Unit 4: Approaches

Connectionism:

Connectionism is a set of approaches in the fields of artificial intelligence, cognitive psychology, cognitive science, neuroscience and philosophy of mind, that models mental or behavioral phenomena as the emergent processes of *interconnected networks of simple units*. There are many forms of connectionism, but the most common forms use neural network models. **Connectionism** is the name for the computer modeling approach to information processing based on the design or architecture of the brain. **Neural networks** are by far the most commonly used connectionist model today

In most connectionist models, networks change over time. A closely related and very common aspect of connectionist models is *activation*. At any time, a unit in the network has an activation, which is a numerical value intended to represent some aspect of the unit. For example, if the units in the model are neurons, the activation could represent the probability that the neuron would generate an action potential spike. If the model is a spreading activation model, then over time a unit's activation spreads to all the other units connected to it. Spreading activation is always a feature of neural network models, and it is very common in connectionist models used by cognitive psychologists.

Connectionist Models: Refer unit 3 Neural Network.

Learning in Neural Networks:

Learning: One of the powerful features of neural networks is learning. **Learning in neural networks is carried out by adjusting the connection weights among neurons**. It is similar to a biological nervous system in which learning is carried out by changing synapses connection strengths, among cells.

The operation of a neural network is determined by the values of the interconnection weights. There is no algorithm that determines how the weights should be assigned in order to solve specific problems. Hence, the weights are determined by a learning process

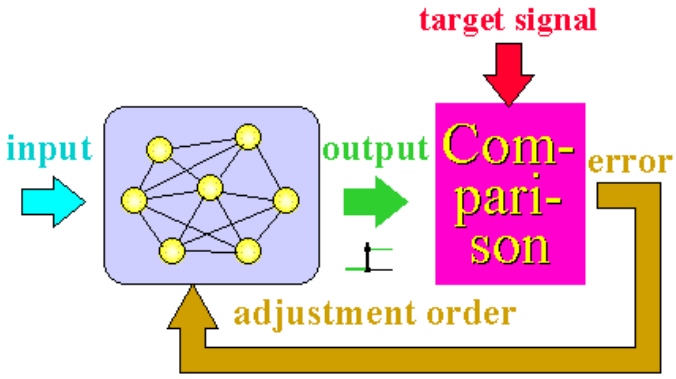
Learning may be classified into two categories:

(1) Supervised Learning
(2) Unsupervised Learning

Supervised Learning:

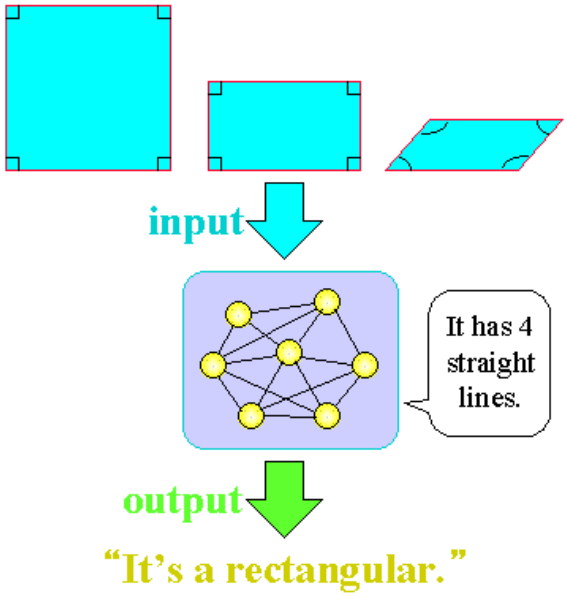
In supervised learning, the network is presented with inputs together with the target (teacher signal) outputs. Then, the neural network tries to produce an output as close as possible to the target signal by adjusting the values of internal weights. The most common supervised learning method is the “error correction method”.

Error correction method is used for networks which their neurons have discrete output functions. Neural networks are trained with this method in order to reduce the error (difference between the network's output and the desired output) to zero.



Unsupervised Learning:

In unsupervised learning, there is no teacher (target signal) from outside and the network adjusts its weights in response to only the input patterns. A typical example of unsupervised learning is **Hebbian learning.**



Consider a machine (or living organism) which receives some sequence of inputs x1, x2, x3, . . ., where xt is the sensory input at time t. In supervised learning the machine is given a sequence of input & a sequence of desired outputs y1, y2, . . . , and the goal of the machine is to learn to produce the correct output given a new input. While, in unsupervised learning the machine simply receives inputs x1, x2, . . ., but obtains neither supervised target outputs, nor rewards from its environment. It may seem somewhat mysterious to imagine what the machine could possibly learn given that it doesn't get any feedback from its environment.

However, it is possible to develop of formal framework for unsupervised learning based on the notion that the machine's goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise.

Hebbian Learning:

The oldest and most famous of all learning rules is Hebb's postulate of learning:

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B is increased"

From the point of view of artificial neurons and artificial neural networks, Hebb's principle can be described as a method of determining how to alter the weights between model neurons. **The weight between two neurons increases if the two neurons activate simultaneously—and reduces if they activate separately.** Nodes that tend to be either both positive or both negative at the same time have strong positive weights, while those that tend to be opposite have strong negative weights.

Hebb's Algorithm:

Step 0: initialize all weights to 0

Step 1: Given a training input, s, with its target output, t, set the activations of the input units: $x_i = s_i$

Step 2: Set the activation of the output unit to the target value: $y = t$

Step 3: Adjust the weights: $w_i(new) = w_i(old) + x_i y$

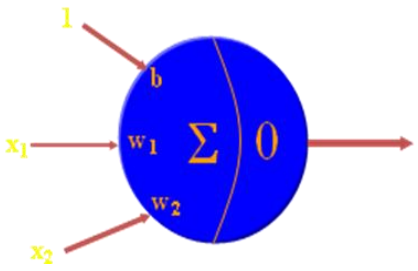
Step 4: Adjust the bias (just like the weights): $b(new) = b(old) + y$

Example:

PROBLEM: Construct a Hebb Net which performs like an AND function, that is, only when both features are "active" will the data be in the target class.

TRAINING SET (with the bias input always at 1):

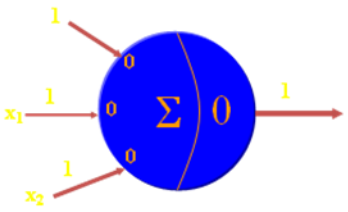
x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



Training-First Input:

- Initialize the weights to 0

Present the first input:
(1 1 1) with a target of 1



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

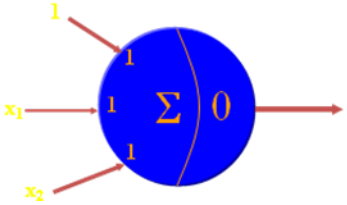
$= 0 + 1 = 1$

$w_2(new) = w_2(old) + x_2 t$

$= 0 + 1 = 1$

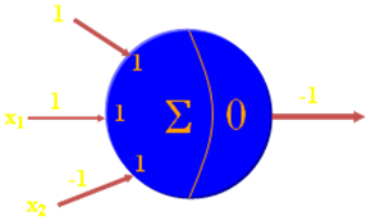
$b(new) = b(old) + t$

$= 0 + 1 = 1$



Training- Second Input:

- **Present the second input:**
  (1 -1 1) with a target of -1



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

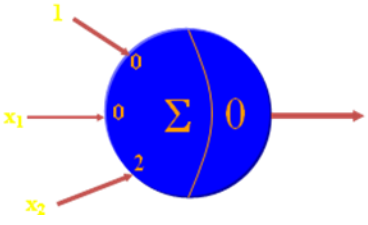
$= 1 + 1(-1) = 0$

$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 2$

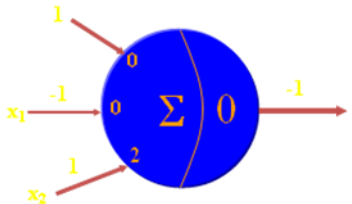
$b(new) = b(old) + t$

$= 1 + (-1) = 0$



Training- Third Input:

- **Present the third input: (-1 1 1) with a target of -1**



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

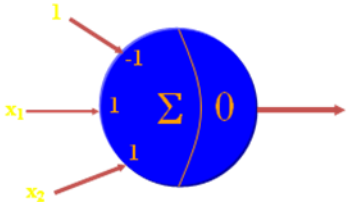
$= 0 + (-1)(-1) = 1$

$w_2(new) = w_2(old) + x_2 t$

$= 2 + 1(-1) = 1$

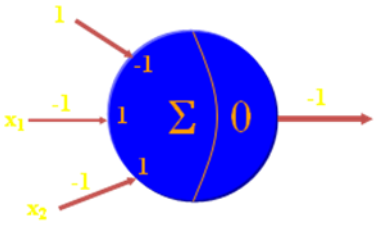
$b(new) = b(old) + t$

$= 0 + (-1) = -1$



Training- Fourth Input:

- **Present the fourth input: (-1 -1 1) with a target of -1**



Update the weights:

$w_1(new) = w_1(old) + x_1 t$

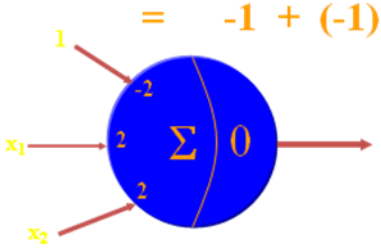
$= 1 + (-1)(-1) = 2$

$w_2(new) = w_2(old) + x_2 t$

$= 1 + (-1)(-1) = 1$

$b(new) = b(old) + t$

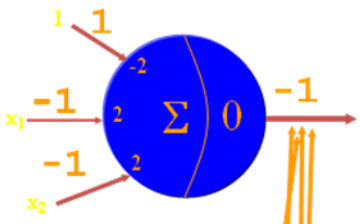
$= -1 + (-1) = -2$



Final Neuron:

- **This neuron works:**

x1	x2	bias	Target
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1



$1*2 + 1*2 + 1*(-2) = 2 > 0$

$(-1)*2 + 1*2 + 1*(-2) = -2 < 0$

$1*2 + (-1)*2 + 1*(-2) = -2 < 0$

$(-1)*2 + (-1)*2 + 1*(-2) = -6 < 0$

Perceptron Learning Theory:

The term "Perceptrons" was coined by Frank RosenBlatt in 1962 and is used to describe the connection of simple neurons into networks. These networks are simplified versions of the real nervous system where some properties are exagerrated and others are ignored. For the moment we will concentrate on Single Layer Perceptrons.

So how can we achieve learning in our model neuron? We need to train them so they can do things that are useful. To do this we must allow the neuron to learn from its mistakes. There is in fact a learning paradigm that achieves this, it is known as supervised learning and works in the following manner.

i. set the weight and thresholds of the neuron to random values.
ii. present an input.
iii. caclulate the output of the neuron.
iv. alter the weights to reinforce correct decisions and discourage wrong decisions, hence reducing the error. So for the network to learn we shall increase the weights on the active inputs when we want the output to be active, and to decrease them when we want the output to be inactive.
v. Now present the next input and repeat steps iii. - v.

Perceptron Learning Algorithm:

The algorithm for Perceptron Learning is based on the supervised learning procedure discussed previously.

Algorithm:

i. Initialize weights and threshold.

   Set $w_i(t)$, $(0 <= i <= n)$, to be the weight i at time t, and $ø$ to be the threshold value in the output node. Set w_0 to be $-ø$, the bias, and x_0 to be always 1.

   Set $w_i(0)$ to small random values, thus initializing the weights and threshold.

ii. Present input and desired output

   Present input $x_0, x_1, x_2, ..., x_n$ and desired output $d(t)$

iii. Calculate the actual output

   $y(t) = g\,[w_0(t)x_0(t) + w_1(t)x_1(t) + ... + w_n(t)x_n(t)]$

iv. Adapts weights

   $w_i(t+1) = w_i(t) + \alpha[d(t) - y(t)]x_i(t)$, where $0 <= \alpha <= 1$ (learning rate) is a positive gain function that controls the adaption rate.

Steps iii. and iv. are repeated until the iteration error is less than a user-specified error threshold or a predetermined number of iterations have been completed.

Please note that the weights only change if an error is made and hence this is only when learning shall occur.

Delta Rule:

The **delta rule** is a gradient descent learning rule for updating the weights of the artificial neurons in a single-layer perceptron. It is a special case of the more general backpropagation algorithm. For a neuron j with activation function $g(x)$ the delta rule for j's ith weight w_{ji} is given by

$$\Delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_i,$$

where α is a small constant called *learning rate*, $g(x)$ is the neuron's activation function, t_j is the target output, h_j is the weighted sum of the neuron's inputs, y_j is the actual output, and x_i is the ith input. It holds $h_j = \sum x_i w_{ji}$ and $y_j = g(h_j)$.

The delta rule is commonly stated in simplified form for a perceptron with a linear activation function as

$$\Delta w_{ji} = \alpha(t_j - y_j)x_i$$

Backpropagation

It is a supervised learning method, and is an implementation of the **Delta rule**. It requires a teacher that knows, or can calculate, the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Backpropagation requires that the activation function used by the artificial neurons (or "nodes") is differentiable.

As the algorithm's name implies, the errors (and therefore the learning) propagate backwards from the output nodes to the inner nodes. So technically speaking, backpropagation is used to calculate the gradient of the error of the network with respect to the network's modifiable weights. This gradient is almost always then used in a simple *stochastic gradient descent algorithm, is a general optimization algorithm, but is typically used to fit the parameters of a machine learning model, to find weights that minimize the error*. Often the term "backpropagation" is used in a more general sense, to refer to the entire procedure encompassing both the calculation of the gradient and its use in stochastic gradient descent. Backpropagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

Backpropagation networks are necessarily multilayer perceptrons (usually with one input, one hidden, and one output layer). In order for the hidden layer to serve any useful function, multilayer networks must have non-linear activation functions for the multiple layers: a

multilayer network using only linear activation functions is equivalent to some single layer, linear network.

Summary of the backpropagation technique:

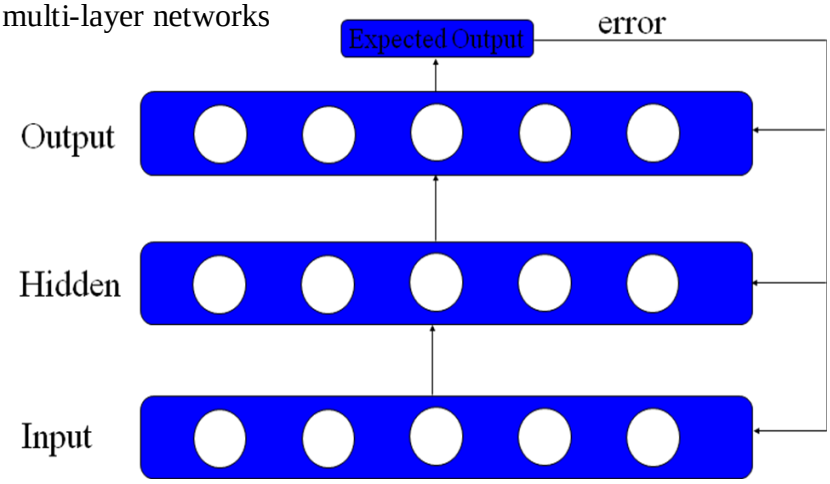
1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a *scaling factor*, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat from step 3 on the neurons at the previous level, using each one's "blame" as its error.

Characteristics:

- A multi-layered perceptron has three distinctive characteristics
  - The network contains one or more layers of hidden neurons
  - The network exhibits a high degree of connectivity
  - Each neuron has a smooth (differentiable everywhere) nonlinear activation function, the most common is the sigmoidal nonlinearity:

$$y_j = \frac{1}{1 + e^{s_j}}$$

- The backpropagation algorithm provides a computational efficient method for training multi-layer networks

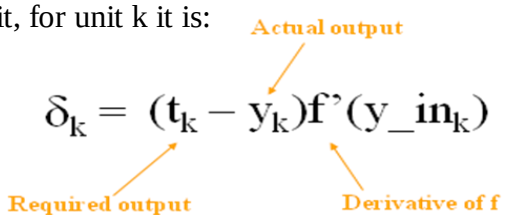


Algorithm:

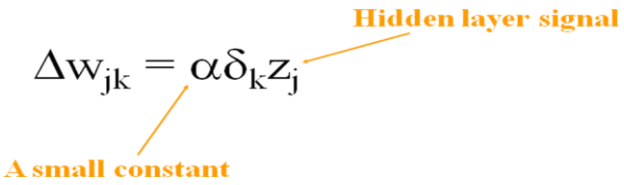
Step 0: Initialize the weights to small random values

Step 1: Feed the training sample through the network and determine the final output

Step 2: Compute the error for each output unit, for unit k it is:



Step 3: Calculate the weight correction term for each output unit, for unit k it is:



Step 4: Propagate the delta terms (errors) back through the weights of the hidden units where the delta input for the j^{th} hidden unit is:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$$

The delta term for j^{th} hidden unit is: $\delta_j = \delta\_in_j f'(z\_in_j)$

Step 5: Calculate the weight correction term for the hidden units: $\Delta w_{ij} = \alpha \delta_j x_i$

Step 6: Update the weights: $w_{ik}(new) = w_{ik}(old) + \Delta w_{ik}$

Step 7: Test for stopping (maximum cylces, small changes, etc)

Note: There are a number of options in the design of a backprop system;

- Initial weights – best to set the initial weights (and all other free parameters) to random numbers inside a small range of values (say –0.5 to 0.5)
- Number of cycles – tend to be quite large for backprop systems
- Number of neurons in the hidden layer – as few as possible

Gelernter Response to Descartes:

David Gelernter uses the term "consciousness" to denote the possession of what philosophers call *qualia*. He's not talking about the differences between the brain states of waking and sleeping animals, and he's not talking about self-consciousness -- an animal's ability to recognize itself in a mirror, or to use the states of its own body (including its brain) as subjects for further cognition. Qualia are the *felt character of experience*. To be conscious, in Gelernter's sense, is to have qualia.

Gelernter divides artificial-intelligence theorists into two camps: cognitivists and anticognitivists. Cognitivists believe that, if human beings have qualia, then a robot that behaves exactly like a human being does, too. Gelernter's initial claims:

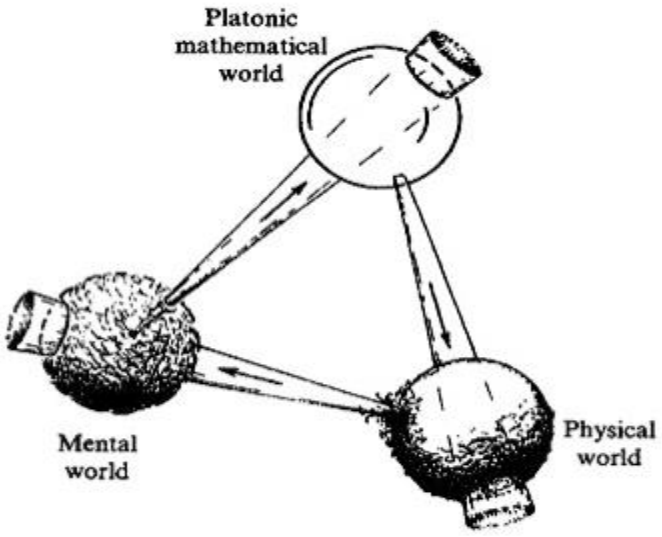
(1) "This subjectivity of mind has an important consequence: *there is no objective way to tell whether some entity is conscious.*"
(2) "we know our fellow humans are conscious."

Human thought, asserts Gelernter, exists along a continuum, spanning from high-focus thinking -- in-depth analytical problem solving -- to low-focus thoughts, consisting of the daydreams and hallucinations that occur when one's mind is wandering. Artificial intelligence research has historically focused on the logical, goal-driven thoughts at the high-end of the spectrum. Gelernter argues that, if the goal truly is to create a computer that can solve problems in a way similar to that of the human mind, then study of unfocused, emotional, low-end thinking must be incorporated into artificial intelligence research, for it is as central to human cognition as logic.

Penrose Response to Descartes:

Roger Penrose has a picture of mind and matter that is not just a relation between logical and physical, but involves three worlds: Platonic, mathematical and physical.

In his more recent *The Road to Reality*, Roger Penrose leaves aside the question of computability in physics, and gets down to the core of physics itself. Once you are past the first thousand pages you can read about his own vision of what it should be like: twistor theory.



Additionally, **Roger Penrose has proposed the idea that the human mind does not use a knowably sound calculation procedure to understand and discover mathematical intricacies.** This would mean that a normal Turing complete computer would not be able to ascertain certain mathematical truths that human minds can.

Penrose presents the argument that human consciousness is non-algorithmic, and thus is not capable of being modeled by a conventional Turing machine-type of digital computer. Penrose hypothesizes that quantum mechanics plays an essential role in the understanding of human consciousness. The collapse of the quantum wavefunction is seen as playing an important role in brain function.

Pinker Response to Descartes:

The mind, for **Steven Pinker** as for almost all other cognitive scientists, is computational. This does *not* mean they think it works just like the computer you're reading this on, but that has representations, which it transforms in a rule-governed, algorithmic way. Moreover, the mind is not a single, general-purpose computer, but a collection of them, of "mental modules" or "mental organs," specialized as to subject matter, each with its own particular learning mechanism ("an instinct to acquire an art," in a phrase Pinker lifts from Darwin). This modularity is evident in studying how children learn, and also from tracing the effects of brain lesions which, if sufficiently localized, impair specific abilities depending on where the brain is hurt, and leave others intact. Just as, baring developmental defects, wounds, or the ravages of disease, all human beings have the same physical organs, we all have the same mental organs, whose general structure is, again, the same from person to person.

By insisting on the complexity of the mind, Pinker claims that;

- thinking is a kind of computation used to work with configurations of symbols,
- the mind is organized into specialized modules or mental organs,
- the basic logic of the modules is contained in our genetic program,
- that natural selection shaped these operations to facilitate replication of genes into the next generation

Pinker thus shows that the computational model of mind is highly significant because it has solved not only philosophical problems, but also started the computer revolution, posed important neuroscience questions, and provided psychology with a very valuable research agenda

Searle Response to Descartes:

Consciousness is a biological phenomenon. We should think of consciousness as part of our ordinary biological history, along with digestion, growth, mitosis and meiosis. However, though consciousness is a biological phenomenon, it has some important features that other biological phenomena do not have. The most important of these is what I (John Searle) have called its `subjectivity'. There is a sense in which each person's consciousness is private to that person, a sense in which he is related to his pains, tickles, itches, thoughts and feelings in a way that is quite unlike the way that others are related to those pains, tickles, itches, thoughts and feelings. This phenomenon can be described in various ways. It is sometimes described as that feature of consciousness by way of which there is something that it's like or something that it feels like to be in a certain conscious state. If somebody asks me what it feels like to give a lecture in front of a large audience I (Searle) can answer that question. But if somebody asks what it feels like to be a shingle or a stone, there is no answer to that question because shingles and stones are not conscious. The point is also put by saying that conscious states have a certain qualitative character; the states in question are sometimes described as **'qualia'**.

In spite of its etymology, consciousness should not be confused with knowledge, it should not be confused with attention, and it should not be confused with self-consciousness. I (Searle) will consider each of these confusions in turn.

Conscious states are caused by lower level neurobiological processes in the brain and are themselves higher level features of the brain. The key notions here are those of *cause* and *feature*. As far as we know anything about how the world works, variable rates of neuron firings in different neuronal architectures cause all the enormous variety of our conscious life. All the stimuli we receive from the external world are converted by the nervous system into one medium, namely, variable rates of neuron firings at synapses. And equally remarkably, these variable rates of neuron firings cause all of the colour and variety of our conscious life. The smell of the flower, the sound of the symphony, the thoughts of theorems in Euclidian geometry -- all are caused by lower level biological processes in the brain; and as far as we know, the crucial functional elements are neurons and synapses.

The first step in the solution of the mind-body problem is: brain processes *cause* conscious processes. This leaves us with the question, what is the ontology, what is the form of existence, of these conscious processes? More pointedly, does the claim that there is a causal relation between brain and consciousness commit us to a dualism of `physical' things and `mental' things? The answer is a definite no. Brain processes cause consciousness but the consciousness they cause is not some extra substance or entity. It is just a higher level feature of the whole system. The two crucial relationships between consciousness and the brain, then, can be summarized as follows: lower level neuronal processes in the brain cause consciousness and consciousness is simply a higher level feature of the system that is made up of the lower level neuronal elements.

John Searle has offered a thought experiment known as the **Chinese Room** that demonstrates this problem. Imagine that there is a man in a room with no way of communicating to anyone or anything outside of the room except for a piece of paper that is passed under the door. With the paper, he is to use a series of books provided to decode and "answer" what is on the paper. The symbols are all in Chinese, and all the man knows is where to look in the books, which then tell him what to write in response. It just so happens that this generates a conversation that the Chinese man outside of the room can actually understand, but can our man in the room really be said to understand it? This is essentially what the computational theory of mind presents us with; a model in which the mind simply decodes symbols and outputs more symbols. It is argued that perhaps this is not real learning or thinking at all. However, it can be argued in response to this that it is the man and the paper together that understand Chinese, albeit in a rudimentary way due to the rudimentary nature of the system; as opposed to if the man learned Chinese, which would create a sophisticated system of communicating Chinese.

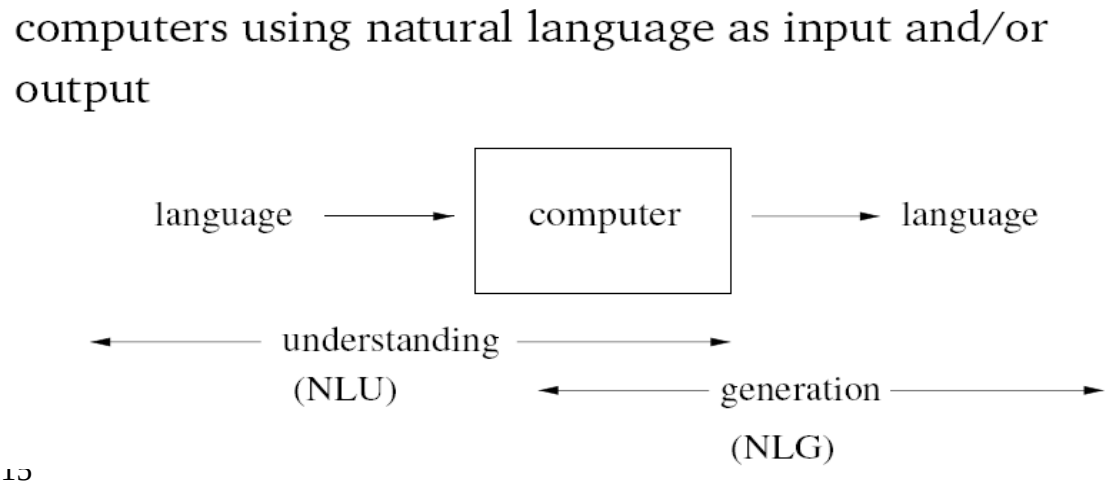
Natural Language Processing:

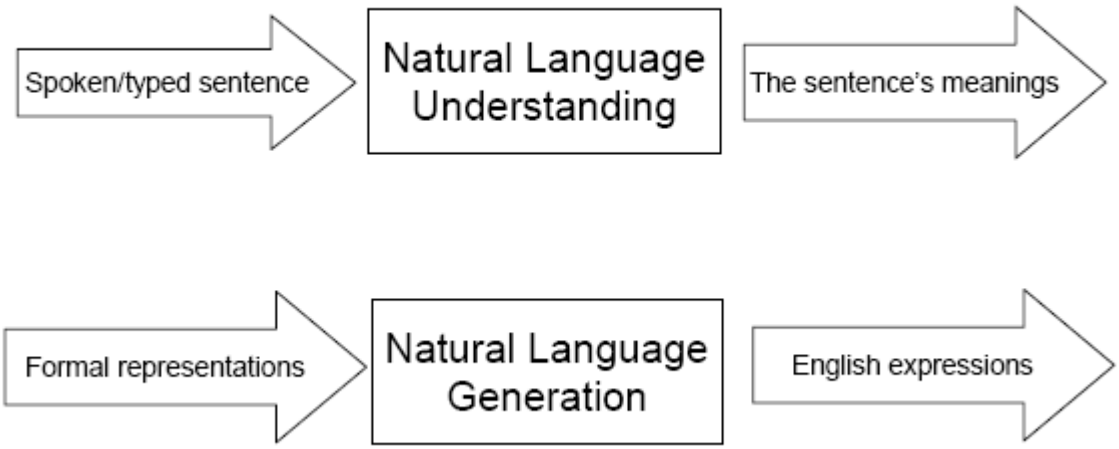
Perception and communication are essential components of intelligent behavior. They provide the ability to effectively interact with our environment. Humans perceive and communicate through their five basic senses of sight, hearing, touch, smell and taste, and their ability to generate meaningful utterances. Developing programs that understand a natural language is a difficult problem. Natural languages are large. They contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. Also, there is much ambiguity in a natural language. Many words have several meanings and sentences can have different meanings in different contexts. This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI.

Developing programs to understand natural language is important in AI because a natural form of communication with systems is essential for user acceptance. AI programs must be able to communicate with their human counterparts in a natural way, and natural language is one of the most important mediums for that purpose. So, Natural Language Processing (NLP) is the field that deals with the computer processing of natural languages, mainly evolved by people working in the field of Artificial Intelligence.

Natural Language Processing (NLP), is the attempt to extract the fuller meaning representation from the free text. Natural language processing is a technology which involves converting spoken or written human language into a form which can be processed by computers, and vice versa. Some of the better-known applications of NLP include:

- **Voice recognition software** which translates speech into input for word processors or other applications;
- **Text-to-speech synthesizers** which read text aloud for users such as the hearing-impaired;
- **Grammar and style checkers** which analyze text in an attempt to highlight errors of grammar or usage;
- **Machine translation systems** which automatically render a document such as a web page in another language.





Natural Language Generation:

"Natural Language Generation (NLG), also referred to as text generation, is a subfield of natural language processing (NLP; which includes computational linguistics)

Natural Language Generation (NLG) is the natural language processing task of generating natural language from a machine representation system such as a knowledge base or a logical form.

In a sense, one can say that an NLG system is like a translator that converts a computer based representation into a natural language representation. However, the methods to produce the final language are very different from those of a compiler due to the inherent expressivity of natural languages.

NLG may be viewed as the opposite of natural language understanding. The difference can be put this way: whereas in natural language understanding the system needs to disambiguate the input sentence to produce the machine representation language, in NLG the system needs to make decisions about how to put a concept into words.

The different types of generation techniques can be classified into four main categories:

- Canned text systems constitute the simplest approach for single-sentence and multi-sentence text generation. They are trivial to create, but very inflexible.
- Template systems, the next level of sophistication, rely on the application of pre-defined templates or schemas and are able to support flexible alterations. The template approach is used mainly for multi-sentence generation, particularly in applications whose texts are fairly regular in structure.

- Phrase-based systems employ what can be seen as generalized templates. In such systems, a phrasal pattern is first selected to match the top level of the input, and then each part of the pattern is recursively expanded into a more specific phrasal pattern that matches some subportion of the input. At the sentence level, the phrases resemble phrase structure grammar rules and at the discourse level they play the role of text plans.
- Feature-based systems, which are as yet restricted to single-sentence generation, represent each possible minimal alternative of expression by a single feature. Accordingly, each sentence is specified by a unique set of features. In this framework, generation consists in the incremental collection of features appropriate for each portion of the input. Feature collection itself can either be based on unification or on the traversal of a feature selection network. The expressive power of the approach is very high since any distinction in language can be added to the system as a feature. Sophisticated feature-based generators, however, require very complex input and make it difficult to maintain feature interrelationships and control feature selection.

Many natural language generation systems follow a hybrid approach by combining components that utilize different techniques.

Natural Language Understanding:

Developing programs that understand a natural language is a difficult problem. Natural languages are large. They contain infinity of different sentences. No matter how many sentences a person has heard or seen, new ones can always be produced. Also, there is much ambiguity in a natural language. Many words have several meaning such as can, bear, fly, bank etc, and sentences have different meanings in different contexts.

Example :- a *can* of juice. I *can* do it.

This makes the creation of programs that understand a natural language, one of the most challenging tasks in AI. Understanding the language is not only the transmission of words. It also requires inference about the speakers' goal, knowledge as well as the context of the interaction. We say a program understand natural language if it behaves by taking the correct or acceptable action in response to the input. A word functions in a sentence as a part of speech. Parts of the speech for the English language are nouns, pronouns, verbs, adjectives, adverbs, prepositions, conjunctions and interjections. Three major issues involved in understanding language.

- A large amount of human knowledge is assumed.
- Language is pattern based, phonemes are components of the words and words make phrases and sentences. Phonemes, words and sentences order are not random.
- Language acts are the product of agents (human or machine).

Levels of knowledge used in Language Understanding

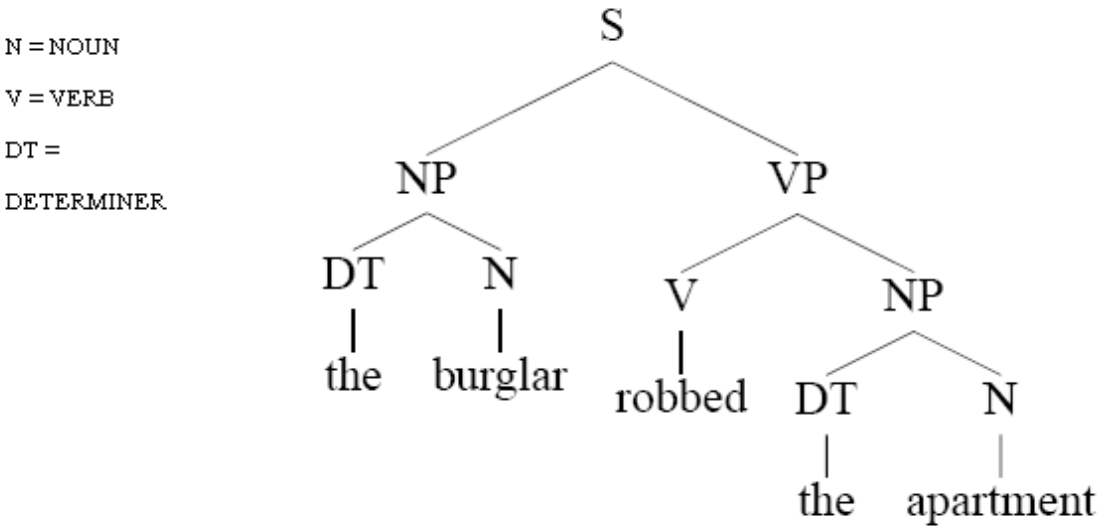
A language understanding knowledge must have considerable knowledge about the structures of the language including what the words are and how they combine into phrases and sentences. It must also know the meanings of the words and how they contribute to the meanings of the sentence and to the context within which they are being used. The component forms of knowledge needed for an understanding of natural languages are sometimes classified according to the following levels.

- **Phonological**
    - Relates sound to the words we recognize. A phoneme is the smallest unit of the sound. Phones are aggregated to the words.
- **Morphological**
    - This is lexical knowledge which relates to the word construction from basic units called morphemes. A morpheme is the smallest unit of meaning. Eg:- *friend* + ly = friendly
- **Syntactic**
    - This knowledge relates to how words are put together or structure red together to form grammatically correct sentences in the language.
- **Semantic**
    - This knowledge is concerned with the meanings of words and phrases and how they combine to form sentence meaning.
- **Pragmatic**
    - This is high – level knowledge which relates to the use of sentences in different contexts and how the context affects the meaning of the sentence.
- **World**
    - Includes the knowledge of the physical world, the world of human social interaction, and the roles of goals and intentions in communication.

Basic Parsing Techniques

Before the meaning of a sentence can be determined, the meanings of its constituent parts must be established. This requires knowledge of the structure of the sentence, the meaning of the individual words and how the words modify each other. The process of determining the

syntactical structure of a sentence is known as parsing. Parsing is the process of analyzing a sentence by taking it apart word – by – word and determining its structure from its constituent parts and sub parts. The structure of a sentence can be represented with a syntactic tree. When given an input string, the lexical parts or terms (root words), must first be identified by type and then the role they play in a sentence must be determined. These parts can be combined successively into larger units until a complete tree has been computed.



Noun Phrases (NP): "the burglar", "the apartment"

Verb Phrases (VP): "robbed the apartment"

Sentences (S): "the burglar robbed the apartment"

To determine the meaning of a word, a parser must have access to a lexicon. When the parser selects the word from the input stream, it locates the world in the lexicon and obtains the word's possible functions and features, including the semantic information.

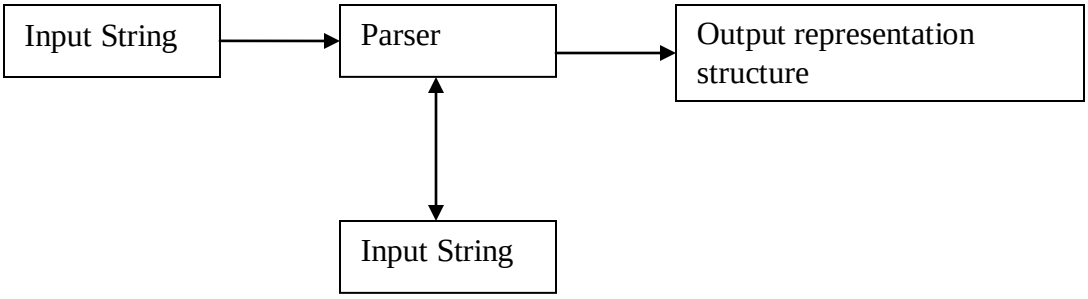


Figure :- Parsing an input to create an output structure

Lexeme (Lexicon) & word forms:

The distinction between these two senses of "word" is arguably the most important one in morphology. The first sense of "word", the one in which *dog* and *dogs* are "the same word", is called a lexeme. The second sense is called *word form*. We thus say that *dog* and *dogs* are different forms of the same lexeme. *Dog* and *dog catcher*, on the other hand, are different lexemes, as they refer to two different kinds of entities. The form of a word that is chosen conventionally to represent the canonical form of a word is called a lemma, or citation form.

A lexicon defines the words of a language that a system knows about. This is includes common words and words that are specific to the domain of the application. Entries include meanings for each word and its syntactic and morphological behavior.

Morphology:

Morphology is the identification, analysis and description of the structure of words (words as units in the lexicon are the subject matter of lexicology). While words are generally accepted as being (with clitics) the smallest units of syntax, it is clear that in most (if not all) languages, words can be related to other words by rules. For example, English speakers recognize that the words *dog*, *dogs*, and *dog catcher* are closely related. English speakers recognize these relations from their tacit knowledge of the rules of word formation in English. They infer intuitively that *dog* is to *dogs* as *cat* is to *cats*; similarly, *dog* is to *dog catcher* as *dish* is to *dishwasher* (in one sense). The rules understood by the speaker reflect specific patterns (or regularities) in the way words are formed from smaller units and how those smaller units interact in speech. In this way, morphology is the branch of linguistics that studies patterns of word formation within and across languages, and attempts to formulate rules that model the knowledge of the speakers of those languages.

Morphological analysis is the process of recognizing the suffixes and prefixes that have been attached to a word.

We do this by having a table of affixes and trying to match the input as:
prefixes +root + suffixes.

- For example: adjective + ly -> adverb. E.g.: [Friend + ly]=friendly
- We may not get a unique result.
- "-s, -es" can be either a plural noun or a 3ps verb
- "-d, -ed" can be either a past tense or a perfect participle

Morphological Information:

- Transform part of speech
  - *green, greenness (adjective, noun)*
  - *walk, walker (verb, noun)*

- Change features of nouns
  - *boat, boats (singular, plural)*

- Bill slept , Bill's bed
  - (subjective case, possessive case)

- Change features of verbs
  - Aspect
    - *I walk. I am walking.* (present, progressive)
  - Tense
    - *I walked. I will walk. I had been walking. (past, future, past progressive)*
  - Number and person
    - *I walk. They walk. (first person singular, third person plural)*

Syntactic Analysis:

Syntactic analysis takes an input sentence and produces a representation of its grammatical structure. A grammar describes the valid parts of speech of a language and how to combine them into phrases. The grammar of English is nearly context free.

A computer grammar specifies which sentences are in a language and their parse trees. A parse tree is a hierarchical structure that shows how the grammar applies to the input. Each level of the tree corresponds to the application of one grammar rule.

It is the starting point for working out the meaning of the whole sentence. Consider the following two sentences.

1. "The dog ate the bone."
2. "The bone was eaten by the dog."

Understanding the structure (via the syntax rules) of the sentences help us work out that it's the bone that gets eaten and not the dog. Syntactic analysis determines possible grouping of words in a sentence. In other cases there may be many possible groupings of words. Consider the sentence "John saw Mary with a telescope". Two different readings based on the groupings.

1. John saw (Mary with a telescope).
2. John (saw Mary with a telescope).

A sentence is syntactically ambiguous if there are two or more possible groupings. Syntactic analysis helps determining the meaning of a sentence by working out possible word structure. Rules of syntax are specified by writing a *grammar* for the language. A parser will check if a sentence is correct according to the grammar. It returns a representation (parse tree) of the sentence's structure. A grammar specifies allowable sentence structures in terms of basic categories such as noun and verbs. A given grammar, however, is unlikely to cover all possible grammatical sentences. Parsing sentences is to help determining their meanings, not just to check that they are correct. Suppose we want a grammar that recognizes sentences like the following.

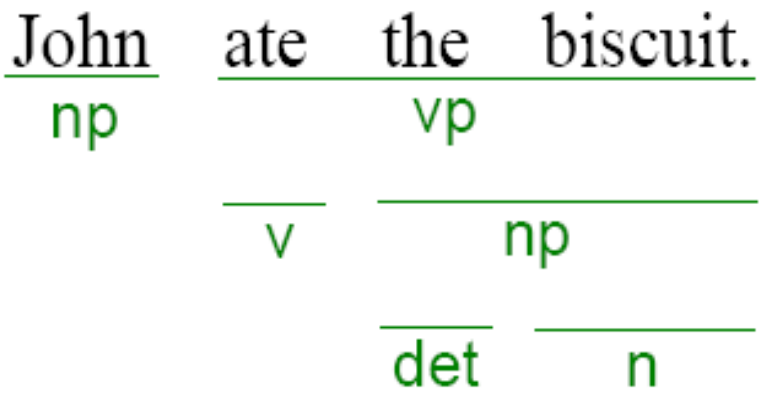
John ate the biscuit.
The lion ate the zebra.
The lion kissed John

But reject incorrect sentences such as

Ate John biscuit the.
Zebra the lion the ate.
Biscuit lion kissed.

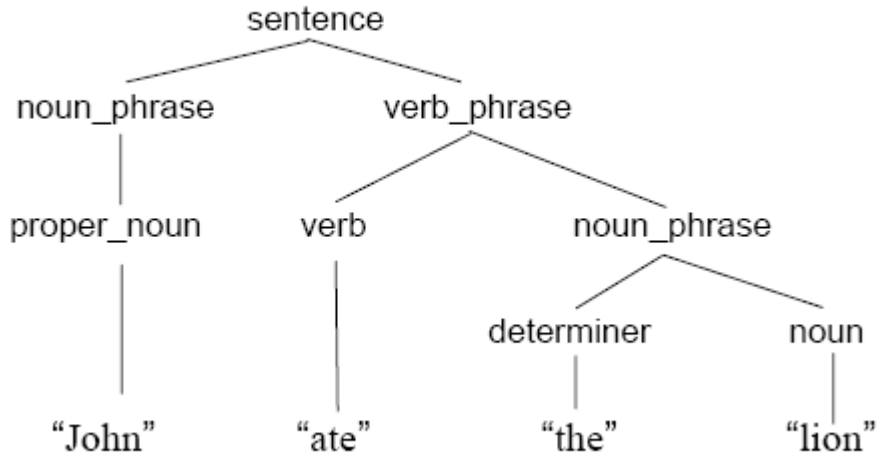
A simple grammar that deals with this is given below

sentence --> noun_phase, verb phrase.
noun_phrase --> proper_noun.
noun_phrase --> determiner, noun.
verb_phrase --> verb, noun_phrase.
proper_noun --> [mary].
proper_noun --> [john].
noun --> [zebra].
noun --> [biscuit].
verb --> [ate].
verb --> [kissed].
determiner --> [the].



Incorrect sentences like "biscuit lion kissed" will be excluded by the grammar.

- A ***parse trees*** illustrates the syntactic structure of the sentence.



Semantic Analysis:

Semantic analysis is a process of converting the syntactic representations into a meaning representation.

This involves the following tasks:

- Word sense determination
- Sentence level analysis
- Knowledge representation

- Word sense

Words have different meanings in different contexts.

Mary had a bat in her office.

- bat = `a baseball thing'

- bat = `a flying mammal'

- Sentence level analysis

Once the words are understood, the sentence must be assigned some meaning

I saw an astronomer with a telescope.

- Knowledge Representation

Understanding language requires lots of knowledge**.**

- Using predicate logic, for example, one can represent sentences like

"John likes Mary" $likes(john,mary)$

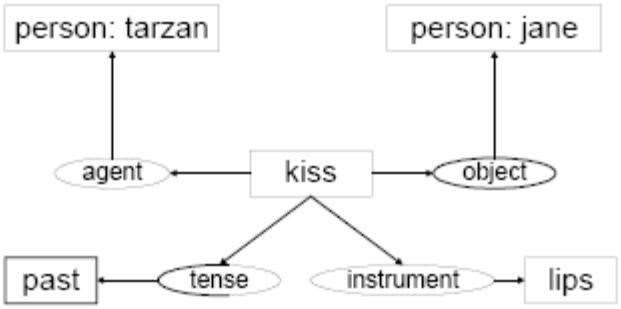
"The man likes Mary" $man(m1) \wedge likes(m1,mary)$

"A man likes Mary" $\exists X(man(X) \wedge likes(X,mary)$

"A tall bearded man likes Mary"

$\exists X(man(X) \wedge tall(X) \wedge bearded(X) \wedge likes(X,mary)$

- Using semantic net, one can represent sentence "Tarzan kissed Jane" as



Parameters in Natural Language Processing:

- Auditory Inputs
- Segmentation
- Syntax Structure
- Semantic Structure
- Pragmatic Analysis

- Auditory Inputs:

Three of our five senses – sight, hearing and touch – are used as *major inputs*. These are usually referred to as the visual, auditory and tactile inputs respectively. They are sometimes called input channels; however, as previously mentioned, the term "channel" is used in various ways, so I will avoid it.

In the fashion of video devices, audio devices are used to either capture or create sound. In some cases, an audio output device can be used as an input device, in order to capture produced sound.

- Microphone
- MIDI keyboard or other digital musical instrument

- Segmentation:

Text segmentation is the process of dividing written text into meaningful units, such as words, sentences, or topics. The term applies both to mental processes used by humans when reading text, and to artificial processes implemented in computers, which are the subject of natural language processing. The problem is non-trivial, because while some written languages have explicit word boundary markers, such as the word spaces of written English and the distinctive initial, medial and final letter shapes of Arabic, such signals are sometimes ambiguous and not present in all written languages.

Word segmentation is the problem of dividing a string of written language into its component words. In English and many other languages using some form of the Latin alphabet, the space is a good approximation of a word delimiter. (Some examples where the space character alone may not be sufficient include contractions like *can't* for *can not*.)

However the equivalent to this character is not found in all written scripts, and without it word segmentation is a difficult problem. Languages which do not have a trivial word segmentation process include Chinese, Japanese, where sentences but not words are delimited, and Thai, where phrases and sentences but not words are delimited.

In some writing systems however, such as the Ge'ez script used for Amharic and Tigrinya among other languages, words are explicitly delimited (at least historically) with a non-whitespace character.

Word splitting is the process of parsing concatenated text (i.e. text that contains no spaces or other word separators) to infer where word breaks exist.

Sentence segmentation is the problem of dividing a string of written language into its component sentences. In English and some other languages, using punctuation, particularly the full stop character is a reasonable approximation. However, even in English this problem is not trivial due to the use of the full stop character for abbreviations, which may or may not also terminate a sentence. For example *Mr.* is not its own sentence in "*Mr. Smith went to the shops in Jones Street."* When processing plain text, tables of abbreviations that contain periods can help prevent incorrect assignment of sentence boundaries. As with word segmentation, not all written languages contain punctuation characters which are useful for approximating sentence boundaries.

Other segmentation problems: Processes may be required to segment text into segments besides words, including morphemes (a task usually called morphological analysis), paragraphs, topics or discourse turns.

A document may contain multiple topics, and the task of computerized text segmentation may be to discover these topics automatically and segment the text accordingly. The topic boundaries may be apparent from section titles and paragraphs. In other cases one needs to use techniques similar to those used in document classification. Many different approaches have been tried.

- Syntax Structure:

Same concept as in the syntactic analysis above

- Semantic Structure:

Same concept as in the semantic analysis above

- Pragmatic Analysis:

This is high level knowledge which relates to the use of sentences in different contexts and how the context affects the meaning of the sentences. It is the study of the ways in which language is used and its effect on the listener. Pragmatic comprises aspects of meaning that depend upon the context or upon facts about real world.

Pragmatics – Handling Pronouns

Handling pronouns such as “he”, “she” and “it” is not always straight forward. Let us see the following paragraph.

“John buys a new telescope. He sees Mary in the distance. He gets out his telescope. He looks at her through it”.

Here, "*her*" refers to *Mary* who was not mentioned at all in the previous sentences. John's telescope was referred to as "*a new telescope*", "*his telescope*" and "*it*".

Let us see one more example

"When is the next flight to Sydney?"
"Does it have any seat left?"

Here, "*it*", refers to a particular flight to Sydney, not Sydney itself.

Pragmatics – Ambiguity in Language

A sentence may have more than one structure such as

"I saw an astronomer with a telescope."

This English sentence has a prepositional phrase "*with a telescope*" which may be attached with either with verb to make phrase "*saw something with telescope*" or to object noun phrase to make phrase "*a astronomer with a telescope*". If we do first, then it can be interpreted as "*I saw an astronomer who is having a telescope*", and if we do second, it can be interpreted as "*Using a telescope I saw an astronomer*".

Now, to remove such ambiguity, one possible idea is that we have to consider the context. If the knowledge base (KB) can prove that whether the telescope is with astronomer or not, then the problem is solved.

Next approach is that; let us consider the real scenario where the human beings communicate. If A says the same sentence "*I saw an astronomer with a telescope*." To B, then in practical, it is more probable that, B (listener) realizes that "*A has seen astronomer who is having a telescope*". It is because, normally, the word "*telescope*" belongs to "*astronomer*", so it is obvious that B realizes so.

If A has says that "*I saw a lady with a telescope*." In this case, B realizes that "*A has seen the lady using a telescope*", because the word "*telescope*" has not any practical relationship with "*lady*" like "*astronomer*".

So, we may be able to remove such ambiguity, by defining a data structure, which can efficiently handle such scenario. This idea may not 100% correct but seemed more probable.