators: For much of the history of AI, this has been a glaring weak point. With a few honorable exceptions, AI sys- tems were built in such a way that humans had to supply the inputs and interpret the outputs,

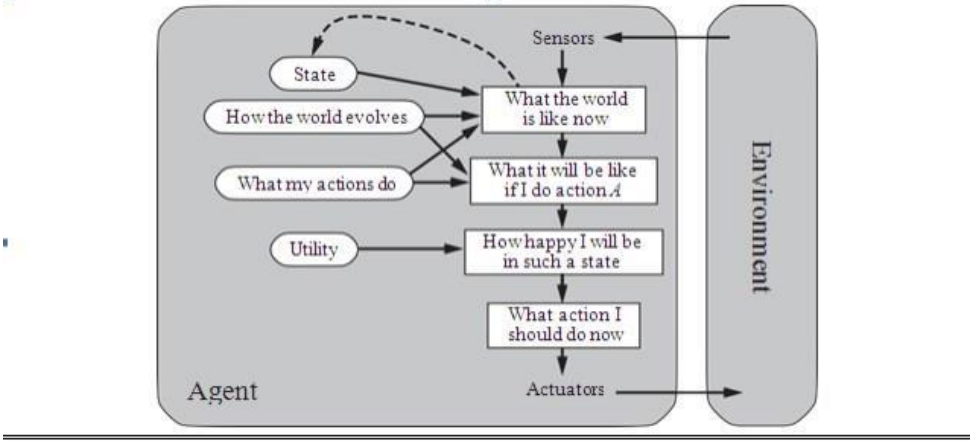


Figure A model-based, utility-based agent

while robotic systems focused on low-level tasks in which high-level reasoning and plan- ning were largely absent. This was due in part to the great expense and engineering effort required to get real robots to work at all. The situation has changed rapidly in recent years with the availability of ready-made programmable robots. These, in turn, have benefited from small, cheap, high-resolution CCD cameras and compact, reliable motor drives. MEMS (micro-electromechanical systems) technology has supplied miniaturized accelerometers, gy- roscopes, and actuators for an artificial flying insect (Floreano *et al.*, 2009). It may also be possible to combine millions of MEMS devices to produce powerful macroscopic actuators.

Keeping track of the state of the world: This is one of the core capabilities required for an intelligent agent. It requires both perception and updating of internal representations. showed how to keep track of atomic state representations, described how to do it for factored (propositional) state representations extended this to first-order logic; and Chapter 15 described **filtering** algorithms for probabilistic reasoning

in uncertain environments. Current filtering and perception algorithms can be combined to do a reasonable job of reporting low-level predicates such as "the cup is on the table." Detecting higher-level actions, such as "Dr. Russell is having a cup of tea with Dr. Norvig while dis- cussing plans for next week," is more difficult. Currently it can be done only with the help of annotated examples.

Projecting, evaluating, and selecting future courses of action: The basic knowledge- representation requirements here are the same as for keeping track of the world; the primary difficulty is coping with courses of action—such as having a conversation or a cup of tea—that consist eventually of thousands or millions of primitive steps for a real agent. It is only by imposing **hierarchical structure** on behavior that we humans cope at all.how to use hierarchical representations to handle problems of this scale; fur- ther more, work in **hierarchical reinforcement learning** has succeeded in combining some of these ideas with the techniques for decision making under uncertainty described in. As yet, algorithms for the partially observable case (POMDPs) are using the same atomic state representation we used for the search algorithms

It has proven very difficult to decompose preferences over complex states in the same way that Bayes nets decompose beliefs over complex states. One reason may be that preferences over states are really *compiled* from preferences over state histories, which are described by **reward functions**

Learning: Chapters 18 to 21 described how learning in an agent can be formulated as inductive learning (supervised, unsupervised, or reinforcement-based) of the functions that constitute the various components of the agent. Very powerful logical and statistical tech- niques have been developed that can cope with quite large problems, reaching or exceeding human capabilities in many tasks—as long as we are dealing with a predefined vocabulary of features and concepts.

AGENT ARCHITECTURES

It is natural to ask, "Which of the agent architectures should an agent use?" The answer is, "All of them!" We have seen that reflex responses are needed for situations in which time is of the essence, whereas knowledge-based deliberation allows the agent to plan ahead. A complete agent must be able to do both, using a **hybrid architecture**. One important property of hybrid architectures is that the boundaries between different decision components are not fixed. For example, **compilation** continually converts declarative in- formation at the deliberative level into more efficient representations, eventually reaching the reflex level.

For example, a taxi-driving agent that sees an accident ahead must decide in a split second either to brake or to take evasive action. It should also spend that split second thinking about the most important questions, such as whether the lanes to the left and right are clear and whether there is a large truck close behind, rather than worrying about wear and tear on the tires or where to pick up the next passenger. These issues are usually studied under the heading of **real-time AI**

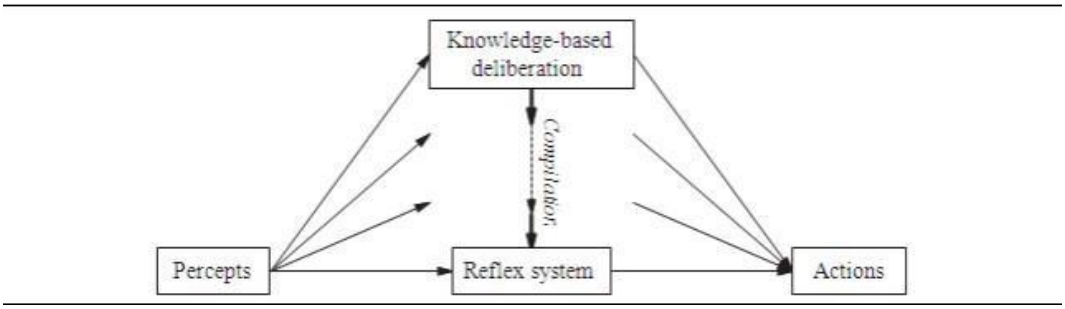


Fig: Compilation serves to convert deliberative decision making into more efficient, reflexive mechanisms. Clearly, there is a pressing need for *general* methods of controlling deliberation, ratherthan specific recipes for what to think about in each situation. The first useful idea is to employ **anytime algorithms**

The second technique for controlling deliberation is decision-theoretic meta reasoning (Russell and Wefald, 1989, 1991; Horvitz, 1989; Horvitz and Breese, 1996). This method applies the theory of information value to the selection of individual computations. The value of a computation depends on both its cost (in terms of delaying action) andits benefits (in terms of improved decision quality). Meta reasoning techniques can be used to design better search algorithms and to guarantee that the algorithms have the anytime property. Meta reasoning is expensive, of course, and compilation methods can be applied so that the overhead is small compared to the costs of the computations being controlled. Meta level reinforcement learning may provide another way to acquire effective policies for controllingdeliberation

Meta reasoning is one specific example of a **reflective architecture**—that is, an architecture that enables deliberation about the computational entities and actions occurring withinthe architecture itself. A theoretical foundation for reflective architectures can be built by defining a joint state space composed from the environment state and the computational stateof the agent itself.

ARE WE GOING IN THE RIGHT DIRECTION?

The preceding section listed many advances and many opportunities for further progress. Butwhere is this all leading? Dreyfus (1992) gives the analogy of trying to get to the moon by climbing a tree; one can report steady progress, all the way to the top of the tree. In this section, we consider whether AI's current path is more like a tree climb or a rocket trip.

Perfect rationality. A perfectly rational agent acts at every instant in such a way as to maximize its expected utility, given the information it has acquired from the environment. We have seen that the calculations necessary to achieve perfect rationality in most environments are too time consuming, so perfect rationality is not a realistic goal.

Calculative rationality. This is the notion of rationality that we have used implicitly in de- signing logical and decision-theoretic agents, and most of theoretical AI research has focusedon this property. A calculatively rational agent *eventually* returns what *would have been* therational choice at the beginning of its deliberation. This is an interesting property for a system to exhibit, but in most environments, the right answer at the wrong time is of no value. In practice, AI system designers are forced to compromise on decision quality to obtain reason- able overall performance; unfortunately, the theoretical basis of calculative rationality does not provide a well-founded way to make such compromises.

Bounded rationality. Herbert Simon (1957) rejected the notion of perfect (or even approximately perfect) rationality and replaced it with bounded rationality, a descriptive theory of decision making by real agents.
Bounded optimality (BO). A bounded optimal agent behaves as well as possible, *given its computational resources*. That is, the expected utility of the agent program for a bounded optimal agent is at least as high as the expected utility of any other agent program running onthe same machine.

WHAT IF AI DOES SUCCEED?

In David Lodge's *Small World* (1984), a novel about the academic world of literary criticism, the protagonist causes consternation by asking a panel of eminent but contradictory literary theorists the following question: "What if you were right?" None of the theorists seems to have considered this question before, perhaps because debating unfalsifiable theories is an end in itself. Similar confusion can be evoked by asking AI researchers, "*What if you succeed?*"

We can expect that medium-level successes in AI would affect all kinds of people in their daily lives. So far, computerized communication networks, such as cell phones and the Internet, have had this kind of pervasive effect on society, but AI has not. AI has been at workbehind the scenes—for example, in automatically approving or denying credit card transactions for every purchase made on the Web—but has not been visible to the average consumer.We can imagine that truly useful personal assistants for the office or the home would have a large positive impact on people's lives, although they might cause some economic dislocation in the short term. Automated assistants for driving could prevent accidents, saving tens of thousands of lives per year. A technological capability at this level might also be applied to the development of autonomous weapons, which many view as undesirable. Some of the biggest societal problems we face today—such as the harnessing of genomic information fortreating disease, the efficient management of energy resources, and the verification of treaties concerning nuclear weapons—are being addressed with the help of AI technologies.

Finally, it seems likely that a large-scale success in AI—the creation of human-level in-telligence and beyond—would change the lives of a majority of humankind. The very natureof our work and play would be altered, as would our view of intelligence, consciousness, andthe future destiny of the human race. AI systems at this level of capability could threaten hu-man autonomy, freedom, and even survival. For these reasons, we cannot divorce AI researchfrom its ethical consequences

In conclusion, we see that AI has made great progress in its short history, but the final sentence of Alan Turing's (1950) essay on *Computing Machinery and Intelligence* is still valid today:

We can see only a short distance ahead, but we can see that much remains to be done.

1

2

3

4