Module 5

	Planning and Learning	12	CO4
5.1	The planning problem, Planning Vs Searching, STRIPS and ADL, Planning with state space search, Partial order planning, Hierarchical planning, Contingent Planning		
	#Self learning : Multiagent planning		
5.2	Learning: Forms of Learning, Inductive Learning, Learning Decision Tree, applications of learning		
	#Self learning : Practical machine learning		
5.3	**Applications of AI:** Natural Language Processing(NLP):Language models, text classification, information retrieval, information extraction **Expert Systems:** Components of expert systems, ES vs Traditional System. Characteristics of expert systems, roles in ES implementation, ES implementation process, applications, advantages and limitations of ES		

PLANNING-Introduction

"How an agent can take advantage of the **structure of a problem** to construct complex plans of action"

- The task of coming up with a sequence of actions that will achieve a goal is called **PLANNING**
  - So far: the search-based problem-solving agent, logical planning agent
- Primarily scaling up to complex planning problems that defeat the approaches we have seen so far.

- Consider only environments that are fully observable, deterministic, finite, static (change happens only when the agent acts), and discrete (in time, action, objects, and effects)
- **These are called classical planning environments.**
  - In contrast, non classical planning is for partially observable or stochastic environments and involves a different set of algorithms and agent designs

THE PLANNING PROBLEM

- Let us consider what can happen when an ordinary problem-solving agent using standard search algorithms—depth-first, A $*$, and so on—comes up against large, real-world problems.
  - That will help us design better planning agents.

Problem-solving Agent Can Be Overwhelmed By Irrelevant Actions

- Consider the task of buying a copy of AI: A Modern Approach from an online bookseller. Suppose there is one buying action for each 10-digit ISBN number, for a total of 10 billion actions
- Goal→own a copy of ISBN 0137903952.
- One buying action for each 10-digit ISBN number, for a total of 10 billion actions.
- A sensible planning agent-explicit goal description such as Have(ISBN 0137903952) and generate the action Buy(ISBN 0137903952) directly
- A general knowledge that Buy(x) results in Have(x).

GOOD HEURISTIC FUNCTION ?

- Suppose the agent’s goal is to buy four different books online. Then there will be 10^40 plans of just four steps.
- Obvious - number of books that remain to be bought
- Not obvious to a problem-solving agent, because it sees the goal test only as a black box that returns true or false for each state.

PROBLEM DECOMPOSITION

- Problem solver might be inefficient because it cannot take advantage of problem decomposition.
  - Consider the problem of delivering a set of overnight packages to their respective destinations, which are scattered across Australia. It makes sense to find out the nearest airport for each destination and divide the overall problem into several sub problems, one for each airport. Within the set of packages routed through a given airport, whether further decomposition is possible depends on the destination city

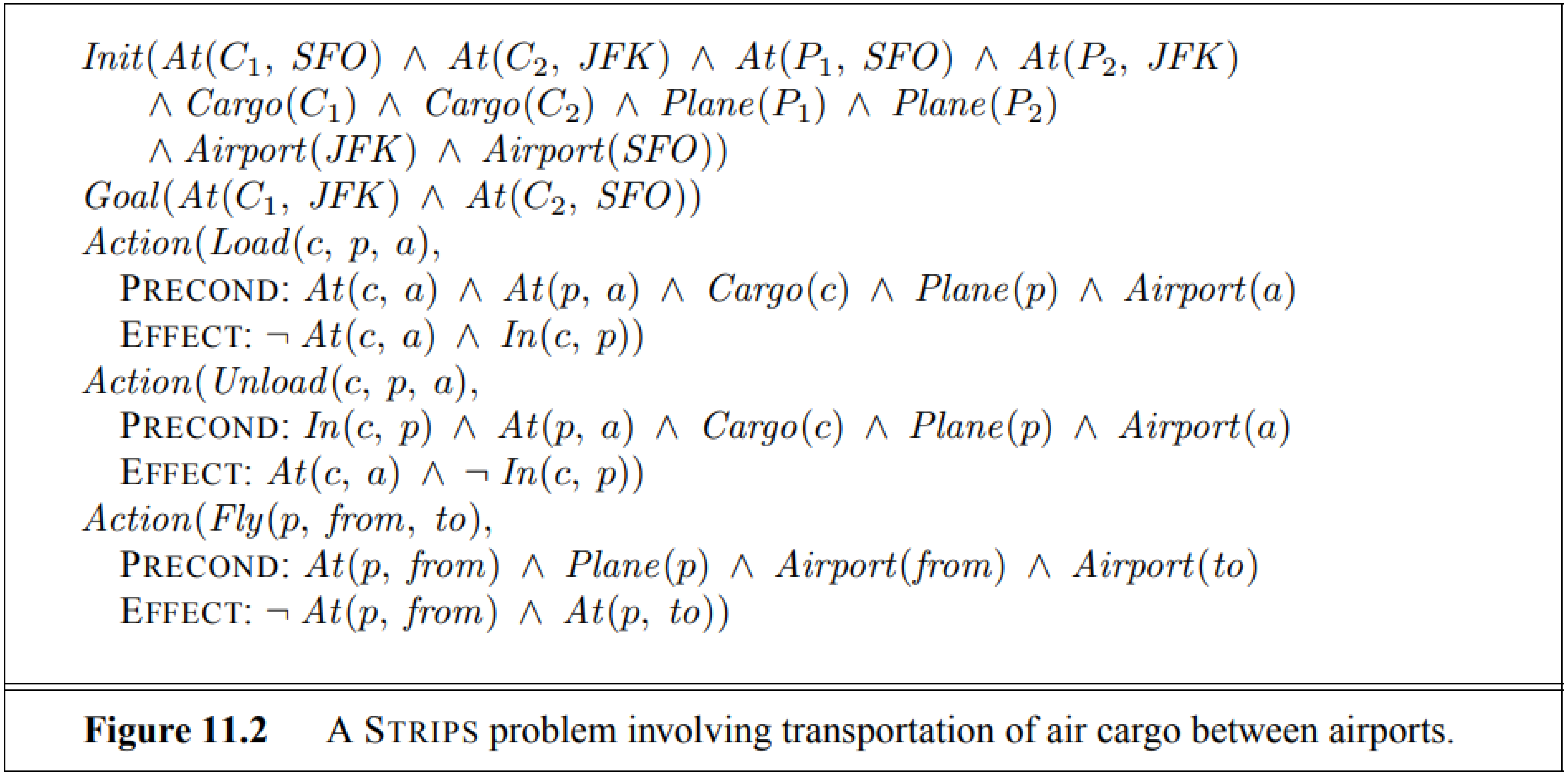
- Assumption that most real-world problems are nearly decomposable.
- NEARLY DECOMPOSABLE - Planner can work on sub goals independently, but might need to do some additional work to combine the resulting sub plan
- Key is to find a language that is expressive enough to describe a wide variety of problems, but restrictive enough to allow efficient algorithms to operate over it.
- Basic representation language of classical planners, known as the STRIPS language- STanford Research Institute Problem Solver.

STRIPS language

- Representation of states
- Representation of goals
- Representation of actions
  - action schema consists of three parts:
    - Action name and parameter
    - PRECONDITION
    - EFFECT

    Fly action could be written as

$$
\begin{array}{l}
Action(Fly(p : Plane, from : Airport, to : Airport), \\
\quad \textsc{Precond}{:}At(p, from) \land (from \neq to) \\
\quad \textsc{Effect}{:}\neg At(p, from) \land At(p, to))\ .
\end{array}
$$



$Init(At(C_1, SFO) \land At(C_2, JFK) \land At(P_1, SFO) \land At(P_2, JFK)$
$\quad \land\ Cargo(C_1) \land Cargo(C_2) \land Plane(P_1) \land Plane(P_2)$
$\quad \land\ Airport(JFK) \land Airport(SFO))$
$Goal(At(C_1, JFK) \land At(C_2, SFO))$
$Action(Load(c, p, a),$
$\quad$ PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
$\quad$ EFFECT: $\neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a),$
$\quad$ PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
$\quad$ EFFECT: $At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to),$
$\quad$ PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
$\quad$ EFFECT: $\neg At(p, from) \land At(p, to))$

Figure 11.2 A STRIPS problem involving transportation of air cargo between airports.

- Following plan is a solution to the problem:

$$[Load(C_1, P_1, SFO), Fly(P_1, SFO, JFK),$$
$$Load(C_2, P_2, JFK), Fly(P_2, JFK, SFO)]\ .$$

Example: The spare tire problem

- Consider the problem of changing a flat tire. More precisely, the goal is to have a good spare tire properly mounted onto the car’s axle, where the initial state has a flat tire on the axle and a good spare tire in the trunk. To keep it simple, our version of the problem is a very abstract one, with no sticky lug nuts or other complications. There are just four actions: removing the spare from the trunk, removing the flat tire from the axle, putting the spare on the axle, and leaving the car unattended overnight. We assume that the car is in a particularly bad neighborhood, so that the effect of leaving it overnight is that the tires disappear

$Init(At(Flat, Axle) \land At(Spare, Trunk))$
$Goal(At(Spare, Axle))$
$Action(Remove(Spare, Trunk),$
  PRECOND: $At(Spare, Trunk)$
  EFFECT: $\neg At(Spare, Trunk) \land At(Spare, Ground))$
$Action(Remove(Flat, Axle),$
  PRECOND: $At(Flat, Axle)$
  EFFECT: $\neg At(Flat, Axle) \land At(Flat, Ground))$
$Action(PutOn(Spare, Axle),$
  PRECOND: $At(Spare, Ground) \land \neg At(Flat, Axle)$
  EFFECT: $\neg At(Spare, Ground) \land At(Spare, Axle))$
$Action(LeaveOvernight,$
  PRECOND:
  EFFECT: $\neg At(Spare, Ground) \land \neg At(Spare, Axle) \land \neg At(Spare, Trunk)$
    $\land \neg At(Flat, Ground) \land \neg At(Flat, Axle))$

Figure 11.3 The simple spare tire problem.

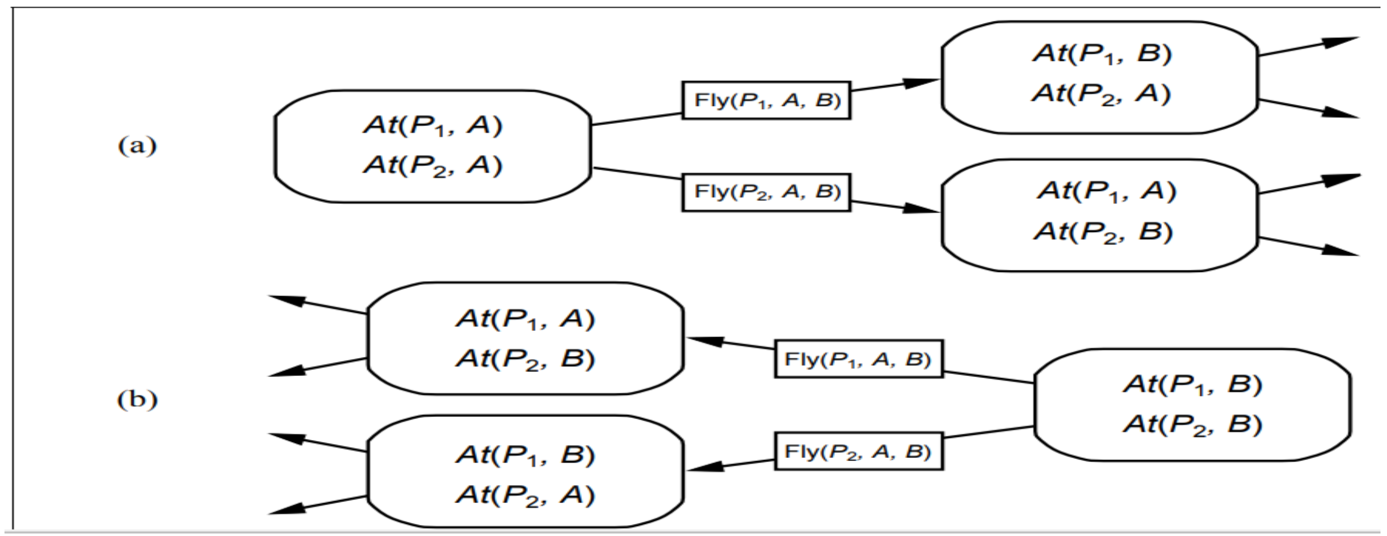
Example: The blocks world

One of the most famous planning domains is known as the blocks world. This domain consists of a set of cube-shaped blocks sitting on a table. The blocks can be stacked, but only one block can fit directly on top of another. A robot arm can pick up a block and move it to another position, either on the table or on top of another block. The arm can pick up only one block at a time, so it cannot pick up a block that has another one on it. The goal will always be to build one or more stacks of blocks, specified in terms of what blocks are on top of what other blocks. For example, a goal might be to get block A on B and block C on D

PLANNING WITH STATE-SPACE SEARCH

- Straightforward approach is to use state-space search.
- Descriptions of actions in a planning problem specify both preconditions and effects, it is possible to search in either direction:
  - either forward from the initial state or backward from the goal

- Forward state-space search Planning- Progression planning, because it moves in the forward direction.
- Backward state-space search- Regression planning , allows us to consider only relevant actions.
  - An action is relevant to a conjunctive goal if it achieves one of the conjuncts of the goal



PARTIAL ORDER PLANNING

- In partial ordered planning (POP), ordering of the actions is partial. Also partial ordered planning don’t specify which action will come first out of the two actions which are placed.
- With partial ordered planning, problem can be decomposed, so it can work well in case the environment is non-cooperative.

It combines two action sequence:

i. First branch covers left sock and left shoe.

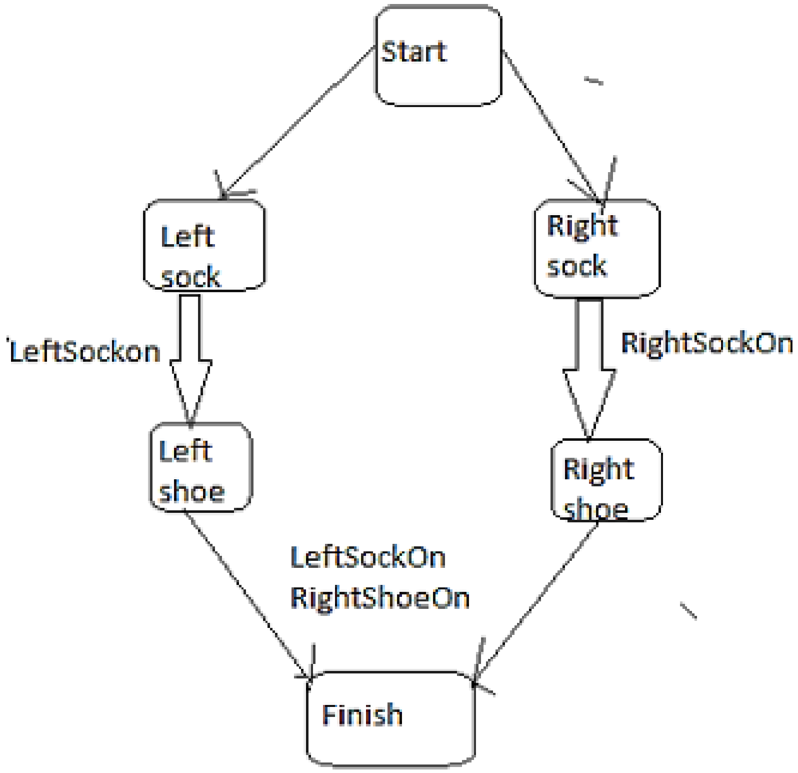
ii. In case, to wear a left shoe, wearing left sock is precondition, similarly.

iii. Second branch covers right shock and right shoe.

iv. Here, wearing a right shock is precondition for wearing the right shoe.

Once these actions are taken we achieve our goal and reach the finish state.

- E.g. Partial order planning of Wearing Shoe



Pop as search problem:

Set of actions:

These are the steps of plan.

For e.g.: Set of Actions = {Start, Rightsock, Rightshoe ,Leftsock, Leftshoe, Finish}

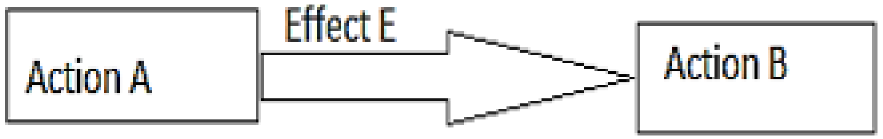
Set of ordering constraints/preconditions:

i. Preconditions are considered as ordering constraints.(i.e. without performing action “x” we cannot perform action “y”)

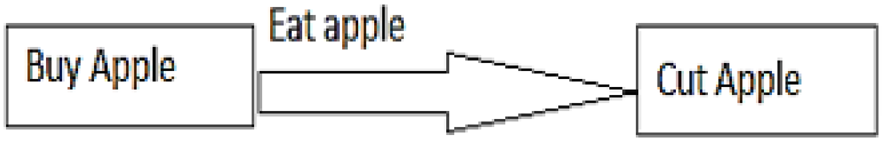
ii. For e.g.: Set of ordering = {Right-sock <right-shoe; left-sock<left-shoe}="" that="" is="" in="" order="" to="" wear="" shoe,="" first="" we="" should="" wear="" a="" sock.<="" p=""">

- i. You can understand that if you buy an apple its effect can be eating an apple and the precondition of eating an apple is cutting apple.
- ii. For e.g. Set of Causal Links = {Right-sock-> Right-sock-on
- Set of open preconditions:
- i. Preconditions are called open if it cannot be achieved by some actions in the plan.
- ii. Consistent Plan is a Solution for POP Problem
- iii. A consistent Plan doesn't have cycle of constraints; it doesn't have conflicts in the causal links and doesn't have open preconditions so it can provide a solution for POP problem.

a.

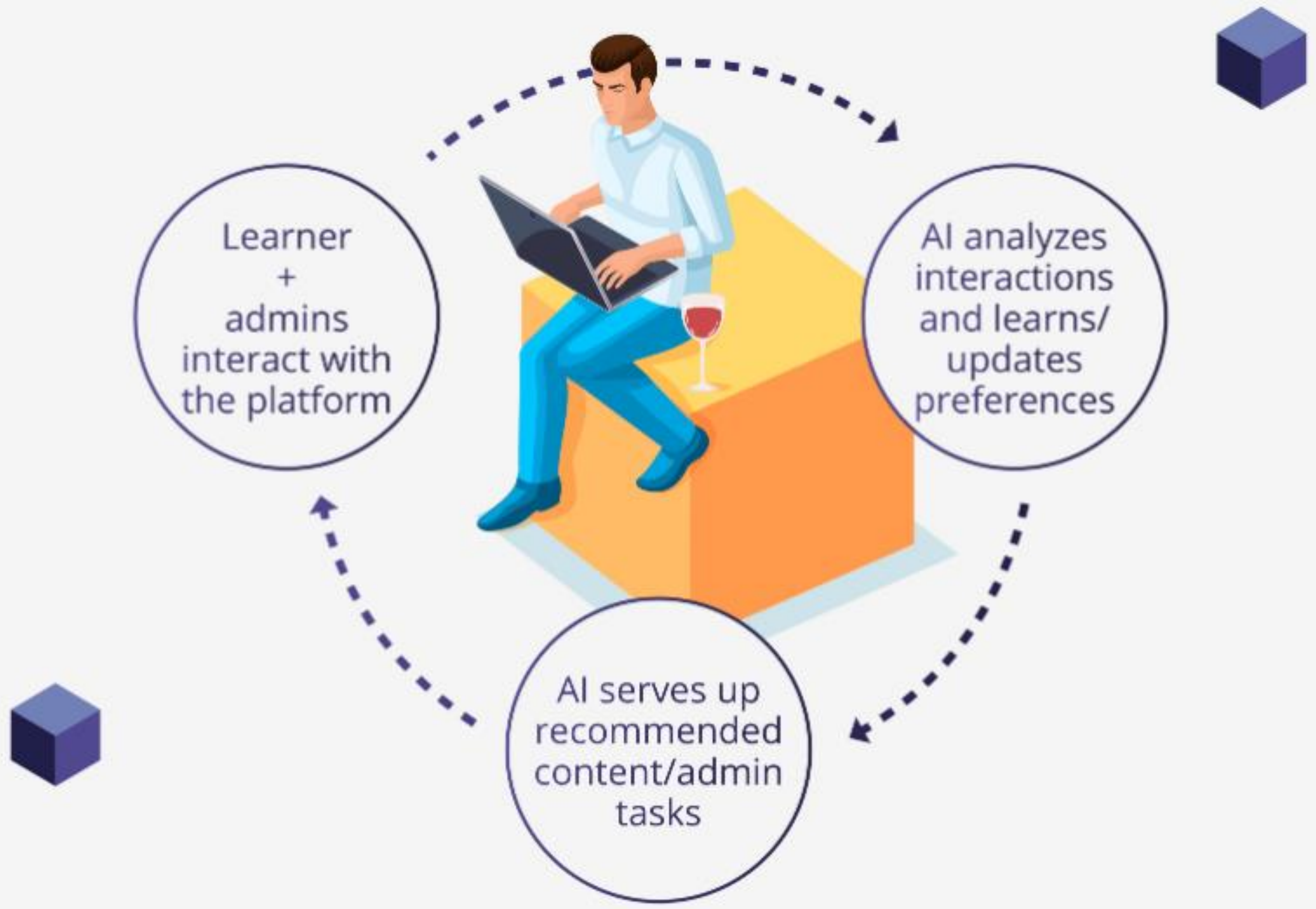


b.



Like A Fine Wine: AI Gets Better Over Time

The more you and your learners use and guide the AI-powered learning platform, the better the results it produces


Learner + admins interact with the platform
AI analyzes interactions and learns/ updates preferences
AI serves up recommended content/admin tasks

LEARNING

- An agent is learning if it **improves its performance on future tasks** after making observations about the world.
- Learning can range from the trivial → jotting down a phone number, to the profound → Albert Einstein, who inferred a new theory of the universe

Why would we want an agent to learn?

- If the design of the agent can be improved, why wouldn’t the designers just program in that improvement to begin with?
- There are three main reasons

- **First**, the designers cannot anticipate all possible situations that the agent might find itself in.
  - For example, a robot designed to navigate mazes must learn the layout of each new maze it encounters.

- **Second**, the designers cannot anticipate all changes over time;
  - a program designed to predict tomorrow’s stock market prices must learn to adapt when conditions change from boom to bust.

- **Third**, sometimes human programmers have no idea how to program a solution themselves.
  - For example, most people are good at recognizing the faces of family members, but even the best programmers are unable to program a computer to accomplish that task, except by using learning algorithms.

FORMS OF LEARNING

- Any component of an agent can be improved by learning from data.
- Improvements and techniques used to make them, depend on four major factors:
- **1. Which component is to be improved**.
- **2. What prior knowledge the agent already has.**
- **3. What representation is used for the data and the component.**
- **4. What feedback is available to learn from**

Components to be learned

- The components of the agents include:

1. A **direct mapping** from conditions on the current state to actions.
2. A means to **infer relevant properties** of the world from the percept sequence.
3. Information about the way the **world evolves** and about the results of possible actions the agent can take.
4. **Utility information** indicating the desirability of world states.
5. **Action-value information** indicating the desirability of actions.
6. **Goals** that describe classes of states whose achievement maximizes the agent's utility.

Each of these components can be learned

Consider, for example, an agent training to become a taxi driver.

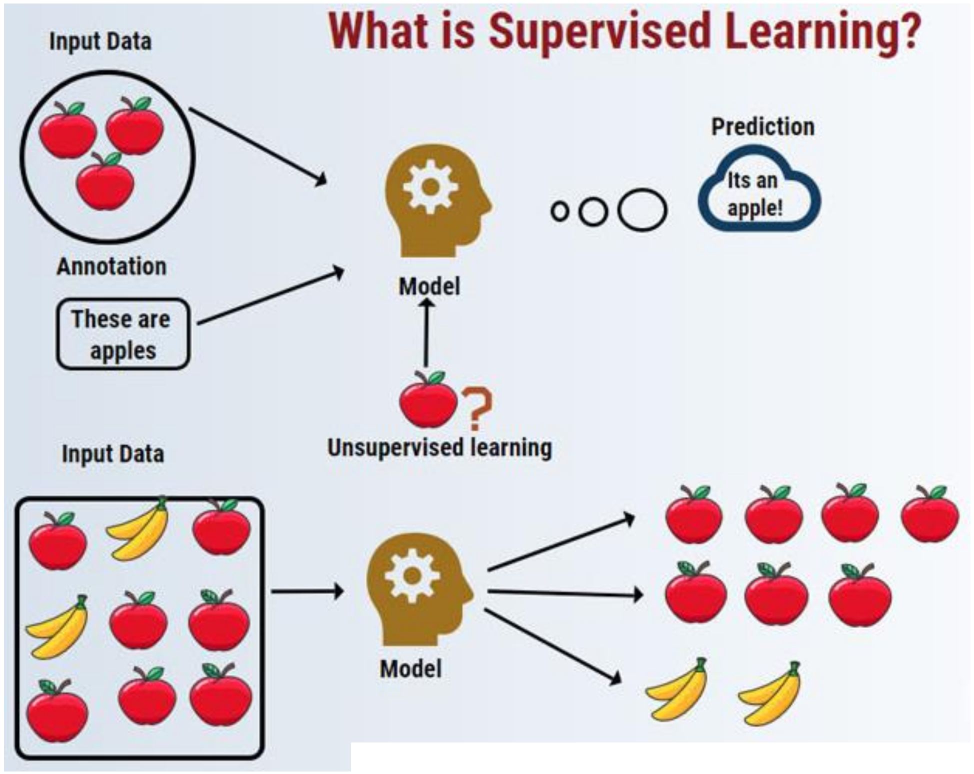
- Every time the instructor shouts "**Brake**!" the agent might learn a condition–action rule for when to brake (component 1);
- the agent also learns every time the instructor does not shout.
- By seeing many camera images that it is told contain buses, it can learn to recognize them (2).
- By trying actions and observing the results—for example, braking hard on a wet road—it can learn the effects of its actions (3).
- Then, when it receives no tip from passengers who have been thoroughly shaken up during the trip, it can learn a useful component of its overall utility function (4).

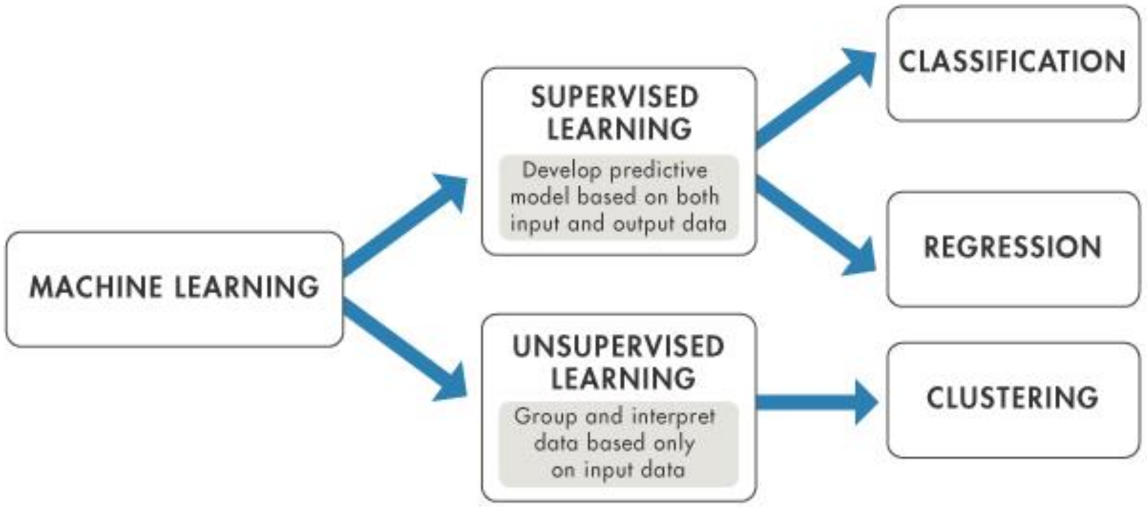
Feedback to learn from

There are three types of feedback that determine the three main types of learning:

- **Supervised Learning**
- **Unsupervised Learning**
- **Reinforcement Learning**

- **Supervised learning** tasks find patterns where we have a dataset of “right answers” to learn from.



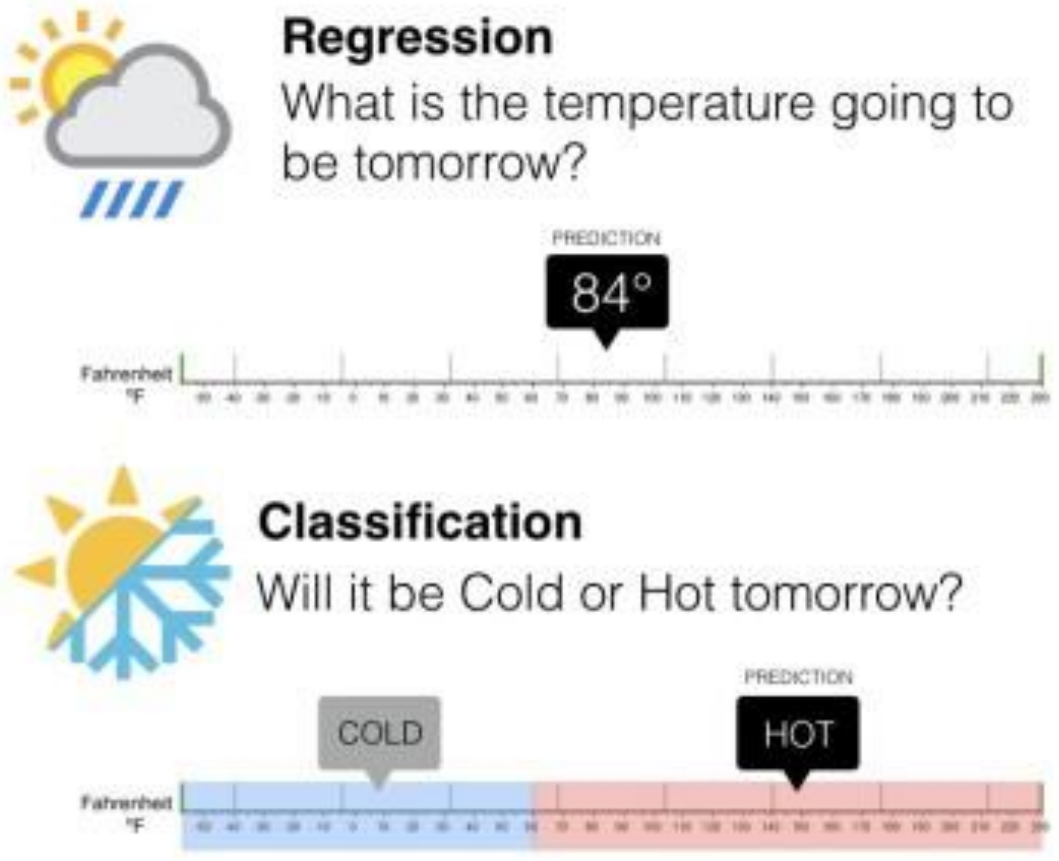

MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Supervised Learning

- Access to examples of **correct input-output pairs** that we can show to the machine during the training phase.
- Common example of handwriting recognition is typically approached as a supervised learning task.
  - We show the computer a number of images of handwritten digits along with the correct labels for those digits, and the computer learns the patterns that relate images to their labels.

Classification of Supervised Learning

- REGRESSION
- CLASSIFICATION



Classification of Supervised Learning

➢ **Regression**

- Problem of **estimating or predicting** a continuous quantity.
  - What will be the value of the S&P 500 one month from today?
  - How tall will a child be as an adult?
  - How many of our customers will leave for a competitor this year?

Gather past examples of “right answer” input/output pairs that deal with the same problem.

For the inputs, we would identify **features** that we believe would be predictive of the outcomes that we wish to predict.

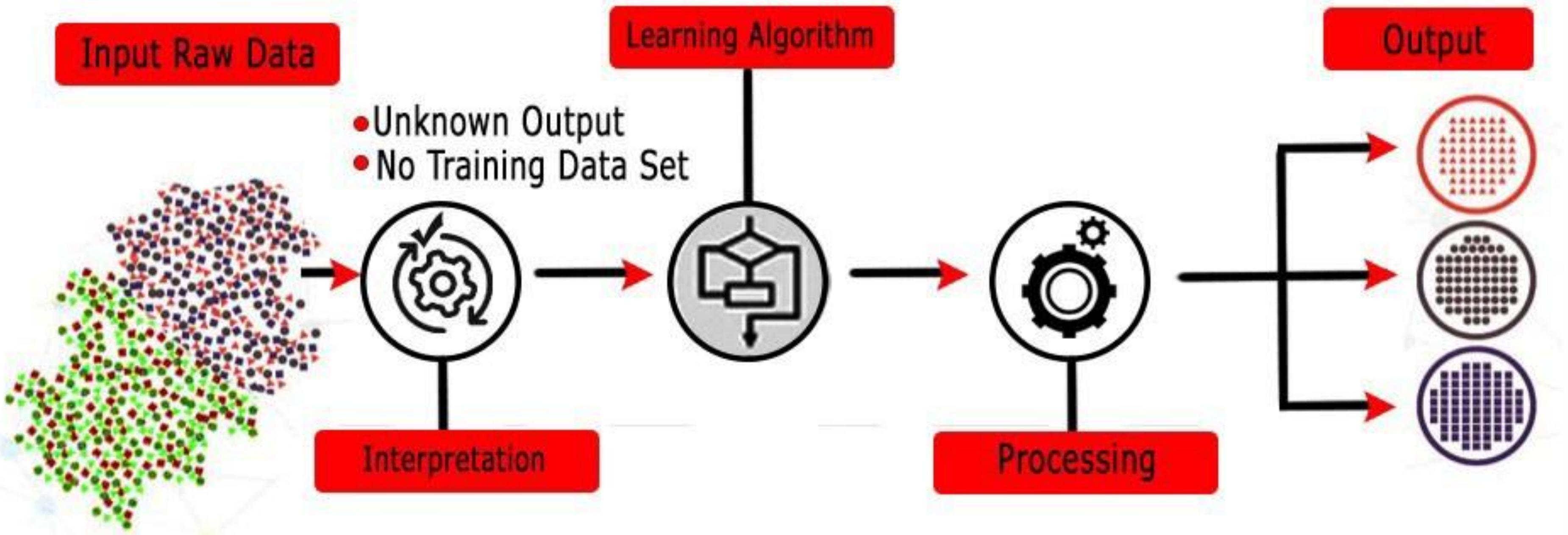
Classification of Supervised Learning

- **CLASSIFICATION** deals with assigning observations into **discrete categories,** rather than estimating continuous quantities.
- Ex: two possible categories;→**binary classification**.
- Many important questions can be framed in terms of binary classification.
  - Will a given customer leave us for a competitor?
  - Does a given patient have cancer?
  - Does a given image contain a hot dog?
    - For instance, a simple solution to the handwriting recognition problem is to simply train a bunch of binary classifiers: a 0-detector, a 1-detector, a 2-detector, and so on, which output their certainty that the image is of their respective digit. The classifier just outputs the digit whose classifier has the highest certainty.

UNSUPERVISED LEARNING

- Agent **learns patterns in the input** even though no explicit feedback is supplied.
  - Most common unsupervised learning task→Clustering- detecting potentially useful clusters of input examples.
    - For example, a taxi agent might gradually develop a concept of “good traffic days” and “bad traffic days” without ever being given labeled examples of each by a teacher.

Unsupervised Learning



- **Unsupervised learning** tasks find patterns where we don’t.
  - The “right answers” are unobservable, or infeasible to obtain, or maybe for a given problem, there isn’t even a “right answer” per se.
- Unsupervised learning is used against data without any historical labels.
- The system is not given a pre-determined set of outputs or correlations between inputs and outputs or a “correct answer.”
- The algorithm must figure out what it is seeing by itself, it has no storage of reference points.
- The goal is to explore the data and find some sort of patterns of structure.

Clustering

- Large subclass of unsupervised tasks is the problem of **clustering**.
  - Clustering refers to grouping observations together in such a way that members of a common group are similar to each other, and different from members of other groups.
- Common application →marketing, where we wish to identify segments of customers or prospects with similar preferences or buying habits.
- **Major challenge**→ Difficult /Impossible to know how many clusters should exist, or how the clusters should look.

Imagine yourself as a person that has never heard of or seen any sport being played. You get taken to a football game and left to figure out what it is that you are observing. You can’t refer to your knowledge of other sports and try to draw up similarities and differences that will eventually boil down to an understanding of football. You have nothing but your cognitive ability.

- Unsupervised learning places AI in an equivalent of this situation and leaves it to learn using only its on/off logic mechanisms that are used in all computer systems.

REINFORCEMENT LEARNING

- Learns by interacting with its environment.
- It receives rewards by performing correctly and penalties for doing so incorrectly.
- Learns without having to be directly taught by a human – **it learns by seeking the greatest reward and minimising penalty.**
    - Learning is tied to a **context** because what may lead to maximum reward in one situation may be directly associated with a penalty in another.

REINFORCEMENT LEARNING

- Reinforcement learning the agent learns from a series of reinforcements—rewards or punishments.
  - For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong.
  - The two points for a win at the end of a chess game tells the agent it did something right.
  - It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it.

- Reinforcement learning tends to be used for gaming, robotics and navigation.
- The algorithm discovers which steps lead to the maximum rewards through a process of trial and error.
- When this is repeated, the problem is known as a **Markov Decision Process.**

Example

Facebook’s News Feed is an example most of us will be able to understand.

Facebook uses machine learning to personalise people’s feeds. If you frequently read or “like” a particular friend’s activity, the News Feed will begin to bring up more of that friend’s activity more often and nearer to the top. Should you stop interacting with this friend’s activity in the same way, the data set will be updated and the News Feed will consequently adjust

Semi supervised learning

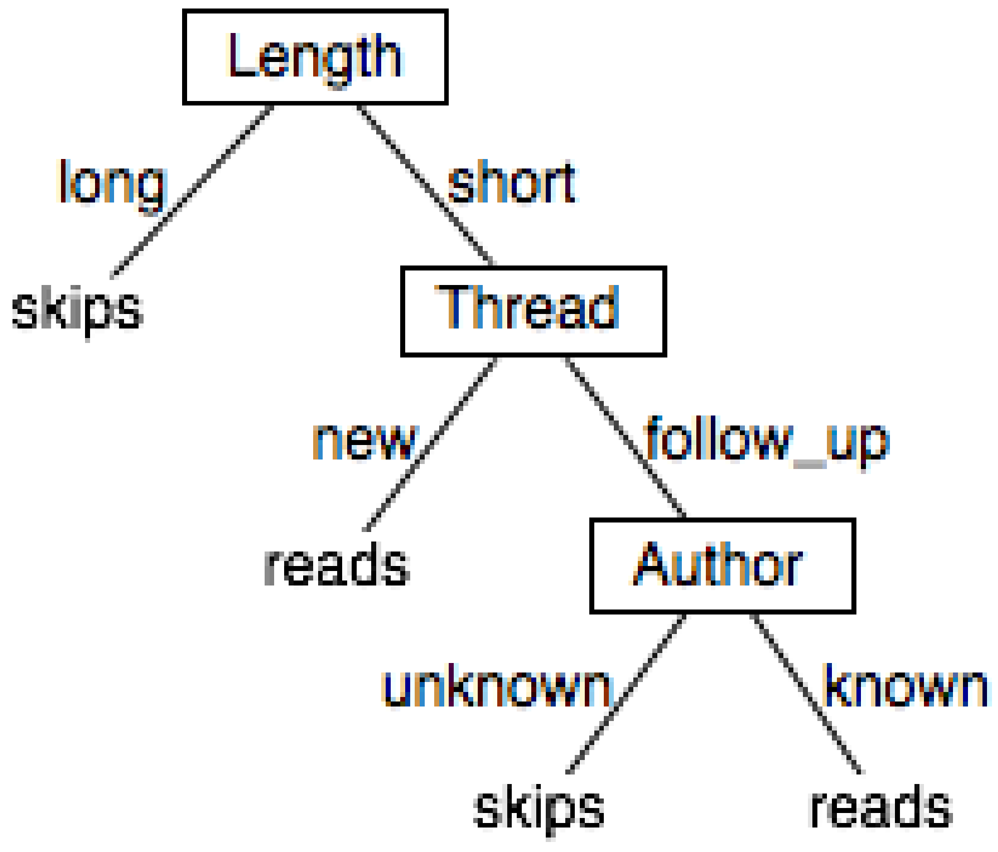
- Semi-supervised learning falls somewhere in the middle of supervised and unsupervised learning.
- It is used because many problems that AI is used to solving require a balance of both approaches.

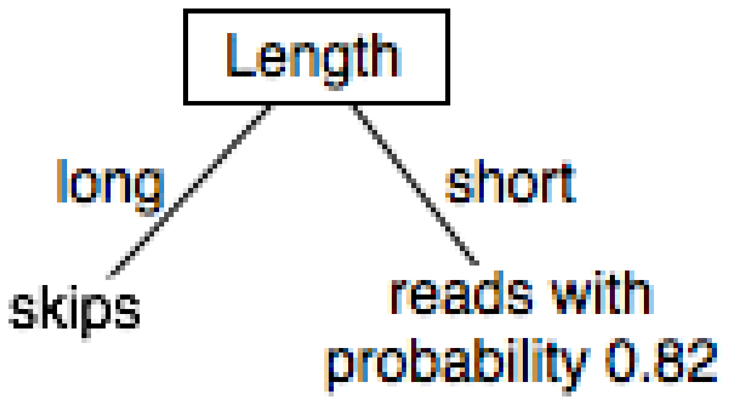
Inductive Learning

- This involves the process of ***learning by example*** -- where a system tries to induce a general rule from a set of observed instances.
- This involves classification -- assigning, to a particular input, the name of a class to which it belongs.
- Classification is important to many problem solving tasks.
- A learning system has to be capable of evolving its own class descriptions
  - Initial class definitions may not be adequate.
  - The world may not be well understood or rapidly changing.
- The task of constructing class definitions is called *induction* or *concept learning*

LEARNING DECISION TREE

- A decision tree is a simple representation for classifying examples.
- Decision tree learning is one of the most successful techniques for supervised classification learning.
  - Assume that all of the features have finite discrete domains, and there is a single target feature called the **classification**.
  - Each element of the domain of the classification is called a **class**.

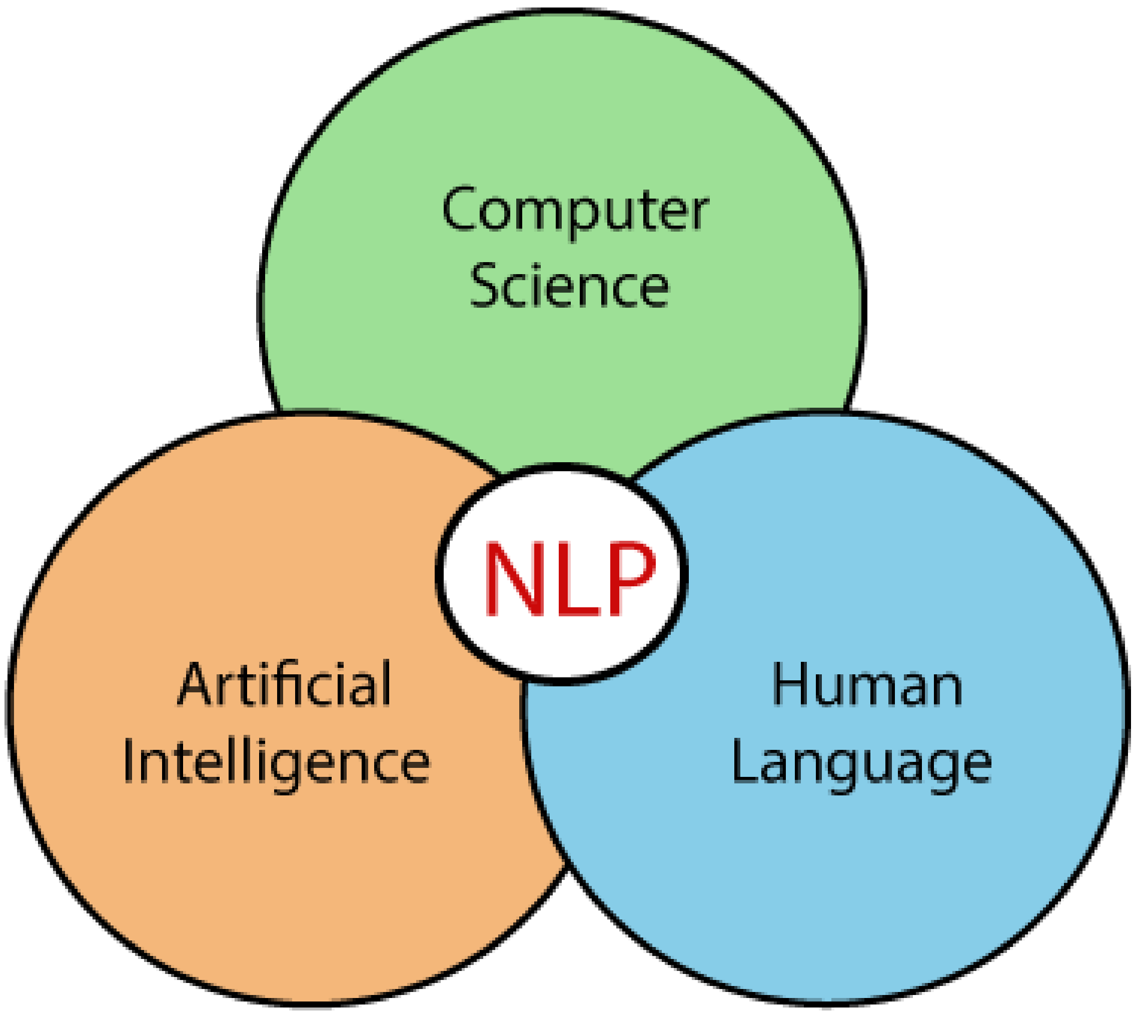

Length
long
short
skips
Thread
new
follow_up
reads
Author
unknown
known
skips
reads


Length
long
short
skips
reads with probability 0.82

Example

Shows possible decision trees for the example Each decision tree can be used to classify examples according to the user's action. To classify a new example using the tree on the left, first determine the length. If it is long, predict skips. Otherwise, check the thread. If the thread is new, predict reads. Otherwise, check the author and predict read only if the author is known. This decision tree can correctly classify

The tree on the right makes probabilistic predictions when the length is short. In this case, it predicts reads with probability 0.82 and so skips with probability 0.18.


Computer Science
Artificial Intelligence
NLP
Human Language

Natural Language Processing (NLP)

- NLP refers to AI method of communicating with an intelligent systems using a natural language such as English.
- Used in →Intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.
- Involves making computers to perform useful tasks with the natural languages humans use.
- The input and output of an NLP system can be −
  - Speech
  - Written Text

Components of NLP

1. Natural Language Understanding (NLU)

- Helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.
- Used in Business applications to understand the customer's problem in both spoken and written language.
- NLU involves the following tasks -
  - Maps the given input into useful representation.
  - Analyzes different aspects of the language.

2. Natural Language Generation (NLG)

- Natural Language Generation (NLG) acts as a **translator** that converts the computerized data into natural language representation.
- It mainly involves Text planning, Sentence planning, and Text Realization.

Steps to build an NLP

- Step1: Sentence Segmentation- Paragraph→ Sentences
- Step2: Word Tokenization- Sentences → Words/Tokens
- Step3: Stemming-Normalize words[intelligence, intelligent,& intelligently→root word "intelligen.“]
- Step 4: Lemmatization-Grouping [Stemming]
- Step 5: Identifying Stop Words- removing is , and , a , the…
- Step 6: Dependency Parsing- relation of words in the sentence
- Step 7: POS tags-Noun, verb, adverb, and Adjective
- Step 8: Named Entity Recognition (NER)-detecting the named entity such as person name, movie name, organization name, or location.; **Steve Jobs** introduced I phones
- Step 9: Chunking-collect the individual piece of information and grouping them into bigger pieces of sentences.

Applications of NLP

Question Answering

- Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

Spam Detection

- Spam detection is used to detect unwanted e-mails getting to a user's inbox.

Sentiment Analysis

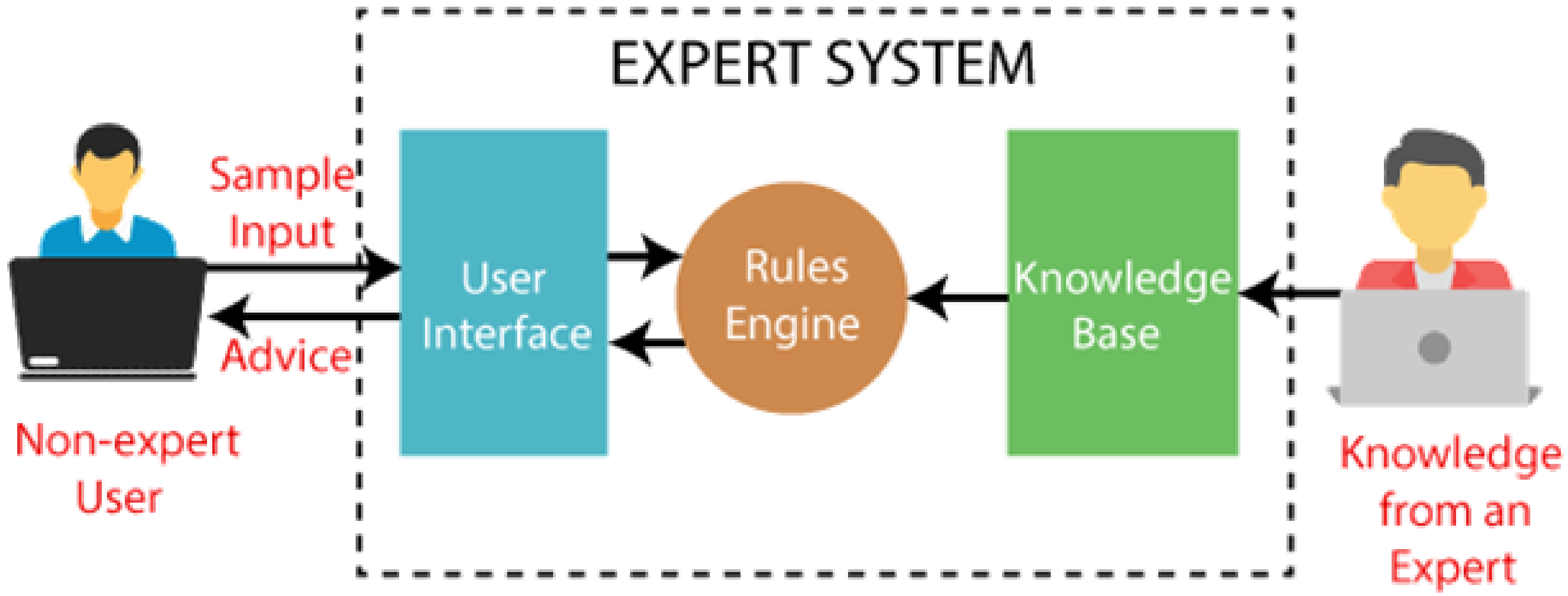
- Sentiment Analysis is also known as **opinion mining**
  - Analyse the attitude, behaviour, and emotional state of the sender.
- **Google Translator, Spelling correction, Speech Recognition , ChatBot,etc…..**

Expert systems (ES)

- One of the prominent research domains of AI.
- It is introduced by the researchers at Stanford University, Computer Science Department.

What are Expert Systems?

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


EXPERT SYSTEM
Sample Input
Advice
Non-expert User
User Interface
Rules Engine
Knowledge Base
Knowledge from an Expert

Characteristics of Expert System

- **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provides the output in the same way.
- **Reliable:** It is much reliable for generating an efficient and accurate output.
- **Highly responsive:** ES provides the result for any complex query within a very short period of time.

Components of Expert System

- An expert system mainly consists of three components:
  - **User Interface**
  - **Inference Engine-Rules(Fwd chaining&Bckward chaining)**
  - **Knowledge Base**

Benefits of Expert Systems

- **Availability** − They are easily available due to mass production of software.
- **Less Production Cost** − Production cost is reasonable. This makes them affordable.
- **Speed** − They offer great speed. They reduce the amount of work an individual puts in.
- **Less Error Rate** − Error rate is low as compared to human errors.
- **Reducing Risk** − They can work in the environment dangerous to humans.
- **Steady response** − They work steadily without getting motional, tensed or fatigued.

Expert Systems Limitations

- No technology can offer easy and complete solution.
- Large systems are costly, require significant development time, and computer resources.
- ESs have their limitations which include −
  - Limitations of the technology
  - Difficult knowledge acquisition
  - ES are difficult to maintain
  - High development costs