- Language meant for communicating with the world.
- Also, By studying language, we can come to understand more about the world.
- If we can succeed at building computational mode of language, we will have a powerful tool for communicating with the world.
- Also, We look at how we can exploit knowledge about the world, in combination with linguistic facts, to build computational natural language systems.

Natural Language Processing (NLP) problem can divide into 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.

Steps in Natural Language Processing

Morphological Analysis

- Individual words analyzed into their components and non-word tokens such as punctuation separated from the words.

Syntactic Analysis

- Linear sequences of words transformed into structures that show how the words relate to each other.
- Moreover, Some word sequences may reject if they violate the language's rule for how words may combine.

Semantic Analysis

- The structures created by the syntactic analyzer assigned meanings.
- Also, A mapping made between the syntactic structures and objects in the task domain.
- Moreover, Structures for which no such mapping possible may reject.

Discourse integration

- The meaning of an individual sentence may depend on the sentences that precede it. And also, may influence the meanings of the sentences that follow it.

Pragmatic Analysis

- Moreover, The structure representing what said reinterpreted to determine what was actually meant.

Summary

- Results of each of the main processes combine to form a natural language system.
- All of the processes are important in a complete natural language understanding system.
- Not all programs are written with exactly these components.
- Sometimes two or more of them collapsed.
- Doing that usually results in a system that is easier to build for restricted subsets of English but one that is harder to extend to wider coverage.

Steps Natural Language Processing

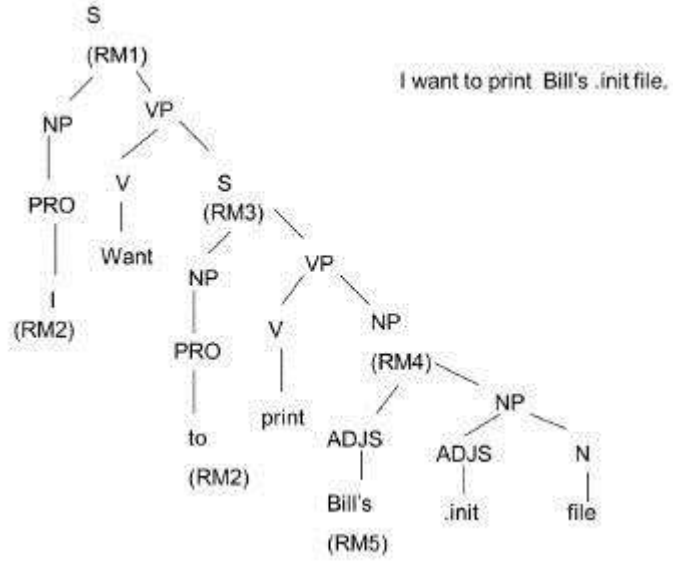
Morphological Analysis

- Suppose we have an English interface to an operating system and the following sentence typed: I want to print Bill's .init file.
- The morphological analysis must do the following things:

- Pull apart the word “Bill’s” into proper noun “Bill” and the possessive suffix “’s”
- Recognize the sequence “.init” as a file extension that is functioning as an adjective in the sentence.
- This process will usually assign syntactic categories to all the words in the sentence.

Syntactic Analysis

- A syntactic analysis must exploit the results of the morphological analysis to build a structural description of the sentence.
- The goal of this process, called parsing, is to convert the flat list of words that form the sentence into a structure that defines the units that represented by that flat list.
- The important thing here is that a flat sentence has been converted into a hierarchical structure. And that the structure corresponds to meaning units when a semantic analysis performed.
- Reference markers (set of entities) shown in the parenthesis in the parse tree.
- Each one corresponds to some entity that has mentioned in the sentence.
- These reference markers are useful later since they provide a place in which to accumulate information about the entities as we get it.



Semantic Analysis

- The 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.

Discourse Integration

- Specifically, we do not know whom the pronoun “I” or the proper noun “Bill” refers to.
- To pin down these references requires an appeal to a model of the current discourse context, from which we can learn that the current user is USER068 and that the only person named “Bill” about whom we could be talking is USER073.
- Once the correct referent for Bill known, we can also determine exactly which file referred to.

Pragmatic Analysis

- The final step toward effective understanding is to decide what to do as a result.

- One possible thing to do to record what was said as a fact and done with it.
- For some sentences, a whose intended effect is clearly declarative, that is the precisely correct thing to do.
- But for other sentences, including this one, the intended effect is different.
- We can discover this intended effect by applying a set of rules that characterize cooperative dialogues.
- The final step in pragmatic processing to translate, from the knowledge-based representation to a command to be executed by the system.

Syntactic Processing

- Syntactic Processing is the step in which a flat input sentence converted into a hierarchical structure that corresponds to the units of meaning in the sentence. This process called parsing.
- It plays an important role in natural language understanding systems for two reasons:
    1. Semantic processing must operate on sentence constituents. If there is no syntactic parsing step, then the semantics system must decide on its own constituents. If parsing is done, on the other hand, it constrains the number of constituents that semantics can consider.
    2. Syntactic parsing is computationally less expensive than is semantic processing. Thus it can play a significant role in reducing overall system complexity.
- Although it is often possible to extract the meaning of a sentence without using grammatical facts, it is not always possible to do so.
- Almost all the systems that are actually used have two main components:
    1. A declarative representation, called a grammar, of the syntactic facts about the language.
    2. A procedure, called parser that compares the grammar against input sentences to produce parsed structures.

Grammars and Parsers

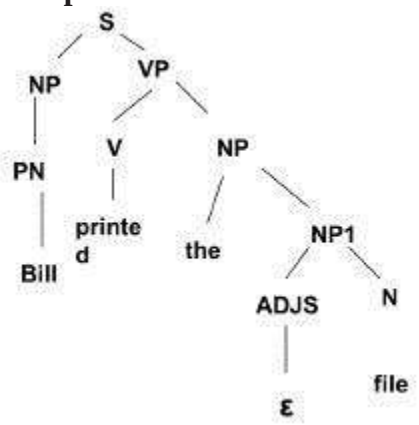
- The most common way to represent grammars is a set of production rules.
- The first rule can read as "A sentence composed of a noun phrase followed by Verb Phrase"; the Vertical bar is OR; ε represents the empty string.
- Symbols that further expanded by rules called non-terminal symbols.
- Symbols that correspond directly to strings that must found in an input sentence called terminal symbols.
- Grammar formalism such as this one underlies many linguistic theories, which in turn provide the basis for many natural language understanding systems.
- Pure context-free grammars are not effective for describing natural languages.
- NLPs have less in common with computer language processing systems such as compilers.
- Parsing process takes the rules of the grammar and compares them against the input sentence.
- The simplest structure to build is a Parse Tree, which simply records the rules and how they matched.
- Every node of the parse tree corresponds either to an input word or to a non-terminal in our grammar.
- Each level in the parse tree corresponds to the application of one grammar rule.

Example for Syntactic Processing – Augmented Transition Network

Syntactic Processing 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. This process called parsing. It plays an important role in natural language understanding systems for two reasons:

1. Semantic processing must operate on sentence constituents. If there is no syntactic parsing step, then the semantics system must decide on its own constituents. If parsing is done, on the other hand, it constrains the number of constituents that semantics can consider.
2. Syntactic parsing is computationally less expensive than is semantic processing. Thus it can play a significant role in reducing overall system complexity.

Example: A Parse tree for a sentence: Bill Printed the file



The grammar specifies two things about a language:

1. Its weak generative capacity, by which we mean the set of sentences that contained within the language. This set made up of precisely those sentences that can completely match by a series of rules in the grammar.
2. Its strong generative capacity, by which we mean the structure to assign to each grammatical sentence of the language.

Augmented Transition Network (ATN)

- An augmented transition network is a top-down parsing procedure that allows various kinds of knowledge to incorporated into the parsing system so it can operate efficiently.
- ATNs build on the idea of using finite state machines (Markov model) to parse sentences.
- Instead of building an automaton for a particular sentence, a collection of transition graphs built.
- A grammatically correct sentence parsed by reaching a final state in any state graph.
- Transitions between these graphs simply subroutine calls from one state to any initial state on any graph in the network.
- A sentence determined to be grammatically correct if a final state reached by the last word in the sentence.
- The ATN is similar to a finite state machine in which the class of labels that can attach to the arcs that define the transition between states has augmented.

Arcs may label with:

- Specific words such as “in’.
- Word categories such as noun.

- Procedures that build structures that will form part of the final parse.
- Procedures that perform arbitrary tests on current input and sentence components that have identified.

Semantic Analysis

- The structures created by the syntactic analyzer assigned meanings.
- A mapping made between the syntactic structures and objects in the task domain.
- Structures for which no such mapping is possible may rejected.
- The semantic analysis must do two important things:
    - It must map individual words into appropriate objects in the knowledge base or database.
    - It must create the correct structures to correspond to the way the meanings of the individual words combine with each other. Semantic Analysis AI
- Producing a syntactic parse of a sentence is only the first step toward understanding it.
- We must produce a representation of the meaning of the sentence.
- Because understanding is a mapping process, we must first define the language into which we are trying to map.
- There is no single definitive language in which all sentence meaning can describe.
- The choice of a target language for any particular natural language understanding program must depend on what is to do with the meanings once they constructed.
- Choice of the target language in Semantic Analysis AI
    - There are two broad families of target languages that used in NL systems, depending on the role that the natural language system playing in a larger system:
    - When natural language considered as a phenomenon on its own, as for example when one builds a program whose goal is to read the text and then answer questions about it. A target language can design specifically to support language processing.
    - When natural language used as an interface language to another program (such as a db query system or an expert system), then the target language must legal input to that other program. Thus the design of the target language driven by the backend program.

Discourse and Pragmatic Processing

To understand a single sentence, it is necessary to consider the discourse and pragmatic context in which the sentence was uttered.
There are a number of important relationships that may hold between phrases and parts of their discourse contexts, including:

1. Identical entities. Consider the text:
    - Bill had a red balloon. o John wanted it.
    - The word “it” should identify as referring to the red balloon. These types of references called anaphora.
2. Parts of entities. Consider the text:
    - Sue opened the book she just bought.
    - The title page was torn.
    - The phrase “title page” should be recognized as part of the book that was just bought.

3. Parts of actions. Consider the text:
    - John went on a business trip to New York.
    - He left on an early morning flight.
    - Taking a flight should recognize as part of going on a trip.
4. Entities involved in actions. Consider the text:
    - My house was broken into last week.
    - Moreover, They took the TV and the stereo.
    - The pronoun “they” should recognize as referring to the burglars who broke into the house.
5. Elements of sets. Consider the text:
    - The decals we have in stock are stars, the moon, item and a flag.
    - I’ll take two moons.
    - Moons mean moon decals.
6. Names of individuals:
    - Dev went to the movies.
7. Causal chains
    - There was a big snow storm yesterday.
    - So, The schools closed today.
8. Planning sequences:
    - Sally wanted a new car
    - She decided to get a job.
9. Implicit presuppositions:
    - Did Joe fail CS101?

The major focus is on using following kinds of knowledge:

- The current focus of the dialogue.
- Also, A model of each participant’s current beliefs.
- Moreover, The goal-driven character of dialogue.
- The rules of conversation shared by all participants.

Statistical Natural Language Processing

Formerly, many language-processing tasks typically involved the direct hand coding of rules, which is not in general robust to natural-language variation. The machine-learning paradigm calls instead for using statistical inference to automatically learn such rules through the analysis of large ***corpora*** of typical real-world examples (a *corpus* (plural, "corpora") is a set of documents, possibly with human or computer annotations).

Many different classes of machine learning algorithms have been applied to natural-language processing tasks. These algorithms take as input a large set of "features" that are generated from the input data. Some of the earliest-used algorithms, such as decision trees, produced systems of hard if-then rules similar to the systems of hand-written rules that were then common. Increasingly, however, research has focused on statistical models, which make soft, probabilistic decisions based on attaching real-valued weights to each input feature. Such models have the advantage that they can express the relative certainty of many different possible answers rather than only one, producing more reliable results when such a model is included as a component of a larger system.

Systems based on machine-learning algorithms have many advantages over hand-produced rules:

- The learning procedures used during machine learning automatically focus on the most common cases, whereas when writing rules by hand it is often not at all obvious where the effort should be directed.
- Automatic learning procedures can make use of statistical inference algorithms to produce models that are robust to unfamiliar input (e.g. containing words or structures that have not been seen before) and to erroneous input (e.g. with misspelled words or words accidentally omitted). Generally, handling such input gracefully with hand-written rules—or more generally, creating systems of hand-written rules that make soft decisions—is extremely difficult, error-prone and time-consuming.
- Systems based on automatically learning the rules can be made more accurate simply by supplying more input data. However, systems based on hand-written rules can only be made more accurate by increasing the complexity of the rules, which is a much more difficult task. In particular, there is a limit to the complexity of systems based on hand-crafted rules, beyond which the systems become more and more unmanageable. However, creating more data to input to machine-learning systems simply requires a corresponding increase in the number of man-hours worked, generally without significant increases in the complexity of the annotation process.

Spell Checking

Spell checking is one of the applications of natural language processing that impacts billions of users daily. A good introduction to spell checking can be found on Peter Norvig's webpage. The article introduces a simple 21-line spell checker implementation in Python combining simple language and error models to predict the word a user intended to type. **The language model estimates how likely a given word `c` is in the language for which the spell checker is designed, this can be written as `P(C)`. The error model estimates the probability `P(w|c)` of typing the misspelled version `w` conditionally to the intention of typing the correctly spelled word `c`.**The spell checker then returns word `c` corresponding to the highest value of `P(w|c)P(c)` among all possible words in the language.

Module 3

LEARNING

Learning is the improvement of performance with experience over time.
Learning element is the portion of a learning AI system that decides how to modify the performance element and implements those modifications.
We all learn new knowledge through different methods, depending on the type of material to be learned, the amount of relevant knowledge we already possess, and the environment in which the learning takes place. There are five methods of learning . They are,

1. Memorization (rote learning)
2. Direct instruction (by being told)
3. Analogy
4. Induction

5. Deduction
Learning by memorizations is the simplest from of le4arning. It requires the least amount of inference and is accomplished by simply copying the knowledge in the same form that it will be used directly into the knowledge base.
Example:- Memorizing multiplication tables, formulate , etc.
Direct instruction is a complex form of learning. This type of learning requires more inference than role learning since the knowledge must be transformed into an operational form before learning when a teacher presents a number of facts directly to us in a well organized manner.
Analogical learning is the process of learning a new concept or solution through the use of similar known concepts or solutions. We use this type of learning when solving problems on an exam where previously learned examples serve as a guide or when make frequent use of analogical learning. This form of learning requires still more inferring than either of the previous forms. Since difficult transformations must be made between the known and unknown situations.
Learning by induction is also one that is used frequently by humans . it is a powerful form of learning like analogical learning which also require s more inferring than the first two methods. This learning re quires the use of inductive inference, a form of invalid but useful inference. We use inductive learning ofinstances of examples of the concept. For example we learn the concepts of color or sweet taste after experiencing the sensations associated with several examples of colored objects or sweet foods.
Deductive learning is accomplished through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationships are logically derived. Deductive learning usually requires more inference than the other methods.
Review Questions:-
1. what is perception ?
2. How do we overcome the Perceptual Problems?
3. Explain in detail the constraint satisfaction waltz algorithm?
4. What is learning ?
5. What is Learning element ?
6. List and explain the methods of learning?
Types of learning:- Classification or taxonomy of learning types serves as a guide in studying or comparing a differences among them. One can develop learning taxonomies based on the type of knowledge representation used (predicate calculus , rules, frames), the type of knowledge learned (concepts, game playing, problem solving), or by the area of application(medical diagnosis , scheduling , prediction and so on).
The classification is intuitively more appealing and is one which has become popular among machine learning researchers . it is independent of the knowledge domain and the representation scheme is used. It is based on the type of inference strategy employed or the methods used in the learning process. The five different learning methods under this taxonomy are:
Memorization (rote learning)
Direct instruction(by being told)
Analogy
Induction
Deduction
Learning by memorization is the simplest form of learning. It requires the least5 amount of inference and is accomplished by simply copying the knowledge in the same form that it will be

used directly into the knowledge base. We use this type of learning when we memorize multiplication tables ,
for example.
A slightly more complex form of learning is by direct instruction. This type of learning requires more understanding and inference than role learning since the knowledge must be transformed into an operational form before being integrated into the knowledge base. We use this type of learning when a teacher presents a number of facts directly to us in a well organized manner.
The third type listed, analogical learning, is the process of learning an ew concept or solution through the use of similar known concepts or solutions. We use this type of learning when solving problems on an examination where previously learned examples serve as a guide or when we learn to drive a truck using our knowledge of car driving. We make frewuence use of analogical learning. This form of learning requires still more inferring than either of the previous forms, since difficult transformations must be made between the known and unknown situations. This is a kind of application of knowledge in a new situation.
The fourth type of learning is also one that is used frequency by humans. It is a powerful form of learning which, like analogical learning, also requires more inferring than the first two methods. This form of learning requires the use of inductive inference, a form of invalid but useful inference. We use inductive learning when wed formulate a general concept after seeing a number of instance or examples of the concept. For example, we learn the concepts of color sweet taste after experiencing the sensation associated with several examples of colored objects or sweet foods.
The final type of acquisition is deductive learning. It is accomplished through a sequence of deductive inference steps using known facts. From the known facts, new facts or relationships are logically derived. Deductive learning usually requires more inference than the other methods. The inference method used is, of course , a deductive type, which is a valid from of inference.
In addition to the above classification, we will sometimes refer to learning methods as wither methods or knowledge-rich methods. Weak methods are general purpose methods in which little or no initial knowledge is available. These methods are more mechanical than the classical AI knowledge – rich methods. They often rely on a form of heuristics search in the learning process.

Rote Learning

Rote learning is the basic learning activity. **Rote learning** is a memorization technique based on repetition. 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.

Rote learning technique can also be used in complex learning systems provided sophisticated techniques are employed to use the stored values faster and there is a generalization to keep the number of stored information down to a manageable level. Checkers-playing program, for ex

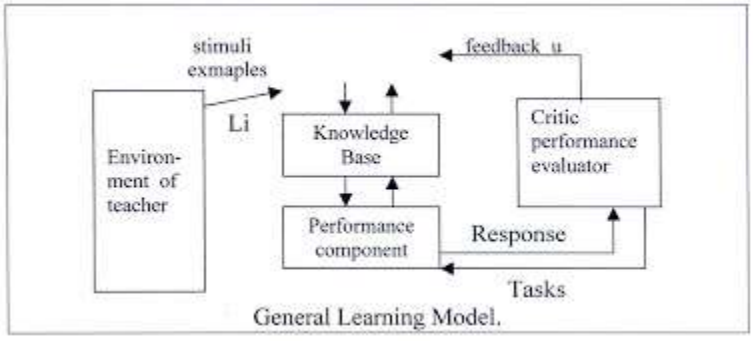
The idea is that one will be able to quickly recall the meaning of the material the more one repeats it. Some of the alternatives to rote learning include meaningful learning, associative learning, and active learning. ample, uses this technique to learn the board positions it evaluates in its look-ahead search.

Learning By Taking Advice.

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 to name a few. 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. Moreover the reliability of the source of knowledge should be considered.
The system should ensure that the new knowledge is conflicting with the existing knowledge. 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). Now this knowledge is ready to use.

General Learning Model.
General Learning Model: - AS noted earlier, learning can be accomplished using a number of different methods, such as by memorization facts, by being told, or by studying examples like problem solution. Learning requires that new knowledge structures be created from some form of input stimulus. This new knowledge must then be assimilated into a knowledge base and be tested in some way for its utility. Testing means that the knowledge should be used in performance of some task from which meaningful feedback can be obtained, where the feedback provides some measure of the accuracy and usefulness of the newly acquired knowledge.
General Learning Model



General Learning Model.

general learning model is depicted in figure 4.1 where the environment has been included as a part of the overall learner system. The environment may be regarded as either a form of nature which produces random stimuli or as a more organized training source such as a teacher which provides carefully selected training examples for the learner component. The actual form of environment used will depend on the particular learning paradigm. In any case, some representation language must be assumed for communication between the environment and the learner. The language may be the same representation scheme as that used in the knowledge base (such as a form of predicate calculus). When they are hosen to be the same, we say the single representation trick is being used. This usually results in a simpler implementation since it is not necessary to transform between two or more different representations.

For some systems the environment may be a user working at a keyboard . Other systems will use program modules to simulate a particular environment. In even more realistic cases the system will have real physical sensors which interface with some world environment.

Inputs to the learner component may be physical stimuli of some type or descriptive , symbolic training examples. The information conveyed to the learner component is used to create and modify knowledge structures in the knowledge base. This same knowledge is used by the performance component to carry out some tasks, such as solving a problem playing a game, or classifying instances of some concept.

given a task, the performance component produces a response describing its action in performing the task. The critic module then evaluates this response relative to an optimal response.

Feedback , indicating whether or not the performance was acceptable , is then sent by the critic module to the learner component for its subsequent use in modifying the structures in the knowledge base. If proper learning was accomplished, the system's performance will have improved with the changes made to the knowledge base.

The cycle described above may be repeated a number of times until the performance of the system has reached some acceptable level, until a known learning goal has been reached, or until changes ceases to occur in the knowledge base after some chosen number of training examples have been observed.

There are several important factors which influence a system's ability to learn in addition to the form of representation used. They include the types of training provided, the form and extent of any initial background knowledge , the type of feedback provided, and the learning algorithms used.

The type of training used in a system can have a strong effect on performance, much the same as it does for humans. Training may consist of randomly selected instance or examples that have been carefully selected and ordered for presentation. The instances may be positive examples of some concept or task a being learned, they may be negative, or they may be mixture of both positive and negative. The instances may be well focused using only relevant information, or they may contain a variety of facts and details including irrelevant data.

There are Many forms of learning can be characterized as a search through a space of possible hypotheses or solutions. To make learning more efficient. It is necessary to constrain this search process or reduce the search space. One method of achieving this is through the use of background knowledge which can be used to constrain the search space or exercise control operations which limit the search process.

Feedback is essential to the learner component since otherwise it would never know if the knowledge structures in the knowledge base were improving or if they were adequate for the performance of the given tasks. The feedback may be a simple yes or no type of evaluation, or it

may contain more useful information describing why a particular action was good or bad. Also , the feedback may be completely reliable, providing an accurate assessment of the performance or it may contain noise, that is the feedback may actually be incorrect some of the time. Intuitively , the feedback must be accurate more than 50% of the time; otherwise the system carries useful information, the learner should also to build up a useful corpus of knowledge quickly. On the other hand, if the feedback is noisy or unreliable, the learning process may be very slow and the resultant knowledge incorrect.

Learning Neural Network

Perceptron

- The perceptron an invention of (1962) Rosenblatt was one of the earliest neural network models.
- Also, It models a neuron by taking a weighted sum of its inputs and sending the output 1 if the sum is greater than some adjustable threshold value (otherwise it sends 0).

$$g(x) = \sum_{k=0}^{n} WkXk \qquad Output(x) = \begin{cases} 1 & \text{if } g(x) > 0 \\ 0 & \text{if } g(x) > 0 \end{cases}$$

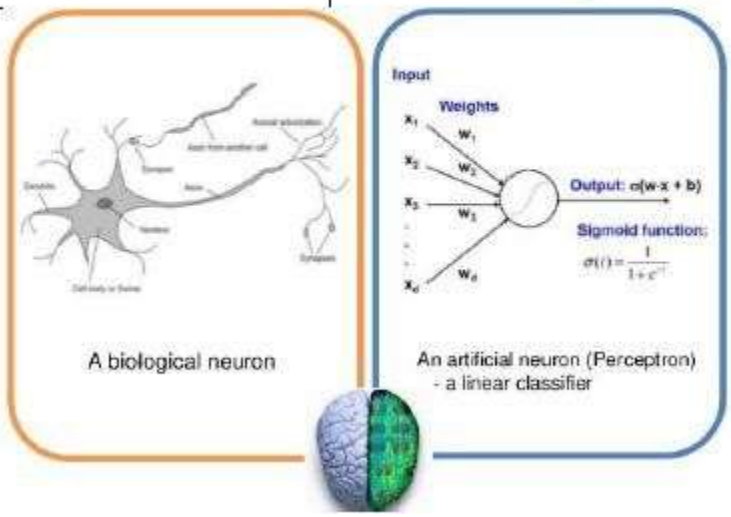


Figure: A neuron & a Perceptron

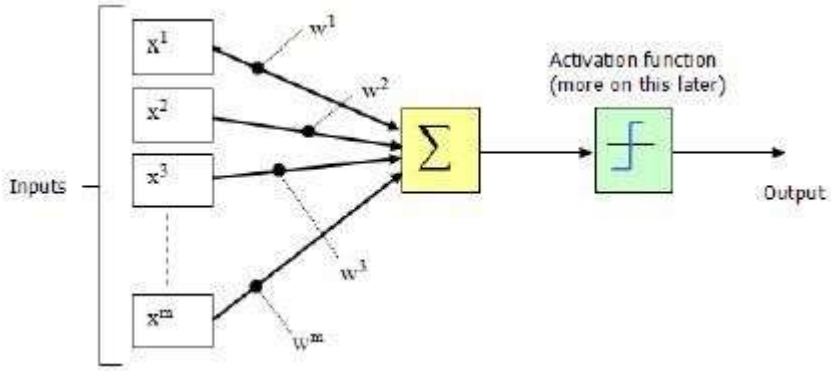


Figure: Perceptron with adjustable threshold

- In case of zero with two inputs g(x) = w0 + w1x1 + w2x2 = 0

- x2 = -(w1/w2)x1 – (w0/w2) → equation for a line
- the location of the line is determined by the weight w0 w1 and w2
- if an input vector lies on one side of the line, the perceptron will output 1
- if it lies on the other side, the perception will output 0
- Moreover, Decision surface: a line that correctly separates the training instances corresponds to a perfectly function perceptron.

Perceptron Learning Algorithm

Given: A classification problem with n input feature (x_1, x_2, …., x_n) and two output classes.
Compute A set of weights (w_0, w_1, w_2,….,w_n) that will cause a perceptron to fire whenever the input falls into the first output class.

1. Create a perceptron with n+ 1 input and n+ 1 weight, where the x_0 is always set to 1.
2. Initialize the weights (w_0, w_1,…., w_n) to random real values.
3. Iterate through the training set, collecting all examples *misclassified* by the current set of weights.
4. If all examples are classified correctly, output the weights and quit.
5. Otherwise, compute the vector sum S of the misclassified input vectors where each vector has the form (x_0, x_1, …, Xn). In creating the sum, add to S a vector x if x is an input for which the perceptron incorrectly fails to fire, but – x if x is an input for which the perceptron incorrectly fires. Multiply sum by a scale factor η.
6. Moreover, Modify the weights (w_0, w_1, …, w_n) by adding the elements of the vector S to them.
7. Go to step 3.

- The perceptron learning algorithm is a search algorithm. It begins with a random initial state and finds a solution state. The search space is simply all possible assignments of real values to the weights of the perception, and the search strategy is gradient descent.
- The perceptron learning rule is guaranteed to converge to a solution in a finite number of steps, so long as a solution exists.
- Moreover, This brings us to an important question. What problems can a perceptron solve? Recall that a single-neuron perceptron is able to divide the input space into two regions.
- Also, The perception can be used to classify input vectors that can be separated by a linear boundary. We call such vectors linearly separable.
- Unfortunately, many problems are not linearly separable. The classic example is the XOR gate. It was the inability of the basic perceptron to solve such simple problems that are not linearly separable or non-linear.

Genetic Learning

Supervised Learning

Supervised learning is the machine learning task of inferring a function from labeled training data.
Moreover, The training data consist of a set of training examples.
In supervised learning, each example a pair consisting of an input object (typically a vector) and the desired output value (also called the supervisory signal).

Training set

A training set a set of data used in various areas of information science to discover potentially predictive relationships.

Training sets used in artificial intelligence, machine learning, genetic programming, intelligent systems, and statistics.
In all these fields, a training set has much the same role and often used in conjunction with a test set.

Testing set

A **test set** is a set of data used in various areas of information science to assess the strength and utility of a predictive relationship.
Moreover, Test sets are used in artificial intelligence, machine learning, genetic programming, and statistics. In all these fields, a test set has much the same role.

Accuracy of classifier: Supervised learning

In the fields of science, engineering, industry, and statistics. The accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's actual (true) value.

Sensitivity analysis: Supervised learning

Similarly, Local Sensitivity as correlation coefficients and partial derivatives can only use, if the correlation between input and output is linear.

Regression: Supervised learning

In statistics, **regression analysis** is a statistical process for estimating the relationships among variables.
Moreover, It includes many techniques for modeling and analyzing several variables. When the focus on the relationship between a dependent variable and one or more independent variables. More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables varied. Moreover, While the other independent variables held fixed.

Expert systems:

Expert system = knowledge + problem-solving methods. ... A knowledge base that captures the domain-specific knowledge and an inference engine that consists of algorithms for manipulating the knowledge represented in the knowledge base to solve a problem presented to the system.
Expert systems (ES) are 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.

Characteristics of Expert Systems

- High performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems

The expert systems are capable of −

- Advising
- Instructing and assisting human in decision making
- Demonstrating
- Deriving a solution
- Diagnosing
- Explaining
- Interpreting input
- Predicting results
- Justifying the conclusion
- Suggesting alternative options to a problem

They are incapable of −

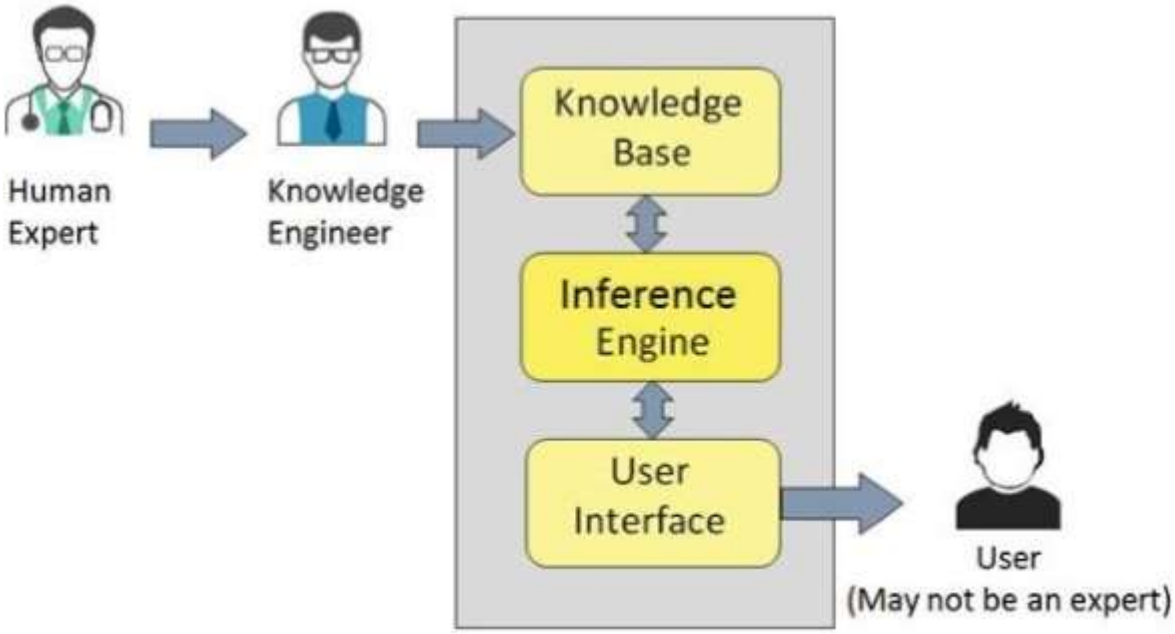
- Substituting human decision makers
- Possessing human capabilities
- Producing accurate output for inadequate knowledge base
- Refining their own knowledge

Components of Expert Systems

The components of ES include −

- Knowledge Base
- Inference Engine
- User Interface

Let us see them one by one briefly −



Knowledge Base

It contains domain-specific and high-quality knowledge. Knowledge is required to exhibit intelligence. The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

What is Knowledge?

The data is collection of facts. The information is organized as data and facts about the task domain. Data, information, and past experience combined together are termed as knowledge.

Components of Knowledge Base

The knowledge base of an ES is a store of both, factual and heuristic knowledge.

- Factual Knowledge − It is the information widely accepted by the Knowledge Engineers and scholars in the task domain.
- Heuristic Knowledge − It is about practice, accurate judgement, one's ability of evaluation, and guessing.

Knowledge representation

It is the method used to organize and formalize the knowledge in the knowledge base. It is in the form of IF-THEN-ELSE rules.

Knowledge Acquisition

The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledge base.

The knowledge base is formed by readings from various experts, scholars, and the Knowledge Engineers. The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills.

He acquires information from subject expert by recording, interviewing, and observing him at work, etc. He then categorizes and organizes the information in a meaningful way, in the form of IF-THEN-ELSE rules, to be used by interference machine. The knowledge engineer also monitors the development of the ES.

Inference Engine

Use of efficient procedures and rules by the Inference Engine is essential in deducting a correct, flawless solution.

In case of knowledge-based ES, the Inference Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

In case of rule based ES, it −

- Applies rules repeatedly to the facts, which are obtained from earlier rule application.
- Adds new knowledge into the knowledge base if required.
- Resolves rules conflict when multiple rules are applicable to a particular case.

To recommend a solution, the Inference Engine uses the following strategies −

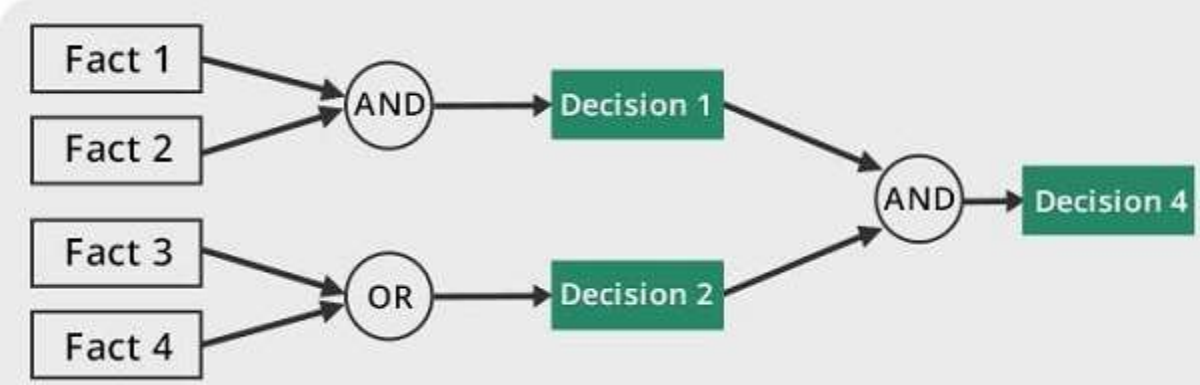
- Forward Chaining
- Backward Chaining

Forward Chaining

It is a strategy of an expert system to answer the question, "What can happen next?"

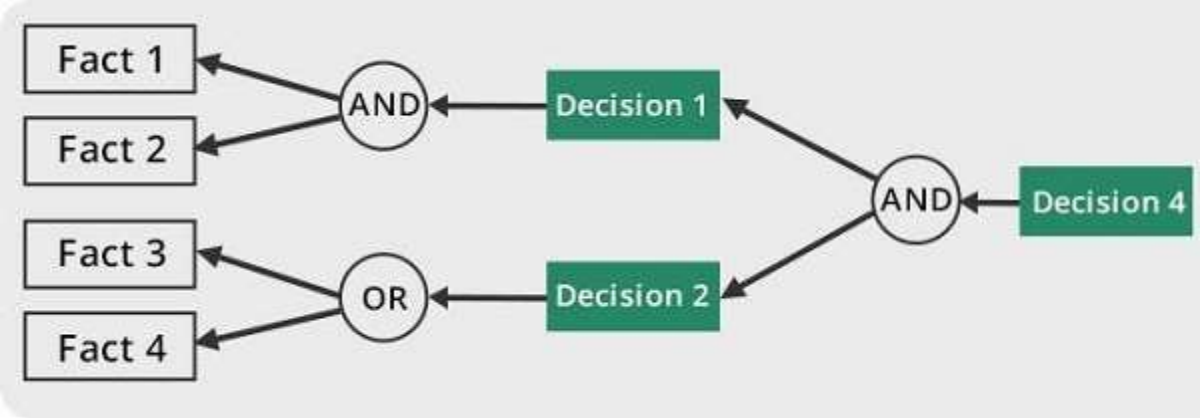
Here, the Inference Engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules, and sorts them before concluding to a solution.

This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

With this strategy, an expert system finds out the answer to the question, "Why this happened?" On the basis of what has already happened, the Inference Engine tries to find out which conditions could have happened in the past for this result. This strategy is followed for finding out cause or reason. For example, diagnosis of blood cancer in humans.



User Interface

User interface provides interaction between user of the ES and the ES itself. It is generally Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.

It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms −

- Natural language displayed on screen.
- Verbal narrations in natural language.
- Listing of rule numbers displayed on the screen.

The user interface makes it easy to trace the credibility of the deductions.

Requirements of Efficient ES User Interface

- It should help users to accomplish their goals in shortest possible way.
- It should be designed to work for user's existing or desired work practices.
- Its technology should be adaptable to user's requirements; not the other way round.
- It should make efficient use of user input.

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

Applications of Expert System

The following table shows where ES can be applied.

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

Expert System Technology

There are several levels of ES technologies available. Expert systems technologies include −

- Expert System Development Environment − The ES development environment includes hardware and tools. They are −
    - Workstations, minicomputers, mainframes.
    - High level Symbolic Programming Languages such as LISt Programming (LISP) and PROgrammation en LOGique (PROLOG).
    - Large databases.
- Tools − They reduce the effort and cost involved in developing an expert system to large extent.
    - Powerful editors and debugging tools with multi-windows.
    - They provide rapid prototyping
    - Have Inbuilt definitions of model, knowledge representation, and inference design.
- Shells − A shell is nothing but an expert system without knowledge base. A shell provides the developers with knowledge acquisition, inference engine, user interface, and explanation facility. For example, few shells are given below −
    - Java Expert System Shell (JESS) that provides fully developed Java API for creating an expert system.
    - *Vidwan*, a shell developed at the National Centre for Software Technology, Mumbai in 1993. It enables knowledge encoding in the form of IF-THEN rules.

Development of Expert Systems: General Steps

The process of ES development is iterative. Steps in developing the ES include −

Identify Problem Domain

- The problem must be suitable for an expert system to solve it.
- Find the experts in task domain for the ES project.
- Establish cost-effectiveness of the system.

Design the System

- Identify the ES Technology
- Know and establish the degree of integration with the other systems and databases.
- Realize how the concepts can represent the domain knowledge best.

Develop the Prototype

From Knowledge Base: The knowledge engineer works to −

- Acquire domain knowledge from the expert.
- Represent it in the form of If-THEN-ELSE rules.

Test and Refine the Prototype

- The knowledge engineer uses sample cases to test the prototype for any deficiencies in performance.
- End users test the prototypes of the ES.

Develop and Complete the ES

- Test and ensure the interaction of the ES with all elements of its environment, including end users, databases, and other information systems.
- Document the ES project well.
- Train the user to use ES.

Maintain the ES

- Keep the knowledge base up-to-date by regular review and update.
- Cater for new interfaces with other information systems, as those systems evolve.

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 System.

DEFINITION - An expert system is a computer program that simulates the judgement and behavior of a human or an organization that has expert knowledge and experience in a particular field. Typically, such a system contains a knowledge base containing accumulated experience and a set of rules for applying the knowledge base to each particular situation that is described to the program. Sophisticated expert systems can be enhanced with additions to the knowledge base or to the set of rules.

Among the best-known expert systems have been those that play chess and that assist in medical diagnosis.

An **expert system** is software that attempts to provide an answer to a problem, or clarify uncertainties where normally one or more human experts would need to be consulted. Expert systems are most common in a specific problem domain, and is a traditional application and/or

subfield of artificial intelligence (AI). A wide variety of methods can be used to simulate the performance of the expert; however, common to most or all are: 1) the creation of a knowledge base which uses some knowledge representation structure to capture the knowledge of the Subject Matter Expert (SME); 2) a process of gathering that knowledge from the SME and codifying it according to the structure, which is called knowledge engineering; and 3) once the system is developed, it is placed in the same real world problem solving situation as the human SME, typically as an aid to human workers or as a supplement to some information system. Expert systems may or may not have learning components.

factors

The MYCIN rule-based expert system introduced a quasi-probabilistic approach called certainty factors, whose rationale is explained below.

A human, when reasoning, does not always make statements with 100% confidence: he might venture, "If Fritz is green, then he is probably a frog" (after all, he might be a chameleon). This type of reasoning can be imitated using numeric values called confidences. For example, if it is known that Fritz is green, it might be concluded with 0.85 confidence that he is a frog; or, if it is known that he is a frog, it might be concluded with 0.95 confidence that he hops. These certainty factor (CF) numbers quantify uncertainty in the degree to which the available evidence supports a hypothesis. They represent a degree of confirmation, and are not probabilities in a Bayesian sense. The CF calculus, developed by Shortliffe & Buchanan, increases or decreases the CF associated with a hypothesis as each new piece of evidence becomes available. It can be mapped to a probability update, although degrees of confirmation are not expected to obey the laws of probability. It is important to note, for example, that evidence for hypothesis H may have nothing to contribute to the degree to which Not_h is confirmed or disconfirmed (e.g., although a fever lends some support to a diagnosis of infection, fever does not disconfirm alternative hypotheses) and that the sum of CFs of many competing hypotheses may be greater than one (i.e., many hypotheses may be well confirmed based on available evidence).

The CF approach to a rule-based expert system design does not have a widespread following, in part because of the difficulty of meaningfully assigning CFs a priori. (The above example of green creatures being likely to be frogs is excessively naive.) Alternative approaches to quasi-probabilistic reasoning in expert systems involve fuzzy logic, which has a firmer mathematical foundation. Also, rule-engine shells such as Drools and Jess do not support probability manipulation: they use an alternative mechanism called salience, which is used to prioritize the order of evaluation of activated rules.

In certain areas, as in the tax-advice scenarios discussed below, probabilistic approaches are not acceptable. For instance, a 95% probability of being correct means a 5% probability of being wrong. The rules that are defined in such systems have no exceptions: they are only a means of achieving software flexibility when external circumstances change frequently. Because rules are stored as data, the core software does not need to be rebuilt each time changes to federal and state tax codes are announced.

Chaining

Two methods of reasoning when using inference rules are forward chaining and backward chaining.

Forward chaining starts with the data available and uses the inference rules to extract more data until a desired goal is reached. An inference engine using forward chaining searches the inference rules until it finds one in which the if clause is known to be true. It then concludes the then clause and adds this information to its data. It continues to do this until a goal is reached. Because the data available determines which inference rules are used, this method is also classified as data driven.

Backward chaining starts with a list of goals and works backwards to see if there is data which will allow it to conclude any of these goals. An inference engine using backward chaining would search the inference rules until it finds one which has a then clause that matches a desired goal. If the if clause of that inference rule is not known to be true, then it is added to the list of goals.

SW Architecture.

The following general points about expert systems and their architecture have been outlined:

1. The sequence of steps taken to reach a conclusion is dynamically synthesized with each new case. The sequence is not explicitly programmed at the time that the system is built.

2. Expert systems can process multiple values for any problem parameter. This permits more than one line of reasoning to be pursued and the results of incomplete (not fully determined) reasoning to be presented.

3. Problem solving is accomplished by applying specific knowledge rather than specific technique. This is a key idea in expert systems technology. It reflects the belief that human experts do not process their knowledge differently from others, but they do possess different knowledge. With this philosophy, when one finds that their expert system does not produce the desired results, work begins to expand the knowledge base, not to re-program the procedures.

End user

There are two styles of user-interface design followed by expert systems. In the original style of user interaction, the software takes the end-user through an interactive dialog. In the following example, a backward-chaining system seeks to determine a set of restaurants to recommend:

Q. Do you know which restaurant you want to go to?

A. No

Q. Is there any kind of food you would particularly like?

A. No

Q. Do you like spicy food?

A. No

Q. Do you usually drink wine with meals?

A. Yes

Q. When you drink wine, is it French wine?

A. Yes

Participants

There are generally three individuals having an interaction in an expert system. Primary among these is the end-user, the individual who uses the system for its problem solving assistance. In the construction and maintenance of the system there are two other roles: the problem domain expert who builds the system and supplies the knowledge base, and a knowledge engineer who assists the experts in determining the representation of their knowledge, enters this knowledge into an explanation module and who defines the inference technique required to solve the problem. Usually the knowledge engineer will represent the problem solving activity in the form of rules. When these rules are created from domain expertise, the knowledge base stores the rules of the expert system.

Inference rule

An understanding of the "inference rule" concept is important to understand expert systems. An inference rule is a conditional statement with two parts: an if clause and a then clause. This rule is what gives expert systems the ability to find solutions to diagnostic and prescriptive problems. An example of an inference rule is:

If the restaurant choice includes French and the occasion is romantic,

Then the restaurant choice is definitely Paul Bocuse.

Procedure node interface

The function of the procedure node interface is to receive information from the procedures coordinator and create the appropriate procedure call. The ability to call a procedure and receive information from that procedure can be viewed as simply a generalization of input from the external world. In some earlier expert systems external information could only be obtained in a

predetermined manner, which only allowed certain information to be acquired. Through the knowledge base, this expert system disclosed in the cross-referenced application can invoke any procedure allowed on its host system. This makes the expert system useful in a much wider class of knowledge domains than if it had no external access or only limited external access.

In the area of machine diagnostics using expert systems, particularly self-diagnostic applications, it is not possible to conclude the current state of "health" of a machine without some information. The best source of information is the machine itself, for it contains much detailed information that could not reasonably be provided by the operator.

The knowledge that is represented in the system appears in the rulebase. In the rulebase described in the cross-referenced applications, there are basically four different types of objects, with the associated information:

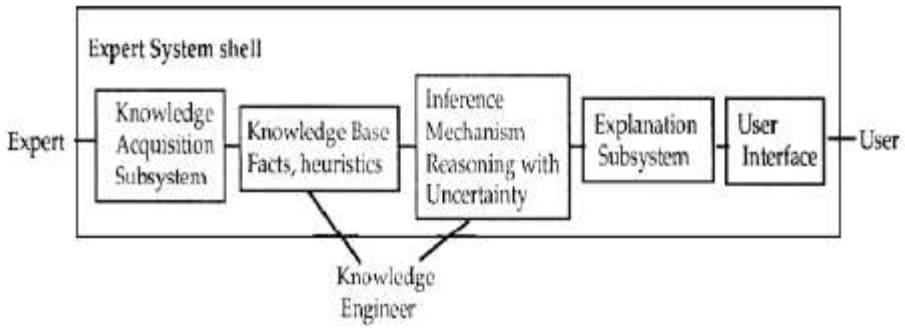
1. Classes: Questions asked to the user.

2. Parameters: Place holders for character strings which may be variables that can be inserted into a class question at the point in the question where the parameter is positioned.

3. Procedures: Definitions of calls to external procedures.

3. Rule nodes: Inferences in the system are made by a tree structure which indicates the rules or logic mimicking human reasoning. The nodes of these trees are called rule nodes. There are several different types of rule nodes.

Expert Systems/**Shells**. The E.S **shell** simplifies the process of creating a knowledge base. It is the **shell** that actually processes the information entered by a user relates it to the concepts contained in the knowledge base and provides an assessment or solution for a particular problem.



Knowledge Acquisition

Knowledge acquisition is the process used to define the rules and ontologies required for a knowledge-based system. The phrase was first used in conjunction with expert systems to describe the initial tasks associated with developing an expert system, namely finding and interviewing domain experts and capturing their knowledge via rules, objects, and frame-based ontologies.

Expert systems were one of the first successful applications of artificial intelligence technology to real world business problems. Researchers at Stanford and other AI laboratories worked with doctors and other highly skilled experts to develop systems that could automate complex tasks such as medical diagnosis. Until this point computers had mostly been used to automate highly data intensive tasks but not for complex reasoning. Technologies such as inference engines allowed developers for the first time to tackle more complex problems.

As expert systems scaled up from demonstration prototypes to industrial strength applications it was soon realized that the acquisition of domain expert knowledge was one of if not the most critical task in the knowledge engineering process. This knowledge acquisition process became an intense area of research on its own. One of the earlier works on the topic used Batesonian theories of learning to guide the process.

One approach to knowledge acquisition investigated was to use natural language parsing and generation to facilitate knowledge acquisition. Natural language parsing could be performed on manuals and other expert documents and an initial first pass at the rules and objects could be developed automatically. Text generation was also extremely useful in generating explanations for system behavior. This greatly facilitated the development and maintenance of expert systems.

A more recent approach to knowledge acquisition is a re-use based approach. Knowledge can be developed in ontologies that conform to standards such as the Web Ontology Language (OWL). In this way knowledge can be standardized and shared across a broad community of knowledge workers. One example domain where this approach has been successful is bioinformatics.

Refferences

1. Elaine Rich, Kevin Knight, & Shivashankar B Nair, Artificial Intelligence, McGraw Hill, 3rd ed.,2009

References:

1) Introduction to Artificial Intelligence & Expert Systems, Dan W Patterson, PHI.,2010

2) S Kaushik, Artificial Intelligence, Cengage Learning, 1st ed.2011