

# Module 5

**Syllabus:** Natural Language Processing, Learning and Expert Systems .

## Chapter 1 : Natural Language Processing

### 1. What is Language? What is Natural Language?

**Answer:**

Language is a mean for communicating about the world with others. By studying language we can come to understand more about the world.

In computing, natural language refers to a human language such as English, Russian, German, or Japanese ( as *distinct from the typically artificial command or programming language with which one usually talks to a computer*). The term usually refers to a written language but might also apply to spoken language.

Entire Language processing problem is divided into the following two tasks:

1. Processing written text using lexical , syntactic and semantic knowledge of the language as well as the required real world information.
2. Processing spoken language using all the information needed above plus additional knowledge about phonology as well as enough added information to handle the further ambiguities that arise in speech.

**2. What is Natural Language Processing (NLP)? What is the Need of It? List and explain the different steps involved in NLP**

**Answer:** Natural Language Processing (NLP) refers to *AI method of communicating with an intelligent system using a natural language such as English*. The input and output of an NLP system can be –

- **Speech**
- **Written Text**

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

The field of NLP involves making computers to perform useful tasks with the natural languages humans use.

**Steps in NLP: There are general five steps –**

1. Morphological Analysis
2. Syntactic Analysis (Parsing)
3. Semantic Analysis
4. Discourse Integration
5. Pragmatic Analysis

### **1. Morphological Analysis**

Morphology is the study of word formation –how words are built up from smaller pieces. Morphological Analysis – Individual words are analyzed into their components(morphemes) , resolving Ambiguity and non-word tokens such as punctuation are separated from the words.

**Example1 :** Morphological analysis of the sentence “*I want to print Bills .doc file*” must do the following things :

- Pull apart the word “Bills” into proper noun “Bill” and the possessive suffix + “s”
- Recognize the sequence “.doc” as file extension that is function as an adjective in the sentence.
- Assign *syntax categories to all the words* in the sentence.

## Example 2:

Washing = Wash + ing ,

Browser = Browse + er

Rats = Rat + s

### Ambiguity

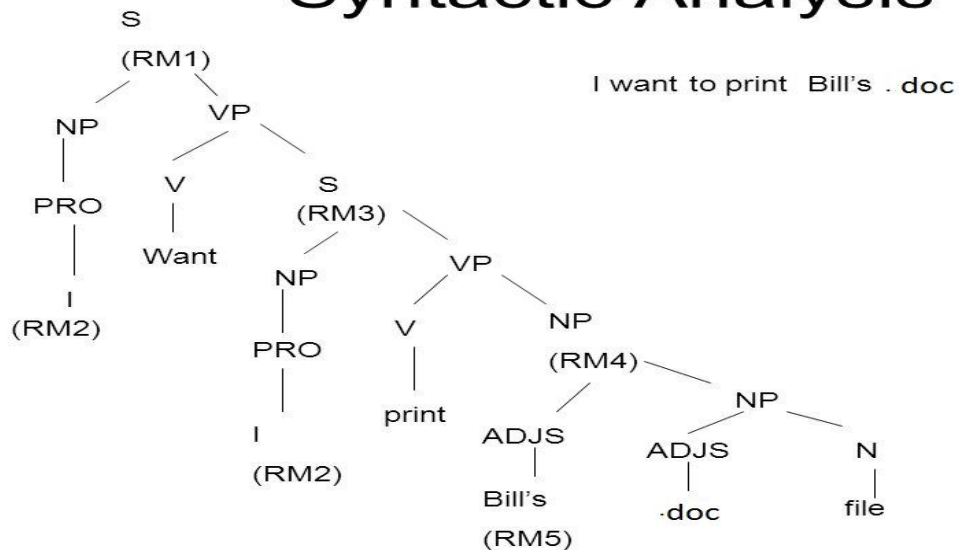
Flies: 1. Fly(Verb) + Action

Fly(Noun) + Plural

## 2. Syntactic Analysis

Syntactic Analysis (Parsing) – It involves analysis of words in the sentence as per the 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. Syntax analysis exploit the results of morphological analysis to build a structural description of sentence . The goal of the process , called *parsing* is to convert the flat list of words that forms sentences into a structure called *parse tree*. The result of syntactic analysis of “I want to print Bills .doc file” is as given below :

## Syntactic Analysis



### 3. Semantic Analysis

**Semantic Analysis** – The structure created by the syntactic analyzer are assigned meanings. 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”. Semantic Analysis must do two important things:

1. *It must map individual words into appropriate objects in the knowledge base or database*
2. *It must create the correct structures to correspond to the way the meanings of the individual words combine with each other .*

**Knowledge Base Fragment for the sentence “I want to print Bills .doc file “**

**User**

isa : Person

\*login\_name : mustbe<string>

**user068**

instance : user

login\_name : rahul

**user073**

instance : user

login\_name: BILL

**F1**

Instance :File\_struct

Name : Stuff

Extension : doc

Owner : user073

Directory : C:/temp/

File\_Struct

isa : information\_object

### Printing

isa : Physical Event

\*agent : mustbe<animate or program>

\*object: mustbe <information \_object>

### Wanting

isa : Mental\_Event

\*agent : mustbe<animate>

\*object:mustbe<state or event>

### Commanding

isa : Mental\_Event

\*agent :must\_be<animate>

\*perofmed:must\_be<animate or program>

\*object:must\_be<event>

### This\_system

Instance: Program

## Partial Meaning for a Sentence

RM1	{The whole Sentence}	
	Instance : Wanting	
	Agent : RM2	{}
	Object:RM3	{ a printing event}
RM2:	{}	
RM3		
	Instance : printing	
	Agent : RM2	{}
	Object :RM4	{Bill's .doc file}
RM4		
	Instance : File_struct	
	Extension: .doc	
	Owner : RM5	{BILL}
RM5		
	Instance : Person	
	First_Name : BILL	

**4.Discourse Integration:** The meaning of any sentence depends upon the meaning of the sentence that precede it and may influence the meaning of the sentences that follow it.

**Example1 :** The word “it” in the sentence , “John wanted it”, depends on the prior discourse context, while the word “John “ may influence the meaning of later sentences ( such as , “He always had”)

**Example2:** In the sentence “I want to print the Bills .doc File “ , the system must model the current discourse context from which one can learn that the current user(who type the word “I” ) is user068 and the owner of the file is “BILL (user073)”.

**5. Pragmatic Analysis:** 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. **For example:** The sentence “*Do you know what time it is ?* Should be interpreted as a request to be told the time.

The final step toward effective understanding is to decide what to do as a result. Some possible things are

- To record the fact an action to be done based on the knowledge base.
- **Some sentences intended effect is clearly declarative** i.e. precisely the correct thing to do.
- **Some sentences intended effect is clearly different.** This intended effect can be discovered by applying a set of rules that characterize co operative dialogues

## Example:

Meaning

Instance : Commanding

Agent : USer068

Performer : This\_System

Object : P27

P27

Instance : printing

Agent : This\_System

Object : F1

In the above example the user068 has issued a command for printing a file F1 through the printer say P27. The final step in pragmatic processing is to translate when necessary from the knowledge based representation to a command to be executed by the system.

**Example :**     lpr   /temp/xyz.doc

### 3.Explain the process of Syntactic Processing of NLP in detail.

**Answer:** It is the step in which a flat input sentence is converted into a hierarchical structure that corresponds to the units of meaning in the sentence as per the given grammar. This process is called parsing. The two important components of the system which perform syntactic processing are

1. Grammar
2. Parser

**1. Grammar:** A grammar specifies two things about a language

- Its weak generative capacity: The set of sentences that are contained within the language
- Its Strong generative Capacity: The structure to be assigned to each grammatical sentence of the language

The most common way to represent grammar is as a set of production rules. Figure shows a simple context free phrase structure grammar for English .

- $S \rightarrow NP VP$
- $NP \rightarrow the NP1$
- $NP \rightarrow PRO$
- $NP \rightarrow PN$
- $NP \rightarrow NP1$
- $NP1 \rightarrow ADJS N$
- $ADJS \rightarrow \epsilon \mid ADJ ADJS$
- $VP \rightarrow V$
- $VP \rightarrow V NP$
- $N \rightarrow file \mid printer$
- $PN \rightarrow Bill$
- $PRO \rightarrow I$
- $ADJ \rightarrow short \mid long \mid fast$
- $V \rightarrow printed \mid created \mid want$

Read the first rule as “ A sentence is composed of a noun phrase followed by a verb phrase”

| means OR

$\epsilon$  means empty string

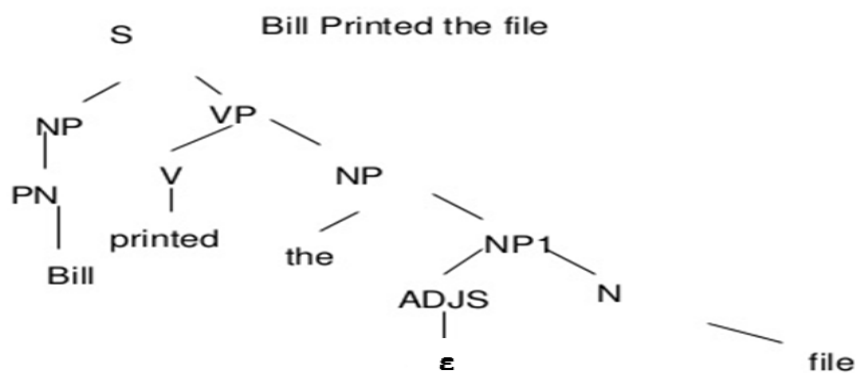
Symbols that are further expanded by rules are called nonterminal symbols .

## 2.Parser

Regardless of the theoretical basis of the grammar the parsing process takes the rules of the grammar and compares them against the input sentence. Each rule that matches adds something to the complete structure that is being built for the sentence. The simplest structure that is being built for the sentence is a parse tree, which simply records the rules and how they are matched. Every node of the parse tree corresponds either to an input word or to a nonterminal in our grammar. Each level in the parse tree corresponds to the application of one grammar rule.

**Example:**

**A Parse tree for a sentence :**

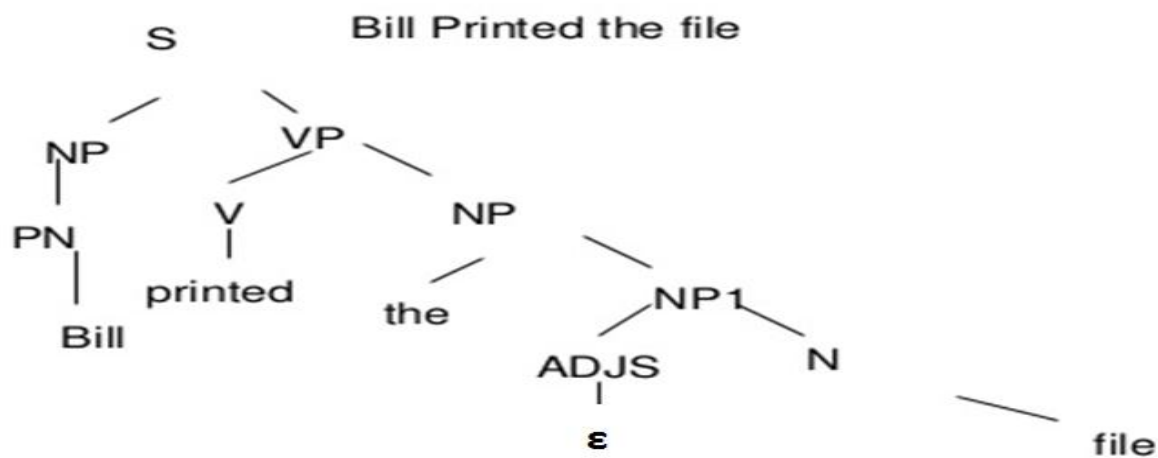


## Top-down versus Bottom-Up parsing

- To parse a sentence, it is necessary to find a way in which that sentence could have been generated from the start symbol. There are two ways this can be done:
  - Top-down Parsing: Begin with start symbol and apply the grammar rules forward until the symbols at the terminals of the tree correspond to the components of the sentence being parsed.
  - Bottom-up parsing: Begin with the sentence to be parsed and apply the grammar rules backward until a single tree whose terminals are the words of the sentence and whose top node is the start symbol has been produced.
- The choice between these two approaches is similar to the choice between forward and backward reasoning in other problem-solving tasks.
- The most important consideration is the branching factor. Is it greater going backward or forward?
- Sometimes these two approaches are combined to a single method called "bottom-up parsing with top-down filtering".



## Example for Top Down Parsing :



## 4. What are the different types of parsers?

### Answer :

1. Chart Parser [Winograd,1983] : Which provide a way of avoiding backup by storing intermediate constituents *so that they can be reused along alternative parsing paths.*
2. Definite clause grammars [ Pereira and Warren,1980] in which grammar rules are written as PROLOG clauses and then PROLOG interpreter is used to perform to down , depth first parsing.
3. Augmented transition networks (or ATNs) [ Woods, 1970] in which the parsing process is described as the transition from a start state to a final state in a transition network that corresponds to a grammar of English.

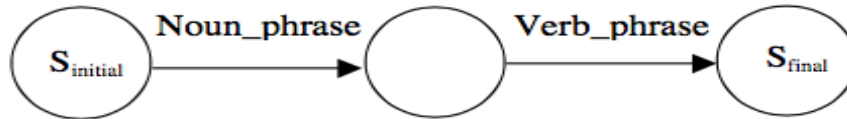
## 5. Explain What is ATN with examples .

### Answer :

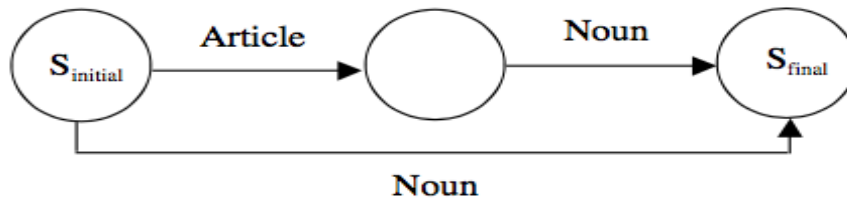
Augmented transition networks (or ATNs) in which the parsing process is described as the transition from a *start state* to a *final state* in a transition network that

corresponds to a grammar of English. ATN is similar to a finite state machine in which the class of labels that can be attached to the arcs that define transitions between states has been augmented.

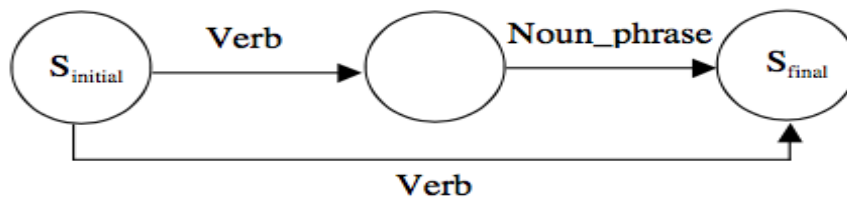
**Sentence:**



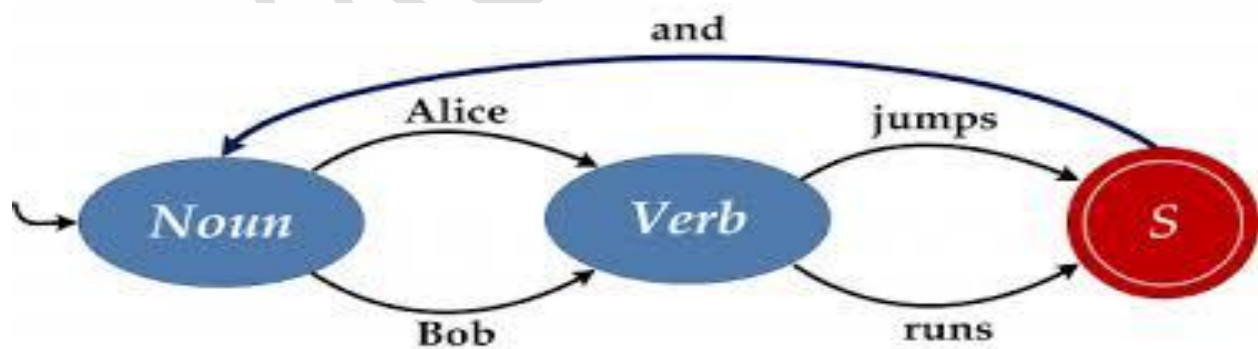
**Noun\_phrase:**



**Verb\_phrase:**



**Example: ATN for “Alice Jumps and Bob Runs”**



## 6. Explain the concept of Spell Checking.

### Answer :

Spell checking involves *identifying words , correct words , wrong words and non words* and also suggesting the *possible alternatives for its correction*.

A spell checker is one of the basic tools required for language processing. It is used in a wide variety of computing environments including *word processing, character or text recognition systems, speech recognition and generation* .

## 7. Explain three causes of Spelling Errors with example?

### Answer : Three Causes of Spelling Errors

Damerau (1964) conducted a survey on misspelled words and found that most of the non words were a result of single error misspellings. Based on this survey it was found that the three causes of errors are :

1. **Insertion** : Insertion of an extra letter while typing .
  - *Eg: maximum typed as maxiimum. The extra i has been inserted within the word.*
2. **Deletion** : A case of letter missing or not typed in a word .
  - *Eg: netwrk instead of network.*
3. **Substitution** : Typing of a letter in place of the correct one .
  - *Eg: As in intellugence where letter i has been wrongly substituted by u.*

## 8. What are the three types of spelling Errors?

### Answer: Types of Spelling Errors

Spelling Errors may be classified into the following types:

1. Typographic errors : As the name suggests these errors are those that are caused due to mistakes committed while typing. A typical example is *netwrk* instead of *network*.
2. Orthographic errors : These are on the other hand, result due to a lack of comprehension of the concerned language on part of the user. Example of such spelling errors are *arithmetic* , *wellcome* and *accomodation*.
3. Phonetic errors : These result due to poor cognition on part of the listener. The word *rough* could be spelt as *ruff* and *listen* as *lisen*. Words like *piece* , *peace* and *peas* , *reed* and *read* and *quite* and *quiet* may all be spelt correctly but can lead to confusion depending on the context.

## 9. Explain the different types of Spell Checking techniques.

**Answer:** Spell checking techniques can be broadly classified into three categories:

1. Non word error detection
2. Isolated word error detection
3. Context dependent Error detection and correction.

### 1. Non-Word Error Detection

This process involves the detection of misspelled words or non words .

**For example** : The word *soper* is a non word , its correct form being *super* or *sober* .  
The most commonly used techniques used to detect such errors are the N- grams analysis and Dictionary look up.

According to N- gram techniques (based on probability theory) those strings that contain highly infrequent sequences are treated as cases of spelling error.

Dictionary look up involves the use of an efficient dictionary look up couple with *pattern matching algorithms* , *dictionary partition schemes* and *morphological methods*.

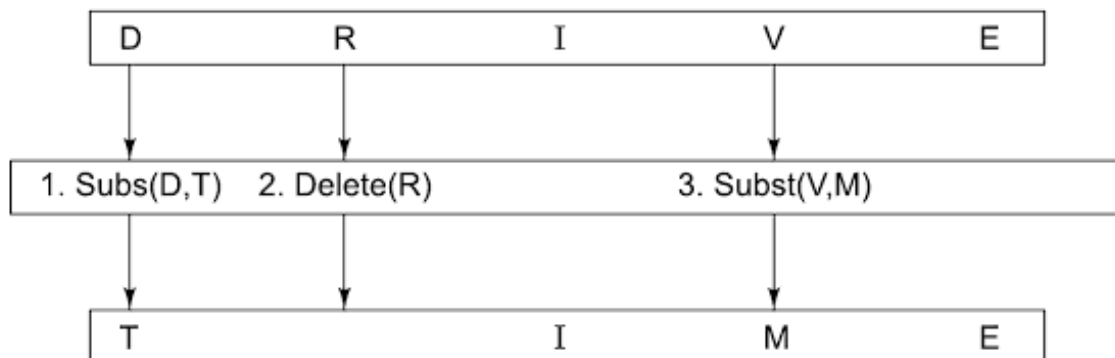
## 2. Isolated word error correction

This process focusses on the correction of an *isolated non-word* by finding its nearest and meaningful word and makes an attempt to rectify the error. The techniques employed herein include the *minimum edit distance technique*, *similarity key techniques*, *rule based methods*, *N-gram*, *probabilistic* and *Neural Network based techniques*. Isolated word error correction may be looked upon as a *combination of three sub problems – error detection, candidate (correct word) generation and ranking of the correct candidates*.

### Minimum Edit Distance Technique

Wagner (1974) defined the minimum edit distance between the misspelled word and the possible correct candidate as the minimum number of edit operations needed to transform the misspelled words to the correct candidate .

**Example :** The minimum edit distance between DRIVE and TIME is 3



## 3. Context Dependent Error Detection

These process, in addition to detect errors, *try to find whether the corrected words fits into the context of the sentence*. These are naturally more complex to implement and require more resources than the previous methods.

**Example:** "Peace comes from Within" may be typed as "Piece comes from within". Such types of errors are solved using statistical natural language processing technique.

## 10. Explain Soundex Algorithm.

### Answer :

The Soundex algorithm can be effectively used as a simple phonetic based spell checker. It makes use of rules found using the phonetics of the language in question. Developed by Robert Russell and Margaret Odell in the early 20<sup>th</sup> century, *The Soundex Algorithm uses a code to check for the closest word.* The code for a word consists of its *first letter followed by three numbers that encode the remaining consonants.*

### Algorithm

- Remove all punctuations marks and capitalize the letters in the word.
- Retain the first letter of the word
- Remove any occurrence of the letters – A,E,I,O,U,H,W,Y apart from the very first letter.
- Replace the letters (other than the first ) by the numbers shown in the table

Letter ( s)	Substitute with Integer
B,F,P,V	1
C,G,J,K,S,X,Z	2
D,T	3
L	4
M,N	5
R	6

- IF two or more adjacent letters , not separated by vowels , have the same numeric value , retain only one of them.
- Return the first four characters ; pad with zeroes if there are less than four.

Nominally the Soundex Code contains : First Character of Word, Code1,Code2,Code3

#### Soundex Code for Some Words

Word	Soundex Code
Grate ,great	G630
Network, network	N362
Henry, Henary	H560
Torn	T650
Worn	W650
Horn	H650

## 11. Explain Statistical NLP and its different issues.

### Answer :

The NLP based on Probability theory is called statistical NLP. Statistical information extracted from a large corpus of the concerned language can aid in disambiguities. Statistical NLP aims to perform statistical inference for the field of NLP. Statistical inference consists of taking some data generated in accordance with some unknown probability distribution and making inferences.

Some of the issues of statistical NLP are :

- Corpora

- Counting the elements in a Corpus

- N-Grams

- Smoothing

### Corpora

Corpora/Corporus is defined as a large collection segments like written text or spoken words of a language. Corpora is available in the form of a collection of *raw text* or in a *sophisticated annotated or marked up form* where in information about

the words is also included to ease the process of language processing . It may contains written or spoken language , new or old texts, texts from one or either different language. Textual content may be the content of a complete book or books , newspaper , web pages , journal , speeches , etc.

**Example :** British National Corporation (BNC) is said to have a collection of around a hundred million written and spoken language samples.

- Some Corpora may contain texts on a particular domain of study or a dialect. Such Corpora are called sublanguage corpora.
- Some corpora contain extra information regarding the content of raw text , labelled with semantic linguistic tag. Such corpora is said to be Annotated.
- A Tree bank is an annotated corpus that contain parse trees and other related syntactic information. Example : Penn Tree bank available at University of Pennsylvania.
- Some Corpora contain a collection of texts which have been translated into one or several other languages. These corpora are referred to as parallel corpora .
- The index or list of the important words in text or a group of texts of corpora is called concordance.
- A Collocation is the collection of words that are often observed together in a text. Example : *Christmas Gifts , Chain Smoker , A hard nut , Extremely beautiful, etc.*

## N-Grams

N- grams are basically sequence of N- words or strings where N could assume the value 1,2,3 -----.

N= 1 mean unigram ( just one word)

N=2 mean bigram (sequence of two words )

N = 3 mean trigram

N= 4 mean tetragrams , etc

Different words in a corpus have their own frequency .

Table below gives the frequencies and probabilities of some words in a corpus of the novel “ The Scarlet Pimpernel” by Baroness Orczy. The novel has around 87163 words and 8822 types of word forms.



### Frequencies and probabilities of some words in a corpus

Word	Word Frequency	Probability = (Word Frequency)/ Total number of words
the	4508	0.051
of	2474	0.028
and	2353	0.027
to	2267	0.026
a	1559	0.018
her	1137	0.013
had	1077	0.012
she	935	0.0107
It	444	0.005
said	399	0.0046
man	174	0.002
Scarlet	98	0.0011
woman	66	0.00076
beautiful	39	0.00045
fool	18	0.00002
However	19	0.00002

If there are  $n$  – words in a sentence and assuming the occurrence of each words at their appropriate places to be independent events , the Probability  $P(w_1, w_2, w_3 \dots w_n)$  can be expressed using chain rule a

$$P(W) = P(w_1) \cdot P(w_2 | w_1) \cdot P(w_3 | (w_1, w_2)) \dots P(w_n | (w_1, w_2 \dots w_{n-1}))$$

According to Markov assumption we refer to a previous word and find the probability of the next word , a more apt sentence is created . For a bigram model

$$P(w_1^n) \approx \prod_{i=1}^{n-1} P(w_i | w_{i-1})$$

Example : Consider the following sentence : “*He never told her and she had never cared to ask* “ . The probability using bigram model can be computed as follows :

$P(\text{He never told her and she had never cared to ask.})$

$$\begin{aligned}
 &= P(\text{He} | \langle \text{nil} \rangle) \cdot P(\text{never} | \text{He}) \cdot P(\text{told} | \text{never}) \cdot P(\text{her} | \text{told}) \cdot P(\text{and} | \text{her}) \cdot P(\text{she} | \text{and}) \cdot P(\text{had} | \text{she}) \cdot \\
 &\quad P(\text{never} | \text{had}) \cdot P(\text{cared} | \text{never}) \cdot P(\text{to} | \text{cared}) \cdot P(\text{ask} | \text{to}) \cdot \\
 &= (207/67675) \cdot (3/512) \cdot (1/60) \cdot (10/31) \cdot (6/1137) \cdot (34/2353) \cdot (168/935) \cdot (6/1077) \cdot (1/60) \cdot (3/11) \cdot \\
 &\quad (3/2267)
 \end{aligned}$$

Each probability term is calculated by finding the number of occurrence of the specific bigram and dividing it by the frequency of the previous word.

Thus

$$P(\text{She/and}) = \frac{\text{(Number of Occurrence of the bigram and She)}}{\text{(Number of Occurrences of the word and )}}$$

**Smoothing :** The draw backs of N – grams is it banks heavily on the corpus which forms the basic training data. Any corpus is finite and there are bound to be many N – grams missing with it. As illustrated in the table There are numerous N – grams which should have had non zero probability but are assigned zero value instead.

**Table**      *Bigram counts (The number in the bracket indicates the probability.)*

	<i>He</i>	<i>never</i>	<i>told</i>	<i>her</i>	<i>and</i>	<i>she</i>	<i>had</i>	<i>never</i>	<i>cared</i>	<i>to</i>	<i>ask</i>
<i>He</i> [207]	0(0)	3(0.014)	0(0)	0(0)	4(0.019)	0(0)	114(0.55)	3(0.0144)	0(0)	2(0.01)	0(0)
<i>never</i> [60]	0(0)	0(0)	1(0.02)	0(0)	0(0)	0(0)	3(0.05)	0(0)	1(0.02)	0(0)	0(0)
<i>told</i> [31]	0(0)	0(0)	0(0)	10(0.323)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)
<i>her</i> [1137]	1(0.00008)	0(0)	0(0)	0(0)	6(0.005)	0(0)	1(0.00008)	0(0)	0(0)	19(0.0167)	0(0)
<i>and</i> [2353]	34(0.0144)	1(0.00004)	0(0)	26(0.011)	0(0)	34(0.014)	23(0.01)	1(0.00004)	0(0)	38(0.016)	0(0)
<i>she</i> [935]	0(0)	1(0.001)	1(0.001)	0(0)	3(0.003)	0(0)	168(0.18)	1(0.001)	2(0.002)	0(0)	0(0)
<i>had</i> [1077]	8(0.007)	6(0.006)	3(0.003)	0(0)	0(0)	9(0.008)	12(0.111)	6(0.006)	1(0.0001)	8(0.007)	0(0)
<i>never</i> [60]	0(0)	0(0)	1(0.02)	0(0)	0(0)	0(0)	3(0.05)	0(0)	1(0.02)	0(0)	0(0)
<i>cared</i> [11]	0(0)	0(0)	0(0)	0(0)	1(0.090)	0(0)	0(0)	0(0)	0(0)	3(0.273)	0(0)
<i>to</i> [2267]	0(0)	0(0)	0(0)	99(0.044)	1(0.00004)	0(0)	0(0)	0(0)	0(0)	0(0)	2(0.00008)
<i>ask</i> [12]	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)	0(0)

Smoothing is a process of assigning the non zero probability to circumvent this problem to some extent.

Two methods are available

1. **Add one Smoothing**
2. **Witten Bell Discounting .**

**1. Add one Smoothing :** Let  $P(w_i)$  be the normal unigram probability without smoothing (unsmoothed) and  $N$  be the total number of words in Corpus then

$$P(w_i) = \frac{c(w_i)}{\sum c(w_i)} = \frac{c(w_i)}{N}$$

This is simplest way of assigning non zero probabilities . Before calculating the probabilities , the count  $c$  of each distinct words within the corpus is incremented

by 1. Note that total of the counts have now increased by **D** the number of distinct types of words in the language.

The new probability of a word after add one smoothing can now be computed as

$$P_{\text{add}}(w_i) = \{ c(w_i) + 1 \} / (N + D)$$

## 2. Witten – Bell Discounting

The probability of **unseen N grams** could be looked upon as things we saw once.

The probability of an unseen N gram is given as

$$P^U = H / I (N+H)$$

H : Total number of the N – grams observed at first time .

N: Unique N grams types in the corpus

I : Total number of N – grams that have never occurred so far ( zero count)

Since the total probability has to be 1 this extra probability distributed amongst unseen N grams has to be discounted from other regions in the probability distribution.

The discount probability of the seen N grams to aid the generation of the extra probability for the unseen ones is given by

$$P_k^s = C_k / (N+H)$$

Where  $C_k$  is the non zero position count of the kth N gram.

# **Chapter2 : LEARNING**

## **1. What is Learning ?**

**Answer :** Learning is the process by which a system improves its performance from experience or environment. It is the art of acquiring new or modifying existing knowledge and skills. It is a change in behavior. If a computer is able to learn new things and adapt to new situations then we say that the computer has been equipped with "*learning programs*". Learning is an area of artificial intelligence focusses on the process of *self improvement*. Computers do the learning process through the following steps :

- 1. Data Acquisition of new Knowledge**
- 2. Problem Solving .**

The different types of Learning are :

1. Rote Learning
2. Learning by taking Advice
3. Learning in Problem Solving
4. Learning from Examples
5. Explanation based Learning
6. Discovery
7. Analogy
8. Formal Learning Theory
9. Neural Nets and
10. Genetic Learning

## **2. Explain Rote Learning .**

**Answer :** **Rote learning is a learning technique which focusses on storing or recording data.** It is also called memorization because the [knowledge](#), without any modification is, simply copied into the knowledge base. As computed values

are stored, this technique can save a significant amount of time. In case of **data caching** we store computed values so that we do not have to recompute them later. When computation is more expensive than recall, this strategy can save a significant amount of time. Caching has been used in AI programs to improve the performance. **Such caching is known as Rote Learning** .

**Example :** Checkers/tic-tac toe -playing [program](#), uses this technique to learn the board positions. No need to re-evaluate a particular sub-tree at a later stage.

### Capabilities of Rote Learning

1. Organized Storage : There must be organized stored of information . In order to use the stored values, the board position should be organized in such a way that *the process of retrieval is the fastest*. This is done by indexing the board position by the number of pieces on the board.
2. Generalization : To keep the number of stored objects at a manageable level , some kind of generalization is necessary.
3. Direction : The program should try to *find the required stored value* intelligently for a given hard position. So we have to focus the attention to a single processing direction.

## 3. Explain Learning by Taking Advice

**Answer :** This is a simple form of learning. Suppose a programmer writes a set of instructions to instruct the computer what to do, ***the programmer is a teacher*** and ***the computer is a student***. Once learned (i.e. programmed), the system will be in a position to do new things. The advice may come from many sources: human experts, internet ,etc. This type ***of learning requires more inference*** than rote learning. The knowledge must be transformed into an ***operational form before stored in*** the knowledge base. **FOO (First Operational Operationaliser)** , for example, is a learning system which is used to learn the game of Hearts. It *converts the advice which is in the form of principles, problems, and methods* into effective executable (LISP) procedures (or knowledge). The knowledge will be ready to use.

## 4. What is Learning Problem Solving? Explain the different types of Learning in Problem Solving.

**Answer :** Learning through problem solving **from own experience** – without an instructor/advisor. Does not involve an increase in knowledge, just the methods in using the knowledge.

**The Utility Problem:** Learnt rules are good in directing the problem solving process, **but** it incurs a cost (utility) because the problem solver needs to store and consult those rules. Can be partially overcome by a utility measure to keep track of *how useful the learnt rules* are, and deleting them when necessary.

### Types of Learning in Problem Solving

#### 1. Learning by Parameter Adjustment

- Use outcomes to adjust the weights for factors in an evaluation function
- Considerations:
  - What are the initial weights?
  - When does certain weights increase?
  - When do they decrease?

#### 2. Learning with Macro-operators

- Rote Learning a sequence of operations found to be useful during problem solving.

#### 3. Learning by Chunking

- Rote learning in the context of a Production System
- Rules which are useful and always fired together are chunked to form one large production

**4.1: Learning by Parameter Adjustment :** Many programs rely on an evaluation procedure to summarize the state of search *etc.* Game playing programs provide many examples of this. However, many programs have a *static evaluation function*. In learning a *slight modification of the formulation of the evaluation of the problem is required*. Here the problem has an evaluation function that is represented as a polynomial of the form such as:

$$E(n) = c_1t_1 + c_2t_2 + c_3t_3 + \text{-----}$$

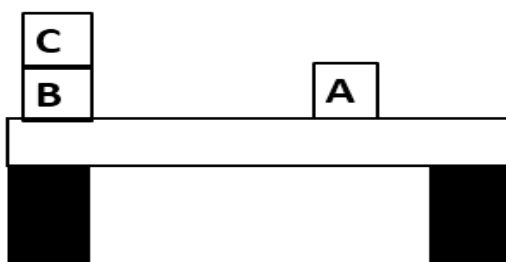
The  $t$  terms a value of features and the  $c$  terms are weights. In designing programs, it is often difficult to decide on the exact value to give each weight initially. So the basic idea of *parameter adjustment* is to:

Start with some estimate of the correct weight settings. Modify the weight in the program based on accumulated experiences. *Features that appear to be good predictors will have their weights increased* and bad ones will be decreased. Samuel's Checkers programs employed 16 such features at any one time chosen from a pool of 38.

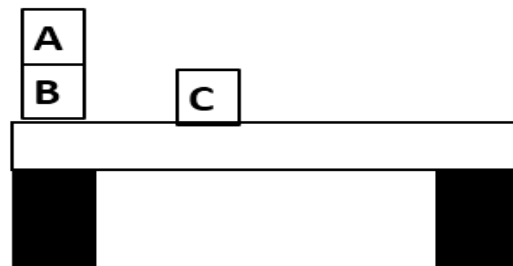
**4.2 Learning with Macro-operators : Sequence of actions** can be collectively called as **macro-operators**. Example: `START_CAR = { Open-door, Sitting_Down, Closing the Door, Adjusting the Mirror, Inserting Key, Turning the Key, .. }`.

- MACRO Operators were used in the early problem-solving system called STRIPS. It is a planning algorithm in which after each problem-solving episode the learning component takes the computed plan and stores it as a macro operator or MACROP.
- A MACROP is just like a regular operator except that it consist of sequence of actions not just a single one.

**Example :** Suppose we are given an initial blocks world situation in which `ON(C,B)` and `ON(A,Table)` are both true . STRIPS can achieve the goal `ON(A,B)` by devising a plan with goal `ON(A,B)` by devising a plan with the four steps `UNSTACK(C,B)` , `PUTDOWN(C)` , `PICKUP(A)` and `STACK(A,B)`.



**Pre Condition**



**Post Condition**

### MACROP : ON(A,B)

Precondition : ON(C,B) ,ON(A,Table)

Post Condition : ON(C,Table), ON(A,B)

STEPS:

1. UNSTACK(C,B)
2. PUTDOWN(C)
3. PICKUP(A)
4. STACK(A,B)

### 4.3 Learning by Chunking

*Chunking* involves similar ideas to Macro Operators and originates from psychological ideas on memory and problem solving. It is a universal learning procedure. **Chunk is larger set of production rules that does work of an entire sequence of smaller ones.** Chunks are generalized before they are stored as in MACROS .

**EXAMPLE :** SOAR exploits chunking so that its performance can increase with experience. SOAR solves problem by firing productive which are stored in long term memory. Some of these firings turn out to be more useful than the others.

***When SOAR detects a useful sequence of production firings it creates a chunk*** which is essentially a large production that does the work of an entire sequence of smaller ones. SOAR is used in learning how play the 8-puzzle game.

**UTILITY PROBLEMS :** PRODIGY[Minton etal ,1989] also acquires control knowledge automatically .It employs learning mechanism like explanation based learning (EBL). It can ***examine a trace of its own problem solving behavior and try to explain why certain path failed.***

**Problems :** The learned control rules can take up large amounts of memory. Search problem takes more time to consider each rule at each step during problem solving. PRODIGY maintains a utility measure for each control rule . IF a production rule has a negative utility it is discarded (or forgotten) .If not it is placed in long term memory and will be used for problem solving in future .



5. What is learning from Examples ? Explain the different types of learning from examples .

**Answer: Learning from Examples (Induction):** Classification: the process of assigning, to a particular input, the name of a class to which it belongs. It is a recognition task. Classification is embedded inside another operation. For example, consider the following production rule

*If: the current Goal is to get from place A to place B and there is a WALL separating the two places*

*Then : Look for a Doorway in the wall and go through it.*

To use this rule the matching routine must be able to identify an object as a wall. The system must be able to recognize a doorway.

The different classes for classification can be done in a variety of ways, Isolating the set of features that are relevant to the task domain. Define each class by a weighted sum of values of these features using functions like:  $c_1t_1+c_2t_2+c_3t_3+....$ .

Where t corresponds to value of a relevant parameter and c represents the weight to be attached to the corresponding t. Negative Weights can be used to indicate features whose presence usually constitutes negative evidence for a given class . Define each class as a structure composed of those features.

**For example:** IF the task is to identify animals the body of each type of animal can be stored as a structure with various features representing such things as color, length of neck and feathers. The techniques of construction of class definition which can produce a classification program that can evolve its own class definitions is called concept learning or induction.

There are three such techniques:

**1. Winston's Learning Program**

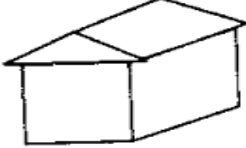
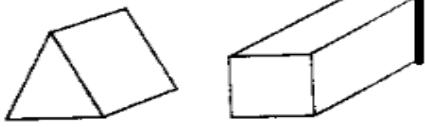
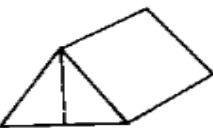
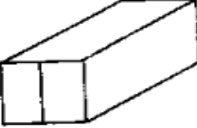
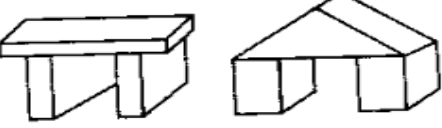
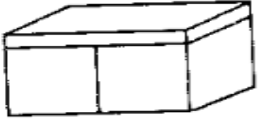
**2. Version Spaces**

**3. Decision trees**

### **5.1 Winston's Learning Program**

Winston [1975] describes an early structural concept learning program in a simple blocks world domain. The goal is to construct structural representation (like semantic nets) of the definitions of concepts in the blocks domain. Concepts such as a house - brick (rectangular block) with a wedge (triangular block) suitably placed on top of it, tent - 2 wedges touching side by side, or an arch - two non-touching

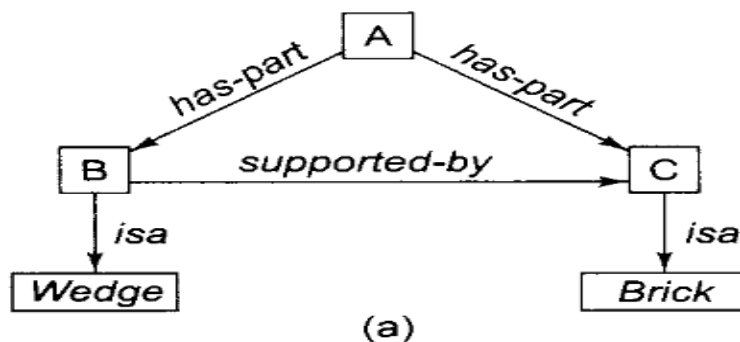
bricks supporting a third wedge or brick, were learned. The idea of *near miss* objects -- similar to actual instances was introduced. Input was a line drawing of a blocks world structure. Input processed to produce a semantic net representation of the structural description of the object

	Concept	Near Miss
House		
Tent		
Arch		

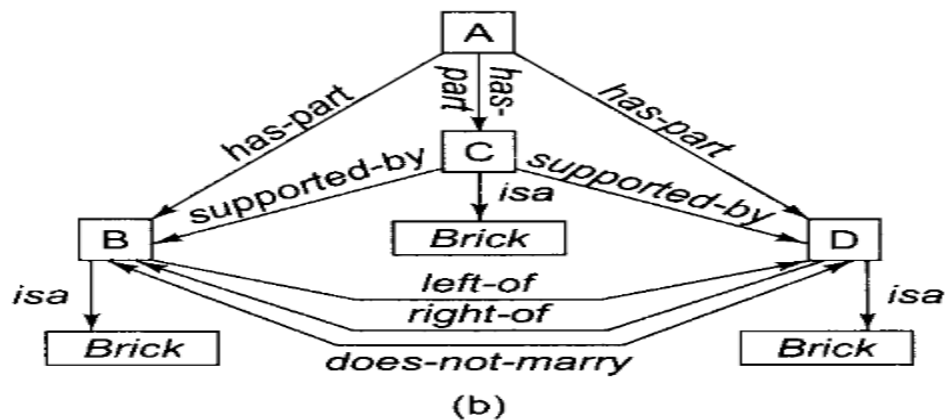
**Fig. 17.2** *Some Blocks World Concepts*

**Construction of Structural representation:** Links in network include *left-of*, *right-of*, *does-not-marry*, *supported-by*, *has-part*, and *isa*. The *marry* relation is important -- two objects with a common touching edge are said to marry. Marrying is assumed unless *does-not-marry* stated.

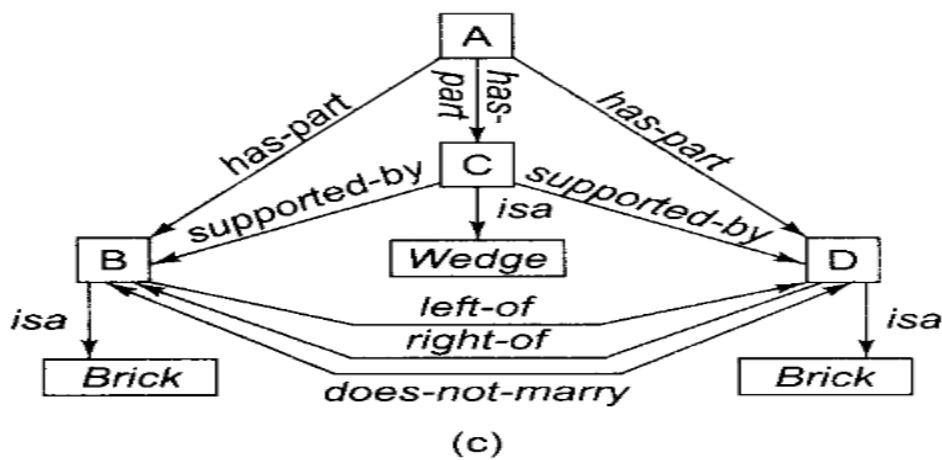
### Structural Description of House using Semantic net



## Structural Description of Tent using Semantic net



## Structural Description of Arch using Semantic net



There are three basic steps to the problem of concept formulation:

1. Select one known instance of the concept. Call this the *concept definition*.
  2. Examine definitions of other known instances of the concept. *Generalize* the definition to include them.
  3. Examine descriptions of *near misses*. *Restrict* the definition to *exclude* these.
- Both steps 2 and 3 rely on comparison and both similarities and differences need to be identified.**

**5.2 : Version Spaces :** Version Space is the largest collection of description that is consistent with all the training examples . The representation language can be used to represent the collection of all discrete value of slots of corresponding frame.

For Example Consider a Frame representing an individual Car :

Car023

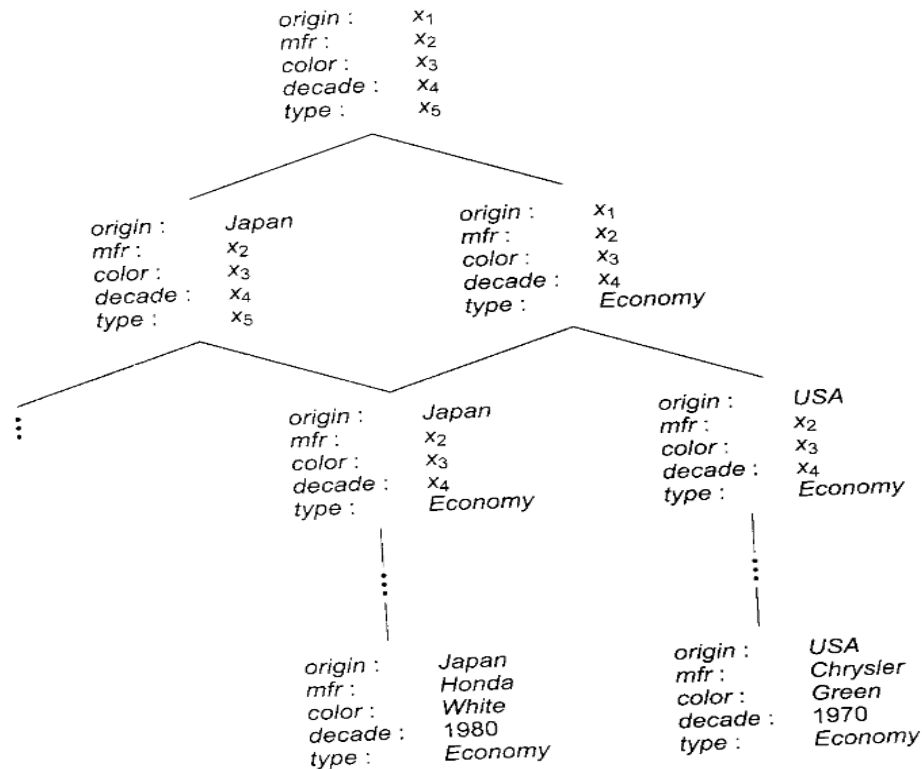
- Origin : Japan
- Manufacturer : Honda
- Color : Blue
- Decade : 1970
- Type : Economy

Each slot may contain the discrete values as given by representation language is as shown below :

Origin	€	{ Japan, USA, Britain, Germany, Italy}
Manufacturer	€	{ Honda, Toyota, Ford, Chrysler, Jaguar, BMW, Fiat}
Color	€	{ Blue , Green, Red, White }
Decade	€	{ 1950,1960,1970,1980,1990,2000}
Type	€	{ Economy, Luxury , Sports}

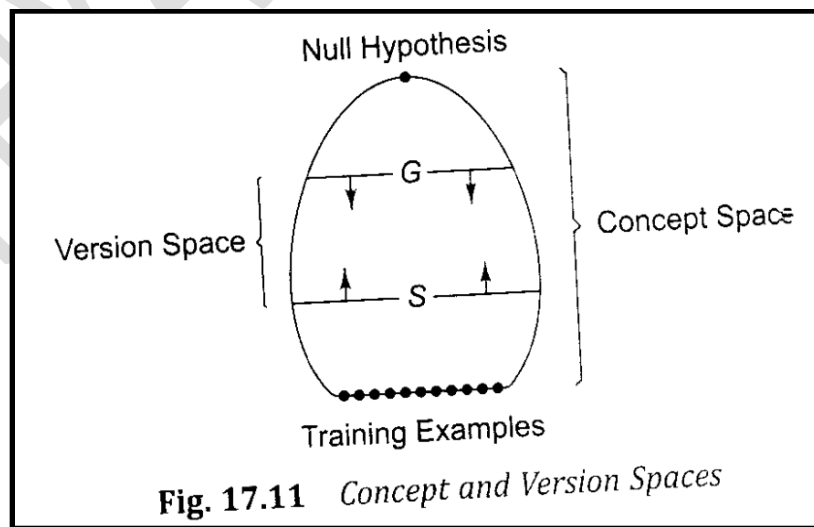
The concept “ Japanese Economy Car ” is as illustrated below :

Origin	Japan
Manufacturer	X1
Color	X2
Decade	X3
Type	Economy



**Fig. 17.10** Partial Ordering of Concepts Specified by the Representation Language

In the above partial ordering of concepts specified by the representation language, some descriptions are more general, and some description are more specific. The entire partial ordering is called concept space and can be depicted as given below:



**Fig. 17.11** Concept and Version Spaces

Version Space is a set of description. The number of descriptions in the *concept space is exponential* in the number of features and values. So enumerating them is prohibitive. But the version space has concise representation . It consists of two subsets of the concept space . *One subset called G contain most general description , the subset S contains the most specific description of training examples.* The version space is the set of all description *that lie between some element of G and some element of S in the partial order of the concept space*

origin: Japan mfr: Honda color: Blue decade: 1980 type: Economy (+)	origin: Japan mfr: Toyota color: Green decade: 1970 type: Sports (-)	origin: Japan mfr: Toyota color: Blue decade: 1990 type: Economy (+)
origin: USA mfr: Chrysler color: Red decade: 1980 type: Economy (-)	origin: Japan mfr: Honda color: White decade: 1980 type: Economy (+)	

**Fig. 17.12** Positive and Negative Examples of the Concept "Japanese economy car"

$$G = \{(x_1, x_2, x_3, x_4, x_5)\}$$

$$S = \{(Japan, Honda, Blue, 1980, Economy)\}$$

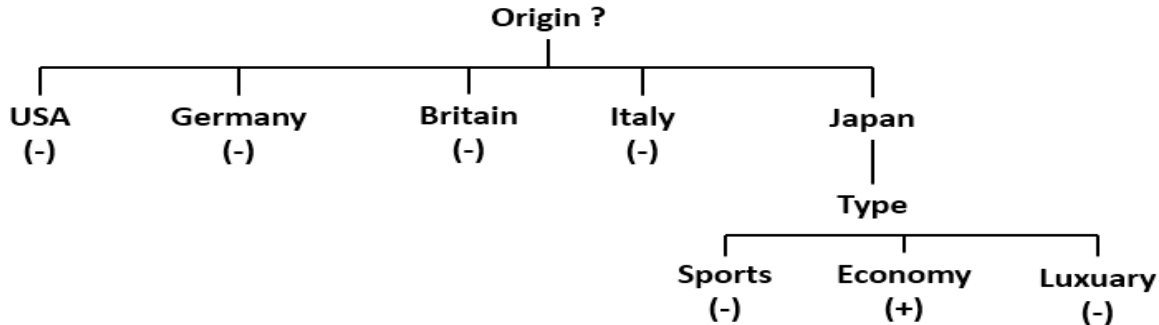
### Candidate Elimination Algorithm

- Given: A representation language and a set of positive and negative examples expressed in that language.
- Compute: A concept description that is consistent with all the positive examples and none of the negative examples.
  1. Initialize G to contain one element: the null description
  2. Initialize S to contain one element: the first positive example
  3. Accept a new training example
    - If it is positive example, first remove from G and Update the Set S to contain the most specific description

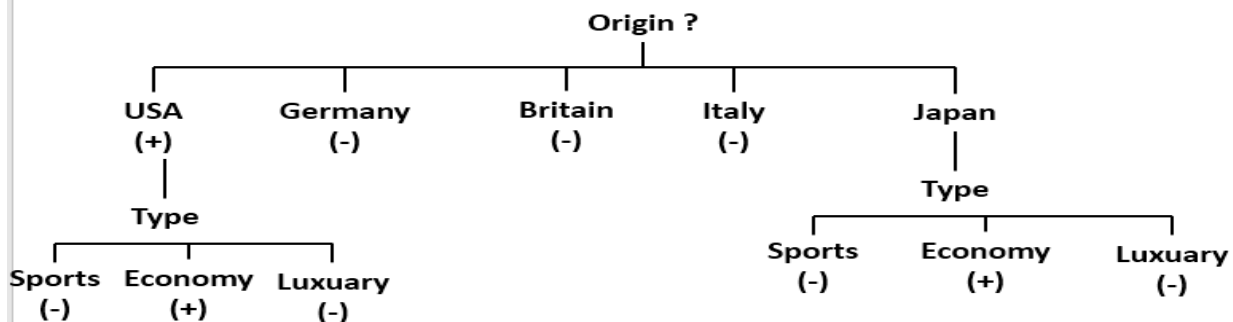
- If it is negative example first remove from S any descriptions that cover the example and then update the G to contain more generic expression.
4. If S and G are both singleton sets , then they are identical , output the value and halt. IF both are different and inconsistent then output the result and halt. Otherwise go step3.

**5.3 Decision Trees :** Quinlan in his (Iterative Dichotomiser 3) ID3 system (1986) introduced the idea of decision trees. The algorithm will generate a decision tree from a data set. ID3 is a program that can build trees automatically from given positive and negative instances. Basically each leaf of a *decision tree* asserts a positive or negative concept. *To classify a particular input we start at the top and follow assertions down until we reach a leaf where the classification is stored .*

#### Decision Tree for the concept : Japanese Economy Car



#### Decision Tree for the concept: American or Japanese Economy Car



**Building decision trees :** ID3 uses *an iterative method to build decision trees*. Simple trees preferred over complex ones , because simple trees are more accurate classifiers of future inputs .

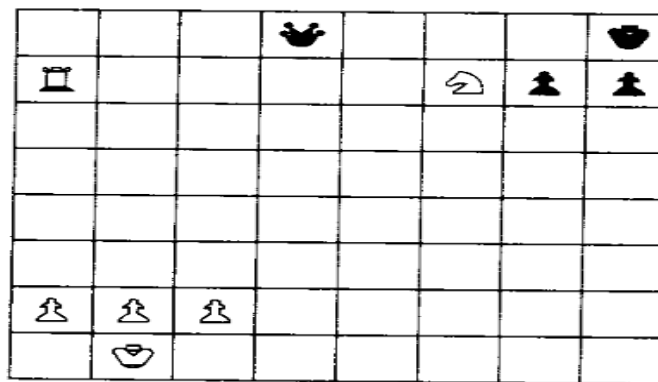
- It begins by choosing a **random subset of the training examples. The subset is called the *window*** .
- Other training examples used to test tree.
- If all examples classified correctly stop.
- Otherwise add a number of training examples to *window* and start again.

Empirical evidence indicates that the iterative strategy is more effective than considering the whole training set at once.

## 6. Describe the Explanation based Learning.

**Answer :** An Explanation-based Learning (EBL ) system accepts an example (i.e. a training example) and explains what it learns from the example. The EBL system takes only the relevant aspects of the training. This explanation is translated into particular form that a problem-solving program can understand. The explanation is generalized so that it can be used to solve other problems. Humans appear to learn quite a lot from one example.

**Basic idea:** Use results from one examples problem solving effort next time around. For example in the figure given below , the black player will learn about the **FORK Trap** in which white Knight attacks both the black queen and king.



**Fig. 17.14** *A Fork Position in Chess*



## In general, the inputs to EBL programs are:

1. **A Training Example** – *What the learning program sees in the world* eg. The Japanese Economy Car
2. **A Goal Concept** – A high level description of *what the program is supposed to learn*
3. **An Operationally Criterion** – A description of *which concepts are usable*.
4. **A Domain Theory (or Knowledge Base)** – A *set of rules* that describe relationships between objects and actions in a domain.

From this EBL computes a *generalization of the training example that is sufficient not only to describe the goal concept but also satisfies the operational criterion*.

**Explanation based Generalization (EBG):** EBG is an algorithm for EBL. It has two steps:

1. **Explain:** -- the domain theory is used to prune away all unimportant aspects of the training example with respect to the goal concept.
2. **Generalize:** -- the explanation is generalized as far possible while still describing the goal concept

Example : Generalizing from Single example of a Cup

### Training Example :

Owner (Object23,John)  $\wedge$  has\_part(Object1,Concavity12)  $\wedge$  is(object23,light)  
 $\wedge$  color(object23,Brown)  $\wedge$ .....

### Domain Knowledge :

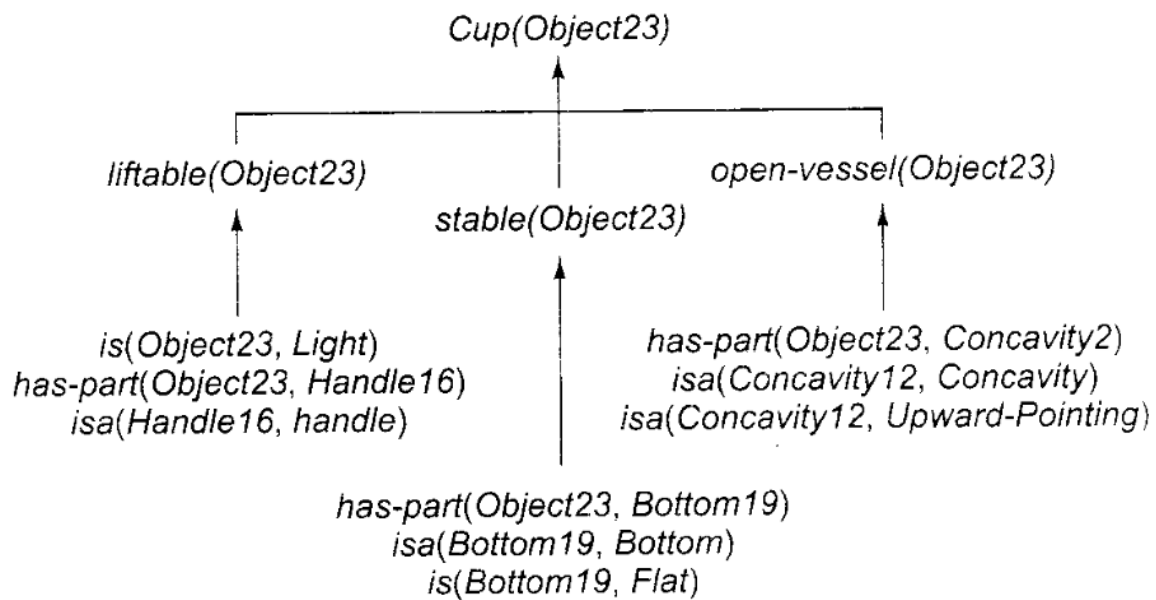
is(x,Light)  $\wedge$  has\_Part(x,y)  $\wedge$  isa(y,Handle)  $\rightarrow$  liftable(x)  
has\_part(x,y)  $\wedge$  isa(y,bottom)  $\wedge$  is(y,Flat)  $\rightarrow$  stable(x)  
has\_part(x,y)  $\wedge$  isa(y,concavity)  $\wedge$  is(y,Upward\_Pointing)  $\rightarrow$  open\_Vessel(x)

- Goal Concept : Cup

X is a cup if x is liftable , stable and open\_vessel

- Operationality Criterion :

Concept definition must be expressed in purely structural terms (eg: Light , Flat ,etc).



**Fig. 17.15** *An Explanation*

## 7. Describe the Discovery Learning and its Types

**Answer :** Learning is the process by which one entity acquires knowledge. Usually that knowledge is already possessed by some number of other entities who may serve as a teacher.

**Discovery** is a restricted form of learning in which one entity acquires knowledge without the help of teacher. There are three types of automated discovery systems :

1. **AM: Theory Driven Discovery**
2. **BACON : Data Driven Discovery**
3. **Clustering**

**7.1 AM: Theory Driven Discovery** : AM is a program based on the concepts of set theory that discovers a good deal of **standard number theory**. It uses a frame system to represent **mathematical concepts**. One of the major activities

of AM is to create new concepts and **fill in their slots**. An example of **AM concept** is shown in the figure below :

- **Name** : Prime Numbers
- **Definition** :
  - **Origin** : Number of Divisors  $(x) = 2$
  - **Predicate Calculus** :  $\text{Prime}(x) \leftrightarrow \text{ForAll } Z, Z|x \Rightarrow Z=1 \text{ or } Z=x$
  - **Iterative for  $(x>1)$**  : For  $i$  from 2 to  $\text{sqrt}(x)$  ,  $i$  does not divides  $x$ .
- **Examples** : 2,3,5,7,11,13,17
- **Boundary** : 2,3
- **Boundary Failures** : 0,1
- **Failures**: 12
- **Generalization** : Number with even number of divisors
- **Specializations** : odd primes, prime pairs ,prime uniquely adables.

## 7.2 BACON : Data Driven Discovery

Many discoveries are made from **observing data obtained from the world and making sense of it** -- *E.g.* Astrophysics - discovery of planets, Quantum mechanics - discovery of sub-atomic particles. BACON is an attempt to provide such AI systems. **BACON system outline is as follows** :

- **Starts with a set of variables for a problem.**
- **Values from experimental data from the problem are inputted.**
- **BACON holds some constant and attempts to notice trends in the data.**
- **Inferences are made.**

BACON has also been applied to **Kepler's 3rd law, Ohm's law, conservation of momentum and Joule's law.**

### Example : Ideal Gas Law

- It started with four variables  **$p$  - gas pressure,  $V$  -- gas volume,  $n$  -- molar mass of gas,  $T$  -- gas temperature.**
- First it holds the variable  $n$  and  $T$  constant performing experiments at different pressures  $p_1, p_2$  and  $p_3$ .

- BACON notices that as the pressure  $p$  increases the volume  $V$  decreases
- It creates a theoretical term  $pV$ . This term remains constant for same  $T$ .
- It tries an experiment with different value of  $T$ .  $PV$  also varies now.
- BACON creates a new term  $pV/T$  and varies the term  $n$ .
- For all values of  $n, P, V$  and  $T$ ,  $pV/nT=8.32$ . This is the ideal gas law.

$n$	$T$	$p$	$V$	$pV$	$pV/T$	$pV/nT$
1	300	100	24.96			
1	300	200	12.48			
1	300	300	8.32	2496		
1	310			2579.2		
1	320			2662.4	8.32	
2	320				16.64	
3	320				24.96	8.32

**Fig. 17.18** *BACON Discovering the Ideal Gas Law*

**Clustering:** Clustering is a very similar to **induction**. In inductive learning a program learns to classify objects based on the **labeling provided by a teacher**. **In clustering no class labelings are provided**. The program must discover for itself the natural classes that exist for the objects, in addition to a method for classifying instances.

**Example:** AUTOCLASS is one program that accepts a number of training cases and hypothesizes a set of classes.

## 8. What is Analogy ? Explain the different types of Analogy ?

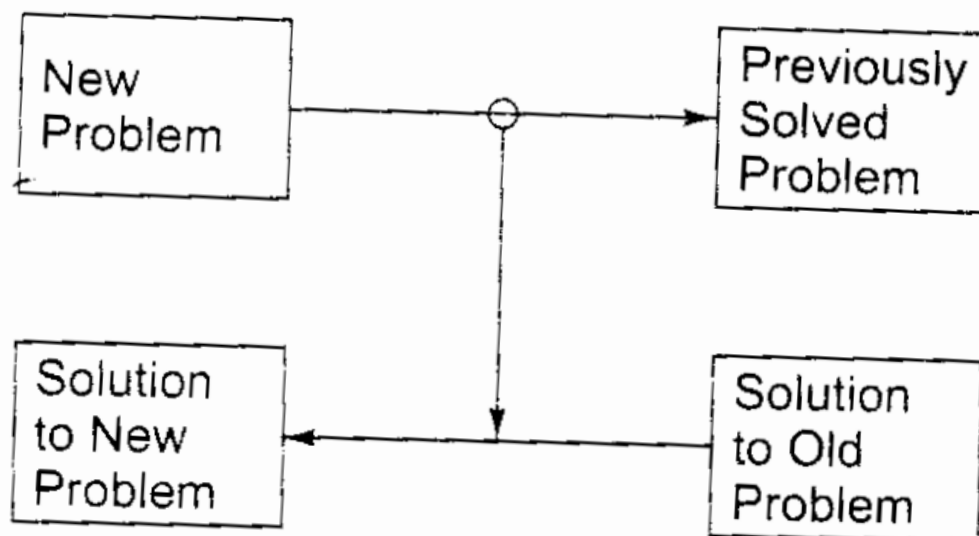
**Answer :** Analogy is a powerful inference tool . Our language and reasoning used in daily life are filled with analogies and metaphors as illustrated below :

- Last month the stock market was a roller coaster
- Bill is like a fire engine
- Problems in electromagnetism are just like problems in fluid flow .

Underlying each of these examples is a complicated mapping between what appear to be dissimilar concepts. Analogical reasoning is an important factor in learning by advice taking and also important in problem solving . Humans often solve problems by making analogies to things they already understand how to do. Two methods of analogical problem solving that have been studied in **AI are transformational and derivational analogy.**

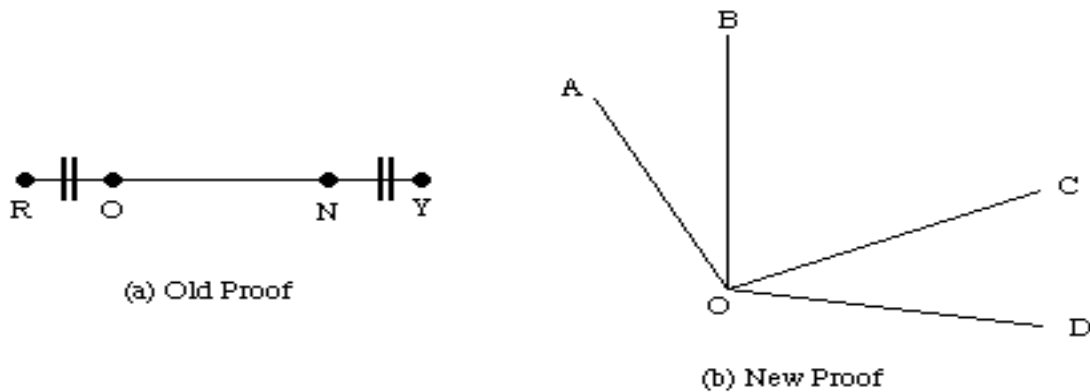
### 8.1 Transformational Analogy

Suppose you are asked to prove a theorem **in plane geometry**. You might look for a previous theorem that is very similar and “copy” its proof , making substitutions when necessary. The idea is to transform a solution to a previous problem into a solution for the current problem.



**Fig. 17.19** *Transformational Analogy*

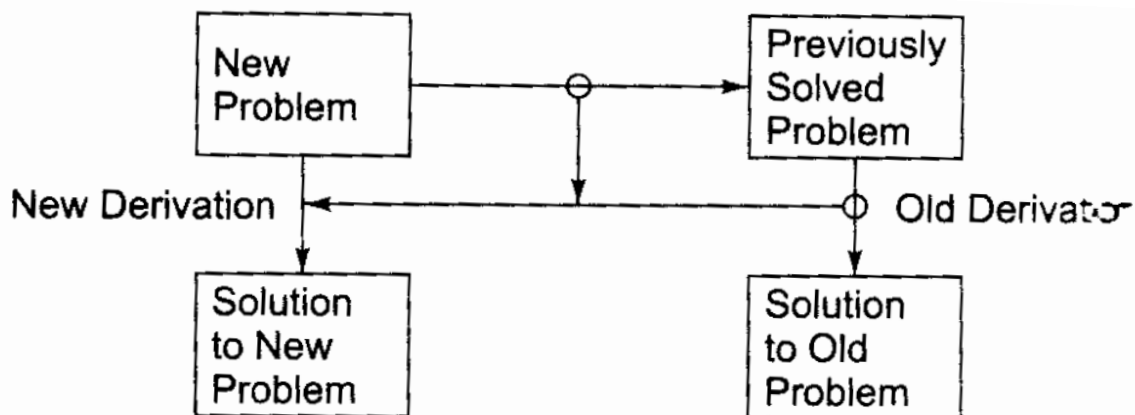
**Example :**



### Transformational Analogy Example

- We know that lines  $RO = NY$  and angles  $AOB = COD$
- We have seen that  $RO + ON = ON + NY$  - additive rule.
- So we can say that angles  $AOB + BOC = BOC + COD$
- So by a transitive rule line  $RN = OY$
- So similarly angle  $AOC = BOD$

**8.2 Derivational Analogy** : The transformational analogy does not look at how the old problem was solved : it only looks at the final solution. The *history* of the problem solution - the steps involved - are often relevant. The detailed history of a problem solving episode is called its derivation. *Analogical reasoning that takes these histories into account is called derivational analogy.*



**Fig. 17.21** Derivational Analogy

## 9. Explain Formal Learning Theory

**Answer :** Valiant[1984] describes a “theory of the learnable” which classifies problems by how difficult they are to learn. Formally a device learns a concept if it can given positive and negative example, produce an algorithm that will classify future examples correctly with probability  $1/h$ . The complexity of learning a concept is a function of three factors :

- The error tolerance ( $h$ )
- The number of binary features present in the example ( $t$ )
- The size of the rule necessary to make the discrimination ( $f$ )

If the number of training examples required is polynomial in  $h, t$  and  $f$  and then the concept is said to be learnable .

**Example :** From the list of positive and negative examples of elephants shown in Figure we want to induce the description “gray, mammal and large “

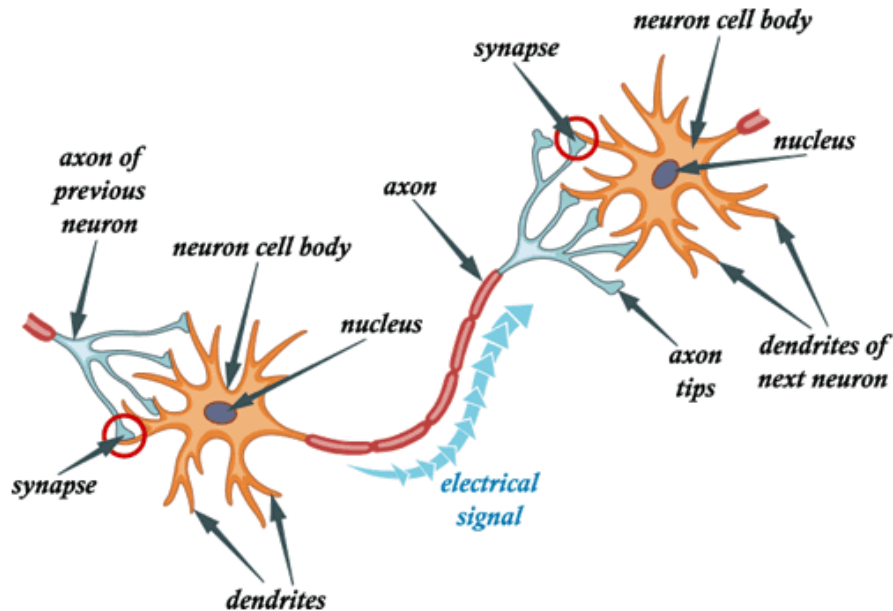
gray?	mammal?	large?	vegetarian?	wild?		
+	+	+	+	+	+	(Elephant)
+	+	+	-	+	+	(Elephant)
+	+	-	+	+	-	(Mouse)
-	+	+	+	+	-	(Giraffe)
+	-	+	-	+	-	(Dinosaur)
+	+	+	+	-	+	(Elephant)

**Fig. 17.22** Six Positive and Negative Examples of the Concept Elephant

10. Write a Short note on a. *Neural Networks* and b. *Genetic Algorithms*.

Answer :

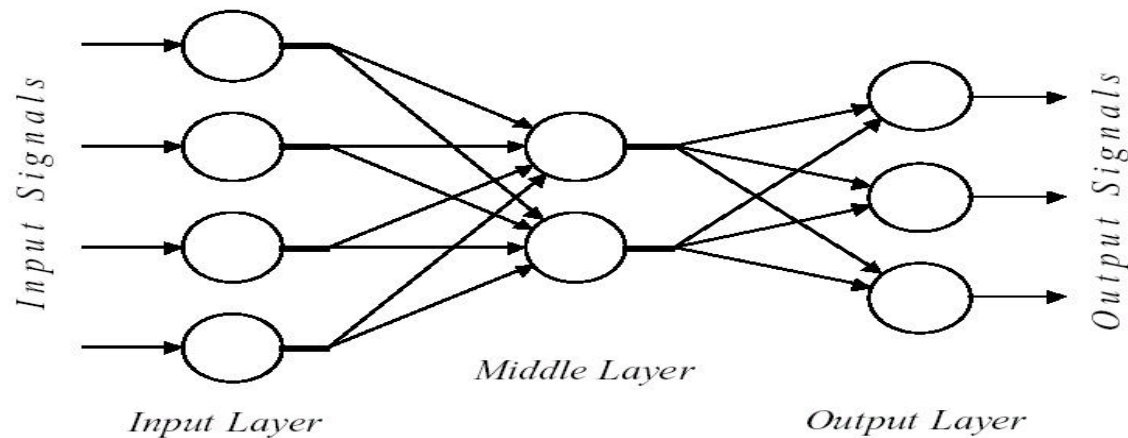
**NEURAL NETS LEARNING** : This type of learning mimics the animal (human) brain learning at a neural level. Collections of idealized neurons were presented with stimuli and produced into changing their behavior. Learning is a fundamental and essential characteristic of biological neural networks. A **neural network** can be defined as a model of reasoning based on the human brain. The brain consists of a densely interconnected set of nerve cells, or basic information-processing units, called **neurons**. The human brain incorporates nearly 10 billion neurons and 60 trillion connections, *synapses*, between them. Our brain can be considered as a highly complex, non-linear and parallel information-processing system. Information is stored and processed in a neural network simultaneously throughout the whole network, rather than at specific locations.



**Artificial Neural Networks** : An Artificial Neural Network (ANN) consists of a number of very simple processors, also called *neurons*, which are analogous to the biological neuron in the brain. The neurons are connected by weighted links passing signals from one neuron to another. The output signal is transmitted through the neuron's outgoing connection. The outgoing connection splits into a number of branches that transmit the same signal. The outgoing branches terminate at the incoming connections of other neurons in the network.



### Architecture of a typical artificial neural network:



b. **Genetic Learning** : Learning mechanisms inspired by evolution are called genetic algorithms . The genetic algorithm works as follows :

- Firstly, define the initial population of entities.
- Defined a function to classify whether a entity is good or bad.
- Then selected good entities for generating new generation .
- And finally, new generation replaces the bad generation from the population and this process repeats.

## 11.. Differentiate between Supervised Learning and Unsupervised Learning

Answer :

**Supervised Learning** : Supervised learning is so named because the data scientist acts as a guide to teach the algorithm what conclusions it should come up with. It's similar to the way a child might learn arithmetic from a teacher. For example, a classification algorithm will learn to identify animals after being trained on a dataset of images that are properly labeled with the species of the animal and some identifying characteristics.

**Unsupervised Learning :** Unsupervised machine learning is more closely aligned with what some call true artificial intelligence — the idea that a computer can learn to identify complex processes and patterns without a human to provide guidance along the way.

Some examples of unsupervised machine learning algorithms include [k-means clustering](#), principal and independent component analysis, and association rules.

THYAGU GOWDA

## Chapter 3: Expert Systems

**1. What are Expert Systems? Explain all the following five important issues of Expert System with example.**

- **a. Representing and Using Domain Knowledge**
- **b. Reasoning with knowledge**
- **c. Expert System Shells**
- **d. Explanation**
- **e. Knowledge Acquisition**

**Answer :**

**Expert Systems are AI Programs that has expert level knowledge** about a particular domain and which can solve problems that are **normally solved by human experts (such as doctor, engineer or lawyer)**. Expert system knows how to use its [knowledge](#) to respond properly. Domain refers to the area within which the task is being performed. Ideally the expert systems should substitute a human [expert](#).

**Issues of Expert System :**

- a. Representing and Using Domain Knowledge (**Knowledge Base**)
- b. Reasoning with Knowledge (**Inference Engine**)
- c. **Expert System Shells**
- d. **Explanation**
- e. **Knowledge Acquisition**

**a. Representing and Using Domain Knowledge**

The most widely used way or representing domain knowledge in expert system is a ***set of production rules*** which are often coupled with a ***frame system that defines the objects*** that occur in the rules. Production rules can be expressed in IF-THEN rules :

IF you are hungry THEN eat

or

Hungry => Eat

**Examples of Rules based Expert Systems :** R1/XCON,MYCIN,PROSPECTOR ,  
DESIGN ADVISOR .

**a.1 R1/XCON (Computer Components Expert) :** The R1 (internally called XCON,) program was a production-rule-based system to assist in the ordering of DEC [VAX](#) computer systems by automatically selecting the computer system components based on the customer's requirements

### R1 Example Rule

DISTRIBUTE-MB-DEVICES-3

IF: the most current active context is distributing massbus devices

& there is a single port disk drive that has not been assigned to a massbus

& there are no unassigned dual port disk drives

& the number of devices that each massbus should support is known

& there is a massbus that has been assigned at least one disk drive and that should support additional disk drives

& the type of cable needed to connect the disk drive to the previous device on the disk drive is known

THEN: assign the disk drive to the massbus

### a.2 MYCIN (Medical Expert)

This expert system was the recommended therapy for **blood /meningitis** infections and the diagnosis normally involved growing cultures of the infecting organisms within 48 hours .

MYCIN is a goal directed system that uses a basic backward chaining technique with rules written in LISP.

### Example Rule of MYCIN

if

1 the infection is primary bacteremia, and

2 the site of the culture is one of the sterilesites, and

3 the suspected portal of entry of the organism is the gastrointestinal tract

**then**

there is suggestive evidence (0.7) that the identity of the organism is Bacteroides.

### a.3 Prospector (Mining Expert)

- PROSPECTOR (**Mining Expert**) is an expert system designed to geologists for decision-making problems in mineral exploration like identify sites for drilling or mining.
- It **aids geologists** in evaluating the favorability of an exploration site or region for occurrences of ore deposits of particular types. Once a site has been identified, PROSPECTOR can also be used for drilling-site selection.
- It makes use of **Bayesian Probability**.

#### Example Prospector Rule

If

Magnetite in disseminated form is present

Then :

**(2,-4)** there is favorable mineralization and texture for the propylitic stage

### a.4 DESIGN ADVISOR

Is a expert system that is used to design chips . It gives advice to the chip designer who decides to accept or reject the advice .

Sample Rule :

**If** : The sequential level count of ELEMENT is greater than 2, **UNLESS** the signal of ELEMENT is resetable

**Then** : Critique for Poor resetability

b. **Reasoning With Knowledge** : Expert Systems exploit many of the representation and reasoning mechanisms . Reasoning Programs are written as rule based

systems. Reasoning Techniques used are : **Forward Chaining** , **Backward Chaining** or **combination of the two**.

### Forward Chaining

**Forward Chaining:** Conclude from "A" and "A implies B" to "B".

A  
A -> B  
B

---

#### Example:

It is raining.  
If it is raining, the street is wet.  
The street is wet.

---

### Backward Chaining

**Backward Chaining:** Conclude from "B" and "A implies B" to "A".

B  
A -> B  
A

---

#### Example:

The street is wet.  
If it is raining, the street is wet.  
It is raining.

## Representation and Reasoning mechanism used in Expert Systems

Expert System	Representation	Inference Mechanism
MYCIN	Rules in LISP	Backward Reasoning/Chaining
PROSPECTOR	Rules and Semantic net	Forward chaining and Bayesian Reasoning
R1/XCON	Rules	Forward Chaining ; subtasking and constraint satisfaction
Design Advisor	Rules	JTMS

c. **Expert System Shells** : Expert System Shells are the interpreters that can be used to construct new expert systems by adding new knowledge corresponding to the new problem domain . Collection of software packages & tools to design, develop, implement, and maintain expert systems . Expert Systems are constructed as a set of declarative representations(rules) combined with an interpreter for those representations. The interpreters can be separated from the domain specific knowledge to construct a new expert systems. Shells serves as the basis for many of the expert systems. Shells typically support rules , frames , truth maintenance Systems and variety of other reasoning mechanisms. Shells provide an easy to use interface between an expert system that is written with the shell and a larger programming environment. **Example** : EMYCIN is a shell which is derived from MYCIN.

d. **Explanation** : In order for an expert system to be an effective tool , people must be able to interact wit it easily. To facilitate this interaction , the expert system must have the following two capabilities in addition to the ability to perform its underlying task :

Explain its reasoning . People must be convinced of the accuracy of the reasoning process. The reasoning process must be in understandable steps and enough meta knowledge must be available for the explanation.

Acquire new Knowledge and modifications of old knowledge.

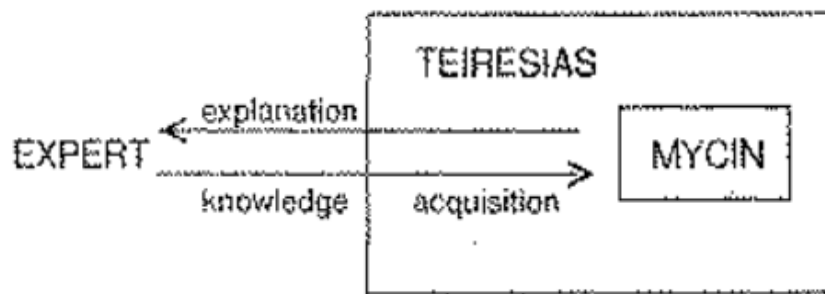
**Example :** TEIRESIAS was the first program to support explanation and knowledge acquisition. It is a front end for the MYCIN system.

Teiresias is designed to function as a bridge between the expert and Mycin ,or more generally, between the expert and the program he is trying to "educate."

This gives the physician (who presumably knows little or no programming) two important capabilities.

(i) *can interact with Mycin to find out why it is performing as it is (information flow from right to left, explanation), and*

(ii) *can "educate" Mycin by editing or adding to its knowledge base (in formation flow from left to right, knowledge acquisition).*



### Teiresias Explanation Example

**Question Asked by MYCIN :** In this series, how many of the cultures were positive for the e.coli (ORGANISM-1)?

**Reason Asked by USER:**

**WHY** { request from user to determine motivation for question }

[i.e. **WHY** is it important to determine the number of cultures which were positive for the organism?]

{Teiresias's rephrasing of the WHY

question, and answer}

**Teiresias Explanation :**

- [1.0] This will help to determine whether there is significant disease associated with this occurrence of the organism.
- It has already been established that
  - [1.1] the culture was taken from a sterile source
  - [1.2] the number of cultures in the series is greater than 2

Therefore, if

- [1.3] the number of cultures which were positive for ORGANISM-1 is greater than 2

Then there is suggestive evidence (.5) that there is significant disease associated with this occurrence of the e.coli (ORGANISM-1). [RULEO49]



e. **Knowledge Acquisition** : Knowledge acquisition is transferring knowledge from human expert to computer. There are many programs available that interact with the domain experts to extract expert knowledge efficiently. These programs provide the support for the following activities :

- Entering Knowledge
- Maintaining knowledge base consistency
- Ensuring knowledge base completeness

The most useful knowledge acquisition programs are those that are restricted to a particular problem solving paradigm eg., diagnosis or design.

Example : MOLE , MOLE-p, SALT, META DENDRAL

**MOLE** : MOLE is a knowledge acquisition system for heuristic classification problems such as diagnosing diseases. It is used in conjunction with *the cover and differentiable* problem solving method. An expert system produced by MOLE accepts input data, then uses differentiated knowledge to determine which one is best.

**MOLE-p** : MOLE interacts with a domain expert to produce a knowledge base that a system called MOLE-p uses to solve problems. The acquisition proceeds using the following steps

1. Initial Knowledge base Construction : MOLE asks the experts to list common symptoms or complaints that might require diagnosis. For each symptom MOLE prompts for a list of possible explanations. MOLE then iteratively seeks out higher level explanations until it comes up with a set of ultimate causes.
2. Refinement of knowledge base : It tries to identify the weakness of the knowledge base.

MOLE has been used to build systems that diagnose problems with car engines, problems in steel rolling mills and inefficiencies in coal burning power plants.

**SALT [Marcus and McDemott,1989]** : It builds a dependency network as it converses with the expert. Each node stands for the value of a parameter that must be acquired or generated. Three kinds of links are used in the network to guide the acquisition process

1. **Contributes to** : There must be at least one contributor to link in every non input node. If it is missing the expert is prompted to fill them

2. **Constraints** : These are used in loops to repeat until the constraints are true.
3. **Suggest Revisions** : These include constraints link and loop until the required revisions are made.

**META DENDRAL [Mitchel ,1978]** : It was the first program to **use learning techniques** to construct rules for an expert system **automatically** . It built rules to be used by DENDRA whose job was to determine the structure of complex chemical compounds.

META DENTRAL was able to induce its rules based on a set of mass spectrometry data ; it was then able to identify molecular structures with very high accuracy.

## 2. Write a Short note of MYCIN and EMYCIN

Answer :

This medical expert system was the recommended therapy for **blood /meningitis** infections and the diagnosis normally involved growing cultures of the infecting organisms within 48 hours . MYCIN is a goal directed system that uses a basic backward chaining technique with rules written in LISP.

### Example Rule of MYCIN

**if**

1 the infection is primary bacteremia, and

2 the site of the culture is one of the sterilesites, and

3 the suspected portal of entry of the organism is the gastrointestinal tract

**then**

there is suggestive evidence (0.7) that the identity of the organism is Bacteroides.

### MYCIN Rule 280

**IF:**

The infection which requires therapy is meningitis, and

The type of the infection is fungal, and  
Organisms were not seen on the stain of the culture, and  
The patient is not a compromised host, and  
The patient has been to an area that is endemic for coccidiomycoses, and  
The race of the patient is one of: Black, Asian, Indian, and  
The cryptococcal antigen in the csf was not positive

**THEN:**

There is suggestive evidence (.5) that cryptococcus is not one of the organisms (other than those seen on cultures or smears) which might be causing the infection.

**JUSTIFICATION:**

Dark-skinned races, especially Filipino, Asian and Black (in that order) have an increased susceptibility to coccidiomycoses meningitis.

**A typical dialog A CONSULTATION WITH MYCIN**

- *Please enter information about the patient. Name: Sally Age: 42 years Sex: Female .*
- *Are there any positive cultures obtained from SALLY? No*
- *Are there any pending cultures of SALLY? Yes*
- *Stain examined? Yes,*
- *Stain of smear: Morph:*
- *Has SALLY recently had symptoms of persistent headache or other abnormal neurologic symptoms (dizziness, lethargy, etc.)? Yes*
- *Has SALLY recently had objective evidence of abnormal neurologic signs (nuchal rigidity, coma, seizures, etc.) documented by physician observation or examination? Yes*
- *Is the meningitis a hospital-acquired infection? No*
- *Is SALLY's clinical history consistent with EPIGLOTTITIS? No*
- *Is SALLY's clinical history consistent with OTITIS-MEDIA? No*
- *Has the patient ever been to an area that is endemic for coccidiomycoses? Unknown*
- *Is SALLY a burn patient? Why?*
- *-----*
- *Do you have reason to suspect that SALLY may have impaired renal functions? No*
- *SALLY's weight in kilograms (or POUNDS): 51.3*

### **MYCIN prescription with Explanation :**

Give the following in combination: " ETHAMBUTAL Dose: 1.289 (13.0 100mg-tablets) q24h PO for 60 days [calculated on basis of 25 mg/kg] then 770 mg (7.5 100mg-tablets) q24h PO [calculated on basis of 15 mg/kg] .

[The dose should be modified in renal failure. One can adjust the dose only if a creatinine clearance or a serum creatinine is obtained.]

### **Additional tests and procedures with Explanation**

- Periodic vision screening tests are recommended for optic neuritis.
- Administer dose on an empty stomach.
- Although I suspect that fungus may be a possible pathogen, in the absence of definitive evidence I will not recommend antifungal therapy at this time. Please obtain samples for fungal, TB, and viral cultures, cytology, VDRL (blood and CSF), cryptococcal antigen (blood and CSF), viral titers (blood and CSF). An infectious disease consult may be advisable.

## **3. Who are the different participants in Expert Systems.**

**Answer:** Following are the Participants in Expert Systems Development and Use

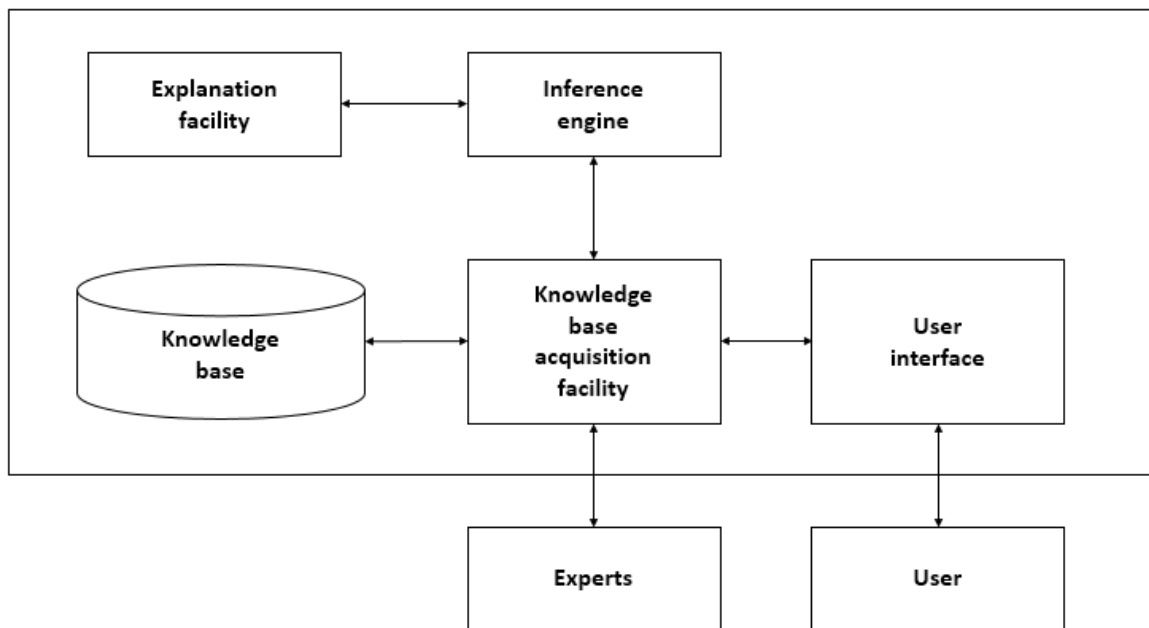
- Domain expert
  - The individual or group whose expertise and knowledge is captured for use in an expert system
- Knowledge user
  - The individual or group who uses and benefits from the expert system
- Knowledge engineer
  - Someone trained or experienced in the design, development, implementation, and maintenance of an expert system

#### 4. With neat diagram explain the different components of the Expert Systems .

Answer :

The different Components of an Expert System are:

1. **Knowledge base:** Stores all relevant information, data, rules, cases, and relationships used by the expert system
2. **Inference engine:** Seeks information and relationships from the knowledge base and provides answers, predictions, and suggestions in the way a human expert would. (Forward or Backward Chaining)
3. **Explanation facility/module:**
4. **Expert Shell:**
5. **Knowledge Acquisition:**
6. **User interface:**



END