CHAPTER – 1

A BRIEF HISTORICAL SURVEY AND

AN

OUTLINE OF THE PROGRAM
Estelar
Estelar
CHAPTER 1

A BRIEF HISTORICAL SURVEY AND AN OUTLINE OF THE PROGRAM

1.1 Introduction

Human languages are generally described by the term “natural language,” because they all are somehow spontaneous collective creations to which one cannot assign a precise date of birth. Natural languages are opposed primarily to “artificial languages” or “formal languages” including computer programming languages or mathematical logic. Behind the apparent diversity of human languages, linguists try to track down common functions and universal structure. Linguists are not necessarily polyglots; they seek more to understand the principles that govern language to multiply the knowledge they have of some of them. Contrary to popular opinion, the language is not prescriptive: it does not say how to speak or to write well. Natural languages are living systems that change, interact, and evolve. Linguists are content to watch as they speak and write without trying to control or limit their natural evolution. Computer science and Mathematics through logical modeling play a very significant role in the study of human languages through the discipline of natural language processing. This discipline implies automatic processing of information.

Natural language processing (NLP) is a combinatory discipline, which combines linguistics, computer science, and artificial intelligence in attempt to create an interactive system between human being and computer. It is a field that explores how natural language, like the English language, can be processed mechanically utilizing computer systems. NLP researches the design of mathematical models of natural language structures and their implementation. Human aspiration to make computers process natural language, is as old as the idea of computers themselves. NLP is not an isolated field of CS research but has emerged in several other sciences, such as computational linguistics in linguistics, speech recognition (SR) in electrical engineering and computational psycholinguistics in psychology.
If we study NLP from the historical perspectives, we observe that in the 1950s, solutions to NLP problems followed an empiricist approach, such as conceiving MT (Machine Translation) as the problem of reconstructing a message transmitted over a noisy channel using channel characteristics and stochastic techniques. From around the 1960s to 1980s of the last century, this paradigm shifted to assuming that human language faculties directly result from extensive knowledge and inference mechanisms hardwired into the human brain. Consequently, the rationalist paradigm dominated the cognition and mindset of the NLP domain leading to embracing symbol processing techniques to treat NLP tasks, such as NLU (Natural language understanding) systems based on pattern matching and syntax generation and acceptance via the grammar formalism and parser. By the 1980s, the availability of large-scale annotated natural language corpora (text, speech), increasing raw computing power and recognition of artificial intelligence (AI) learning techniques, have restored attention and confidence in the empiricist paradigm. Consequently, problems like recognizing speech, determining the functional role of words and translating or summarizing text were attacked by developing probabilistic models trained on problem-specific data pools (e.g., reports, email correspondence).

Lately, two trends have emerged in the development of NLP solutions. Firstly, there has been a focus on unsupervised learning approaches to avoid upfront costs associated with the design of annotated corpora used to implement models relying on supervised learning. Secondly, there has been the development of hybrid NLP systems, i.e. incorporating elements from empiricist and rationalist approach.

We take the ultimate goal of natural language processing (NLP) to be the ability to use natural languages as effectively as humans do. Natural language, whether spoken, written, or typed, is the most natural means of communication between humans, and the mode of expression of choice for most of the documents they produce. As computers play a larger role in the preparation, acquisition, transmission, monitoring, storage, analysis, and transformation of information, endowing them with the ability to understand and generate information expressed in natural languages becomes more and more necessary. Some tasks currently performed by humans cannot be automated without endowing computers with natural language processing capabilities, and these provide two major challenges to NLP systems:
1. Documentation of spoken languages, with applications, in such areas as in science, diplomacy, multinational commerce, and intelligence.

2. To enable Computers to understand input in more than one language, provide output in more than one language, and translate between languages.

Research into Natural Language Processing - the use of computers to process language - has developed over the last couple of decades into one of the most vigorous and interesting areas of current work on language and communication. One such field is of Prolog and natural language analysis, is especially well-suited for Natural Language Processing as the programming language used is Prolog, can be used by those with little or no background in computing. Due to the 'user friendly' nature of Prolog, simple yet effective programs may be written from an early stage and therefore Prolog has continued to attract a great deal of interest in the computer science community, and has turned out to be a basis for an important new generation of programming languages and systems for Artificial Intelligence. Prolog can be used to solve actual problems in syntax, parsing, and semantic interpretation which can be defined as the process of finding documents whose content is relevant to the query need of a user and also relevant to the user defined system. It can also be defined as the science of searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web. It is the outcome of Logic programming, an important new method of computer programming resulting from recent research in artificial intelligence and computer science that has proved to be especially appropriate for solving problems in natural-language processing.

A linguistic competence model aims at characterizing a person’s knowledge of a language while a linguistic performance model aims at describing the actual production and perception of natural language sentences in concrete situation. This dichotomy has become the methodological paradigm for all formal linguistic theories. It is assumed that the primary goal of linguistics is the development of a theory of language competence. Linguistic theory has adopted the formal language of logic and mathematic as its paradigm examples: a language we viewed as a well-defined infinite of sentence/meaning pairs that we explicitly characterized by a consistent and non-redundant system of formal rules (a” competence grammar).
It is conjectured that human mind actually employ such rule systems in producing and comprehending new utterances. At the same time, linguistic theory acknowledges explicitly that a competence grammar alone cannot account for all aspects of human language: a person’s language performance is also influenced by several other mental properties that do not belong to the core-business of linguistics.

A definite clause grammar (DCG) is a way of expressing grammar, either for natural or formal languages, in a logic programming language such as Prolog. It is closely related to the concept of attribute/affix grammars from which Prolog was originally developed. DCGs are usually associated with Prolog, but similar languages such as Mercury also include DCGs. They are called definite clause grammars because they represent a grammar as a set of definite clauses in first-order logic. The term DCG refers to the specific type of expression in Prolog and other similar languages; not all ways of expressing grammars using definite clauses are considered DCGs. However, all of the capabilities or properties of DCGs will be the same for any grammar that is represented with definite clauses in essentially the same way as in Prolog.

Prolog is a logical and a declarative programming language. The name itself, Prolog, is short for PROgramming in LOGic. The "first" Prolog was "Marseille Prolog" based on work by Colmerauer [44]. The major influence on the nature of this first Prolog was that it was designed to facilitate natural language processing. It is the major example of a fourth generation programming language supporting the declarative programming paradigm. The Japanese Fifth-Generation Computer Project, announced in 1981, adopted Prolog as a development language, and thereby focused considerable attention on the language and its capabilities. The programs in this tutorial were written in "standard" (University of) Edinburgh Prolog. The other major kind of Prolog is the PrologII family of Prologs which are the descendants of Marseille Prolog. The reference to Giannesini, et.al. [97] uses a version of PrologII. There are differences between these two varieties of Prolog; part of the difference is syntax, and part is semantics. However, students who learn either kind of Prolog can easily adapt to the other kind.

Prolog is the vital area of natural language processing (NLP) and can be defined as the process of finding documents whose content is relevant to the query need of a user and also relevant to the user defined system.
Chapter 1

It can also be defined as the science of searching for documents, for information within documents, and for metadata about documents, as well as that of searching relational databases and the World Wide Web. A language in which we could conveniently express pattern matching rules over the parse trees and other annotations (such as named entity recognition results), and a technology that could execute these rules very efficiently. We found that Prolog was the ideal choice for the language due to its simplicity and expressiveness. The information in the parse is easily converted into Prolog facts.

A linguistic competence model aims at characterizing a person’s knowledge of a language while a linguistic performance model aims at describing the actual production and perception of natural language sentences in concrete situation. This dichotomy has become the methodological paradigm for all formal linguistic theories. It is assumed that the primary goal of linguistics is the development of a theory of language competence. Linguistic theory has adopted the formal language of logic and mathematic as its paradigm examples: a language we viewed as a well-defined infinite of sentence/meaning pairs that we explicitly characterized by a consistent and non-redundant system of formal rules (a” competence grammar). It is conjectured that human mind actually employ such rule systems in producing and comprehending new utterances. At the same time, linguistic theory acknowledges explicitly that a competence grammar alone cannot account for all aspects of human language: a person’s language performance is also influenced by several other mental properties that do not belong to the core-business of linguistics.
A definite clause grammar (DCG) is a way of expressing grammar, either for natural or formal languages, in a logic programming language such as Prolog. It is closely related to the concept of attribute/affix grammars from which Prolog was originally developed. DCGs are usually associated with Prolog, but similar languages such as Mercury also include DCGs. They are called definite clause grammars because they represent a grammar as a set of definite clauses in first-order logic. The term DCG refers to the specific type of expression in Prolog and other similar languages; not all ways of expressing grammars using definite clauses are considered DCGs. However, all of the capabilities or properties of DCGs will be the same for any grammar that is represented with definite clauses in essentially the same way as in Prolog.

This Thesis is an introduction to elementary computational linguistics from the point of view of logic programming. The connection between computational linguistics and logic programming has both formal and utilitarian aspects. On the formal side, we shall explore the restricted logical language of definite clauses as a means of expressing linguistic analyses and representations. On the utilitarian side, we shall introduce the Logic-programming language Prolog, whose backbone is the definite clause formalism, as a tool for implementing the basic components of natural-language-processing systems. The main goal of thesis is to enable the user/reader/person to acquire, as quickly as possible, a working understanding of basic computational linguistic and logic programming concepts. To achieve this goal, the thesis is organized around specific concepts and programming techniques, with examples supported by working programs. Most of the problems involve programming and also supplement the material in the main research paper. The development of logic programming has been closely tied to the search for computational formalisms for expressing syntactic and semantic analyses of natural-language sentences. One of the main purposes in developing Prolog was to create a language in which phrase structure and semantic-interpretation rules for a natural-language question-answering system could be easily expressed. Phrase-structure rules for a language state how phrases of given types combine to form larger phrases in the language. Natural language analysis is interdisciplinary, based on computer science, mathematics, library science, information science, information architecture, cognitive psychology, linguistics, and statistics. Research in Prolog language is aimed at designing
Chapter 1

and evaluating systems that try to fulfill a user’s information need. Often, this is a user posing a query to a database. For instance, it can present a list of documents presumably containing the information the user is looking for (ad-hoc retrieval), or it can directly answer a user’s question by generating natural language sentences or extracting sentences from the database (Question-Answering).

One of the main goals of the development of symbolic logic has been to capture the notion of logical consequence with formal, mechanical, means. If the conditions for a certain class of problems can be formalized within a suitable logic as a set of premises, and if a problem to be solved can be stated as a sentence in the logic, then a solution might be found by constructing a formal proof of the problem statement from the premises. Linguistic rules can be put in the format of Prolog, the basis for the usefulness of definite clauses in language analysis. This fact has not only theoretical but also practical importance, in that linguistic rules encoded as definite clauses can be run directly by Prolog, providing an efficient and direct computational realization of grammars and interpretation rules.

Logic programming languages in general, and Prolog in particular, differ from conventional programming languages (such as Pascal or FORTRAN) in several important ways. First of all, Prolog can be thought of as a largely declarative language; that is, a Prolog program can be viewed as stating what is computed, independent of a particular method for computation. Pascal, on the other hand, is procedural, in that what a Pascal program computes is definable only in terms of how it performs the computation. Prolog programs are nondeterministic, since several elements can be in a particular relation to a given element. Because conventional languages are geared toward functions, that is, relations in which one element is uniquely defined in terms of the others, computation proceeds deterministically in such languages. In this Thesis we generally introduce finite
state automata and how they can be used to recognize and generate (formal) languages, explain the difference between deterministic and non-deterministic automata, say something about the limitations of finite state methods, and finally, we will write a simple PROLOG recognizer/generator for finite state automata to introduce like Very different than other programming languages. A new paradigm (e.g., imperative, functional, object-oriented, parallel, etc.) Prolog does not specify "how" to resolve a problem (algorithm). It
also specify what is true (facts) or how we can prove it is true (rules) and freely available SWI-Prolog. Because of its declarative semantics, built-in search, and pattern matching, Prolog provides an important tool for programs that process natural language. Indeed, natural language understanding was one of Prolog’s earliest applications. As we will see with many examples in this Thesis, we can write natural language grammars directly in Prolog, for example, context-free, context-sensitive, recursive descent semantic network, as well as stochastic parsers. Semantic representations are also easy to create in Prolog. The grammar rules specify a subset of legitimate sentences of English; the Prolog grammar code represents these specifications. The interpreter is asked questions about them and the answer is a function of the specifications and the question asked. Since there are no constraints enforced across the sub trees that make up the full parse of a sentence.

A context-sensitive parser in Prolog desire to have proper noun–verb agreement enforced by the grammar rules themselves. In the dictionary entry for each word its singular or plural form can be noted as such. Then in the grammar specifications for noun phrase and verb phrase a further parameter is used to signify the Number of each phrase. This enforces the constraint that a singular noun has to be associated with a singular verb. Similar constraints for article–noun combinations can also be enforced. The technique we are using in this Thesis is constraining sentence components by enforcing variable bindings across the sub trees of the parse of the sentence. Natural language is complex and ambiguous, and the pattern matching rules that we require are therefore more complex. The Prolog language is recognized to be an excellent solution for the problem of pattern matching and all problems that involve a depth-first search and backtracking Leon Sterling et al.[148] and Michael A. et al. [169], Although simple, the Prolog language is very expressive allowing recursive rules to represent reach ability in parse trees and the operation of negation-as-failure to check the absence of conditions. Prior to our decision to use Prolog for this task, we had implemented custom pattern matching frameworks over parses. Chomsky Noam [45] brought a revolutionary approach to linguistics. He believed that language users utilize underlying grammatical generators, rather than simply repeating or responding to outside influences. This has led to modern linguistics searching for these
underlying characteristics. Linguistics research methodology by Tiffany Silver berg [171] describes the structure used in the work done by linguists worldwide. World Languages Network AMARAUNA is a project, which aims to promote the linguistic rights, and its goal is to strengthen relations among people and institutions that have realized about the importance of the heritage and diversity of the languages of the world. The study of the linguistic landscape in its own right is a relatively recent development.

The Question-Answering (QA) problem requires a machine to go beyond just matching keywords in documents, which is what a web-search engine does, and correctly interpret the question to figure out what is being asked. The QA system also needs to find the precise answer without requiring the aid of a human to read through the returned documents. To address these challenges, the research team at IBM developed software architecture called DeepQA, on which Watson is implemented. The DeepQA architecture assumes and pursues multiple interpretations of the question, generates many plausible answers or hypotheses, collects evidence for these hypotheses, and evaluates the evidence to determine if it supports or refutes those hypotheses David Ferrucci and Watson et al.[59]. Watson’s NLP begins by applying a parser Michael C. McCord [176] that converts each text sentence into a more structured form: a tree that shows both surface structure and deep, logical structure. An ontology-based architecture for editing and maintaining web portals in an easier way has been developed by Fellbaum, C., Palmer et al. [79] and the Community Web Portals project has been undertaken by Staab et al., 2003. In this field of Research Green’s [92] observations linking computation to deduction had no effective realization until the development of more goal-oriented linear resolution proof procedures, in particular Kowalski and Kuehner’s SL resolution [135]. This development allowed Kowalski [134] to suggest a general approach to goal-directed deductive computation based on appropriate control mechanisms for resolution theorem proverbs and the further specialization of SL resolution to Horn clauses, and the corresponding procedural interpretation of Horn clauses, was first described in principle by Kowalski [134] [135]. It eventually became clear that a particular kind of linear
resolution restricted to definite clauses had just the right goal-directness and efficiency,

and also enough expressive power for linguistic rules and some important aspects of the question-answering problem, this approach was first described as a tool for natural-language
processing applications by Colmerauer et al. [48]. The resulting deductive system, supplemented with a few other computational devices, was the first Prolog system, known as “Marseille Prolog”, this first detailed description of Prolog was the language manual for the Marseille Prolog interpreter Roussel et al. [216].

Prolog was originally developed for natural-language processing, besides this the original piece of work was developed by in the form of Prolog applications Colmerauer et al., [49] other early influential work includes systems by Pasero [197] and Dahl et al. [58]. The research paper entitled “collection Readings in Natural Language Processing” where writer dignify the relationship between Prolog and Natural Language Resources was first developed by Grosz et al. [99] and that paper shall also cover a wide variety of topics in prolog and natural-language formalism. Besides natural-language processing, logic programming and Prolog have been used in many other application areas, particularly in artificial intelligence. For this piece of idea the work had been edited by van Caneghem and Warren et al. [260] and the extensive logic-programming in NLP was prepared by Balbin and Lecot [30]. Since the original implementation in Marseille, Prolog implementation techniques, including compilation and various space-saving devices, have progressed to the point that Prolog is today at least comparable with other symbolic-processing languages, such as LISP, for a variety of problems areas, in particular natural-language processing this original piece of work was cited by Warren et al [ ] . Natural language programming (NLP) and PROLOG is an ontology-assisted way of programming in terms of natural language sentences, e.g. English. A structured document with Content, sections and subsections for explanations of sentences forms a NLP document, which is actually a computer program. NLP may also be defined as a form of human-to-computer interaction where the elements of human language, be it spoken or written, are formalized so that a computer can perform value-adding tasks based on that interaction. The last decade has been one of dramatic progress in the field of Natural Language Processing (NLP). This hitherto largely academic discipline has found itself at the center of an information revolution ushered in by the Internet age, as demand for human-computer communication and information access has exploded. Emerging applications in computer-assisted information production and dissemination, automated understanding of news, understanding of spoken language, and processing of
foreign languages have given impetus to research that resulted in a new generation of robust tools, systems, and commercial products. Necessary and sufficient conditions of unacceptable solutions for a number of nonlinear programming discriminated models have been developed by Baichun Xiao [21] and Bill Manaris [36] in his research paper has dealt with Natural Language processing from the perspective of human Computer Interaction. This paper examines the field of natural language processing as it related with human-computer interaction by focusing on its history, interactive application areas, theoretical approaches to linguistic modeling, and relevant computational and philosophical issues.

Some sophisticated approaches involve methods to determine an orientation for equations so that the result of applying an equation is in some appropriate sense simpler than the starting expression, and methods for completing a set of equations so that the order of application of oriented equalities does not matter. These techniques are surveyed by Buchberger et al. [38]. Sub categorization has been a primary phenomenon of interest in modern linguistics, and there are as many analyses of the phenomenon as there are linguistic theories, if not more. Olena Medelyan et al. [187] have provided a comprehensive description of NLP processing framework which focuses on research that extracts and makes use of the concepts, relations, facts and descriptions found in Wikipedia. Jiajun Yan, et al. [120] has integrated multiple classifiers like Naive Bayesian, Decision Tree and Maximum Entropy classifier for Chinese Semantic Dependency analysis. Wiley [266] has encompassed the NLP approaches that use computers to analyze, determine semantic similarity, and translate between languages. V.F. López et al [256] have described the process by which the word alignment task performed within SOMA gent works in collaboration with the statistical machine translation system in order to learn a phrase translation table in NLP. A variety of work in language analysis by Bobrow and Webber [232], Sondheimer et al. [232], Lytinen [219], Hirst [83] and language generation by Jacobs [94], Sondheimer and Nebel [233] exploits some STRUCTURED knowledge representation. A structured representation is one in which entities, or FRAMES, and their slots, or ROLES, may be related to more abstract objects and roles, as in FRAIL [81], or KODIAK [91]. The main advantage of structured representations for natural language is that knowledge about how a given frame or role is expressed linguistically may be taken as default knowledge about the expression of more specific frames and roles.
Introduction and Brief outline of the program

knowledge representation in both processes. Certain knowledge representation foundations, such as structured inheritance networks and feature-based linguistic representations, have proved useful in a variety of language processing tasks. Augmentations to this common framework, however, are required to handle particular issues, such as the ROLE RELATIONSHIP problem: the task of determining how roles, or slots, of a given frame, are filled based on knowledge about other roles. Three knowledge structures are discussed that address this problem. The semantic interpreter of an analyzer called TRUMP (Transportable Understanding Mechanism Package) uses these structures to determine the fillers of roles effectively without requiring excessive specialized information about each frame.

SEMANTIC INTERPRETATION is the part of the language analysis process that consists of constructing a correct, complete representation of the content of a natural language input. The task is difficult because there are no hard and fast rules about the relationship between linguistic structure and underlying meaning.

TRUMP (Transportable Understanding Mechanism Package) [80] is a natural language system designed for use in a variety of domains. The theoretical basis of the system is that the Ace knowledge base design makes it possible for a core of linguistic knowledge to be exploited across domains. The algorithm that TRUMP uses to perform linguistic analysis consists of the following mechanisms: (1) A syntactic parser, which identifies linguistic constituents and instantiates linguistic relations that are tied to matched structures, (2) A mapping mechanism, which produces conceptual structures by following VIEW and REF links from these instantiated linguistic structures, and (3) A phase called concretion, which finds the most specific frame suggested by other information and fills out the roles of that frame. For more details of this process, the reader is referred to Jacobs [190].

Three knowledge-based systems are especially similar in design to TRUMP. These are the KL-ONE based work described by Sondheimer, Weischedel and Bobrow [232], the Absity program of Hirst [83], and Lytinen's MOPTRANS [111]. The KL-ONE based work is closest in knowledge base design to TRUMP, as the systems use similar, uniform representational frameworks for both the linguistic "syntaxonomy" and the conceptual knowledge base. The translation rules of this system, corresponding closely to the REF
associations of Ace, can also be applied through inheritance, thus facilitating the use of
generalized role relationships.

These rules, however, are not expressed in a declarative form; thus it is not as clear
how they would apply to language generation, and it proves difficult to handle
metaphorical expressions. Hirst's system also applies a similar knowledge organization,
but for theoretical reasons Hirst does not allow role relationships to be inherited in his
system, requiring case-role specifications for each word sense. The "Polaroid words" in
Hirst's system correspond closely to the concretion mechanism of TRUMP, in that roles
are filled and more specific frames activated as more information propagates from the
linguistic mechanism of the system. Lytinen's MOPTRANS system is similar to TRUMP
not only in the organization of knowledge structures (MOPs) but also in the choice of
some of the higher level frames that are used to derive specialized interpretations via
frame selection. MOPs, however, do not really support structured inheritance: although
constraints and prototypes may be assigned at the concept_ROLE level, a role or slot of a
particular frame cannot derive from multiple roles. Thus, as in Absity, much of the
knowledge used to handle role relationships must be handled at a very specific level.
Also, this knowledge is attached to the individual MOPs, rather than being declaratively
represented.

The database subset of Prolog is discussed in more detail by Sterling and Shapiro
[237]. The relationship between first-order logic and Prolog is covered to some extent in
that book, and is also addressed in the books by [134], Gallier, Clocksin and Mellish
[47]. The dual interpretation of logic programs, declarative and procedural, was first
discussed by Kowalski [135], and related to denotation semantics for programming
languages by van Emde and Kowalski [134]. Lloyd’s book gives a detailed
mathematical account of the semantics of definite-clause programs. Any Turing machine
can be represented by a pure Prolog program T’arnlund [242], the recursive insolubility
of the Turing-machine halting problem implies that the determination of whether a goal is
provable is in general an undesirable question.

A linguistically oriented overview of context-free grammars can be found in the
book by Partee et al. [196]. A full mathematical treatment of context-free grammars, their
properties, and parsing algorithms is given, for instance, by Harrison [111]. The
representation of context-free grammars in first-order logic has been in the folklore for a
long time, but the first reference to the idea in print that we know of is by Kowalski [134].
Jose Perez-Carballo [126] has investigated the use of NLP techniques in the design phase of information systems. Lipez [153] has aimed at investigating in detail the role of syntactic analysis in NLP and in finding answers to the question why it works better for some queries and worse for others. Elisabeth Métais [70] has investigated the use of NLP techniques in the design phase of information systems. In the lecture notes in Computer Science, published by Springer, Ricardo Baeza-Yates [215] has explored the challenges to effectively use natural language processing for information retrieval. R.W. Smith [212] has explored the technical challenges in constructing natural language interfaces, including a discussion of representative interfaces that illustrate selecting the narrations; verbs from a huge corpus of English database, and future trends in natural language interface research and development. Stephanie Chua [238] has explored the effects of different POS on text categorization effectiveness.

We may illustrate this dichotomy by focusing on one particular performance issue, the problem of disambiguation. As soon as a competence grammar is large enough to cover a non-trivial fragment of a natural language, it assigns to many sentences an extremely large number of alternative syntactic analyses. Human language however, tent to perceive only one or two these. The combinatorial explosion of syntactic analyses (and corresponding semantic interpretations) of natural language sentences has been ignored by linguistic. Almost from its origin, the development of logic programming has been closely tied to the search for computational formalisms for expressing syntactic and semantic analyses of natural-language sentences. One of the main purposes in developing Prolog was to create a language in which phrase structure and semantic-interpretation rules for a natural-language question-answering system could be easily expressed.

This work deals with the aspect of computational linguistics from the point of view of logic programming. The connection between computational linguistics and logic programming has both formal and utilitarian aspects. On the formal side, we shall explore the restricted logical language of definite clauses as a means of expressing linguistic analyses and representations. On the utilitarian side, we shall introduce the logic-programming language Prolog, whose backbone is the definite clause
formalism, as a tool for implementing the basic components of natural language processing systems, for this whole function of Prolog we design some software’s and related work by other researcher are as:

CPIDR (Computerized Propositional Idea Density Rater, pronounced “spider”) is a computer program that determines the propositional idea density (P-density) of an English text automatically on the basis of part-of-speech tags. The key idea is that propositions correspond roughly to verbs, adjectives, adverbs, prepositions, and conjunctions. After tagging the POS using MontyLingua [89], CPIDR applies numerous rules to adjust the count, such as combining auxiliary verbs with the main verb. It is well known that propositional idea density (proposition density, P-density), in the sense of Kintsch [140] and Turner and Greene [254], can be approximated by the number of verbs, adjectives, adverbs, prepositions, and conjunctions divided by the total number of words Snowdon et al. [235]. In this continuation an earlier study developed in the same software by Brown, Snodgrass, et al. [27]. Refined this technique and used a POS tagger, plus readjustment rules, to obtain accurate idea density measures.

NLP strives to help the computer understand the natural language that is being input into it. This is very useful specifically for writers as natural language processing can "read" natural language input and process them. We have made the use of NLP technology in the generation of automatic English grammar rules checking software. These programs can check the process of tense conversion, narration conversion and extraction of key sentences and other rules of English Grammar. In order to develop realistic NLP applications and support advanced research in computational linguistics, large scale grammars are needed. By the end of 90’s, several such grammars had been developed by hand; especially for English and French language. The first source of literature in the field of Natural language processing is found in the book “Different NLP problems and the application of statistical techniques” written by Klein, Dan et. al. [133]. Other books in this field are the book of R. Daleet al. [152] and Jurafsky Daniel, et al. [53].

A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'.
Almost from its origin, the development of logic programming has been closely tied to the search for computational formalisms for expressing syntactic and semantic analyses of natural-language sentences. One of the main purposes in developing Prolog
was to create a language in which phrase structure and semantic-interpretation rules for a natural-language question-answering system could be easily expressed.

CATPAC (http://www.terraresearch.com/):
CATPAC reads text files and produces a variety of outputs ranging from simple diagnostics (e.g., word and alphabetical frequencies) to a summary of the "main ideas" in a text. It uncovers patterns of word usage and produces such outputs as simple word counts, cluster analysis (with icicle plots), and interactive neural cluster analysis.

Computer Programs for Text Analysis:
This is not a single computer program but rather a series of separate programs by Eric Johnson that each perform one or two basic functions, including analyzing appearances of characters in a play (ACTORS program), getting KWIC (CONCORD program), computing the amount of quotation in texts (DIALOG program), and comparing the vocabulary of two texts (IDENT program). The programs seem ideal for literary-type analyses.

Concordance 2.0 (http://www.concordancesoftware.co.uk/):
Concordance is a flexible text analysis program which lets you gain better insight into electronic texts and analyze language in depth. You can make concordances, word lists, and indexes from electronic text. Count word frequencies, find phrases, lemmatise, see word collocations, and more. User-definable alphabet, contexts, and references.

Diction 5.0 (http://www.dictionsoftware.com):
Diction 5.0 contains a series of built-in dictionaries that search text documents for 5 main semantic features (Activity, Optimism, Certainty, Realism and Commonality) and 35 sub-features (including tenacity, blame, ambivalence, motion, and communication). After the user's text is analyzed, Diction compares the results for each of the 40 dictionary categories to a "normal range of scores" determined by running more than 20,000 texts through the program.

DIMAP (http://www.clres.com/):
DIMAP stands for Dictionary Maintenance Programs, and its primary purpose is dictionary development. The program includes a variety of tools for lexicon building rooted in computational linguistics and natural language processing Litkowski [151].
The system through which any user can measure the translation power of a computer grammar with respect to writing speed is called a Computer assisted writing system; it will be a more flexible system for defining translations and uses NLP as a domain, or we can say that Computer assisted writing system (CAWS) refers to instruction or remediation presented on a computer. Miyao and Tsujii, [263] have discussed maximum-entropy models based on Computer assisted writing systems by estimating them using a computer based algorithm approach. A note on the strong and weak generative powers of Computer assisted writing system has been given by Joshi, Aravind K. [127]. The research paper of A. Ramsay [1], proposes that a machine that could understand natural language has been one of the dreams of Artificial Intelligence ever since computers were invented. In the paper entitled “Computer assisted writing system Improving Readability with respect to Information system”, Nobo Komagata [185] has developed a Computer-Assisted Writing system focusing on information structure. In order to perform the integrated approach to check the cross lingual tasks of errors, the software company IBM has proposed Computer assisted writing system by using the statistical IBM 1 word alignment model (M1) as is evident from the research paper entitled “A statistical approach to cross lingual natural language tasks” appeared in Journal of Algorithms, published by David Pinto, Jorge Civera, et al. [167] and Chien-Liang Liu et al. [68] have designed and implemented a computer assisted writing system by using NLP as the application domain.

Selection of texts from data base is usually accomplished, using statistical methods (often coupled with manual encoding), but it is now widely believed that this traditional method has reached its limit. These limits are particularly acute for text databases, where natural language processing (NLP) has long been considered necessary. This problem has been addressed by researchers, prominent among them being Allan J. Raghavan [7] for using part-of-speech patterns to reduce query ambiguity; Jijkoun et, al., [121] for establishing the superiority of the method of extracting syntactic relations between entities in a text against the method of using surface–based patterns. Syntactic parsing of the database contents has been attempted in order to extract linguistically motivated "syntactic phrases", which presumably were better indicators of contents than "statistical phrases" where words were grouped solely on the basis of
physical proximity and knowledge bases for each new text domain. A novel use of statistical parsing to extract information from text can be seen in the work of Miller et al. [181]. The complexity of some extensions of Range connection Grammar RCG parsing has been addressed by Bertsch, Eberhard et al. [34]. Klein, Dan [155] has worked for un-lexicalized parsing accurate. In another approach of parsing Bikel, Daniel M. [36] has developed the parsing model to extract the sentences from the corpus, for this purpose different researcher work on this filed and developed many programs as follows:-

General Inquirer http://www.wjh.harvard.edu/~inquirer/

This program, created by Phillip J. Stone, "now provides English-language content analysis capabilities using both the "Harvard" and "Lasswell" general-purpose dictionaries as well as any dictionary categories developed by the user.

General Inquirer (Internet version) (http://www.webuse.umd.edu:9090/)

The General Inquirer has found new life on the World Wide Web. The online version of the General Inquirer gets our vote for the simplest and quickest way to do a computer text analysis—simply visit the Internet General Inquirer site, type or paste some text into a box, click submit, and your text will be analyzed.

HAMLET (http://www.apb.cwc.net/homepage.htm)

"The main idea of HAMLET is to search a text file for words in a given vocabulary list, and to count joint frequencies within any specified context unit, or as collocations within a given span of words. Individual word frequencies (fi) , joint frequencies (fij) for pairs of words (i,j), both expressed in terms of the chosen unit of context, and the corresponding standardized joint frequencies are displayed in a similarities matrix, which can be submitted to a simple cluster analysis and multi-dimensional scaling.

Lexa (http://nora.hd.uib.no/lexainf.html)

Designed with linguists in mind, Lexa Corpus Processing Software is a suite of programs for tagging, lemmatization, type/token frequency counts, and several other computer text analysis functions.

SALT(SystematicAnalysisofLanguageTranscripts) http://www.languageanalysislab.com

This program is designed mainly to help clinicians identify and document specific
language problems. It executes a myriad of analyses, including types of utterances (e.g., incomplete, unintelligible, nonverbal), mean length of utterances, number and length of pauses and rate of speaking, and frequencies for sets of word (e.g., negatives, conjunctions, and custom dictionaries).

Scheduling and management of ADA (Advanced Designing and Algorithm) resources is an area of ongoing research and development. Several open source or proprietary schedulers have been developed for clusters of servers, including Maui [54], Secondary Data Algorithm (SDA) [54], they typically run in batch mode, can be customized to special environment, and attempt to balance the load among the various servers. However, the primary objective of most existing approaches is to improve overall system performance (e.g., utilization), while the ADA experienced by any nave users is at best of secondary Algorithm [57]. We always reanalyzed the concept of ADA by tagging the word/narratives in a sequence of above format and then recurs the function F1……FN and count the narratives in a text either by ascending or descending order.

In this Thesis we emerge a ADA and Recursive function together, because recursive function can collaborate ADA again and again till the finite result, in the same field we cite the pioneering work of Gold et al. [68], Solomon off [204], Thiele et al. [101], Blum and Blum et al. [37], and the work done in Riga, inductive inference of recursive functions has fascinated many researchers. By definition, inductive inference is the process of generating hypotheses for describing an unknown object from finitely many data points about the unknown object. The mathematical basis for the work presented in this survey goes back to Solomon off [204] who proposed criteria for selecting a hypothesis explaining given data best, Putnam [102] who anticipated several of the earlier results (though on an informal basis) and Gold et al. [68] who has provided a thorough recursion theoretic basis of inductive inference. One important open problem in Natural language processing (NLP) is of Word sense disambiguation (WSD). It is the process of identifying correct sense of word in a given sentence and has played the major role in this work because to extract the data from database we are using natural English language so in order to identify the meaning of the sentence, ambiguous words are to be disambiguated.
Chapter 1

engines, anaphora resolution, coherence, inference, information retrieval, machine translation and others.

The first difficulty in analyzing Chinese is segmentation and definition of a word. The most basic unit of the Chinese language is a morpheme 23. A morpheme is one Chinese character. On a computer, Chinese morpheme can be expressed in GB and BIG5 encoding, where each character is made up of 8 bytes. In Unicode, this character may be encoded in either 16 or 32 bytes depending on which version of Unicode.

TACT (Text Analysis Computing Tools) is a text-analysis and retrieval system for MS-DOS that permits inquiries on text databases in European languages. It has been developed by a team of programmers, designers, and textual scholars. It was begun under the IBM-University of Toronto Cooperative in the Humanities during 1986-89.

In the paper entitled “Unsupervised word sense disambiguation for Korean through the acyclic weighted digraph using corpus and dictionary”, Yeohoon Yoon, Choong-Nyoung Seon, Songwook Lee, Jungyun Seo [270], have defined that word sense disambiguation (WSD) is the process of assigning the most appropriate sense to a polysemous word according to its context. They have presented a method for automatic WSD using only two resources: a raw text corpus and a machine-readable dictionary (MRD). Yukiko Sasaki Alam [271] has presented a new model of lexical-semantic representation of the lexicon intended for use in word sense disambiguation and semantic representation of a text.

Part-of-speech (POS) categories are elementary building blocks for the syntactic analysis of text that play an important role in many natural-language-processing tasks, from machine translation to information extraction. While English and a handful of other languages are fortunate enough to have comprehensive POS-annotated corpora such as the Penn Treebank, Marcus, Marcinkiewicz, & Santorini [105], most of the worlds’ languages have extremely limited linguistic resources. Text categorization (a.k.a. text classification) is the task of assigning predefined categories to free-text documents. It can provide conceptual views of document collections and has important applications in the real world. Xia, Fei et al. [267], have developed guidelines for ensuring consistency in Chinese text categorization. Tao Li, Shenghuo Zhu, Mitsunori Ogharah [244] have discussed text categorization via generalized discriminated analysis and have presented a
Introduction and Brief outline of the program

simple and efficient solution to multi-class text categorization form the database. Lifei Chen, Gongde Guo, Kaijun Wang [151] have introduced a class-dependent projection method to text categorization and have projected the document categories into their special reduced subspaces so as to make different classes easily separable.

Today parametric associative CAD systems must help companies to create more efficient virtual development processes. While dealing with complex parts (e.g. the number of surfaces of the solid) no CAD modeling methodology is existing. Based on the analysis of industrial designers' practices as well as student practices on CAD, we identified key factors that lead to better performance. Our objective is to propose a practical method for complex parts modeling in parametric CAD system, for this related work one researcher Yannick Bodein et al [268]. Apply this rule in CAD canalization. For this CAD canalization there are some basic tools used in this thesis, and also followed by the main modeling activities are carried out in a common workspace in the server side. A thin/strong representation in client/server, respectively, has been proposed to enhance the performance of the system effectively in this performance the researcher Li Wd et al. [56] developed the performance based system, the developed systems include Alibre Designe, OneSpacee, van den Berg et al. [8], Li et al. [57], Bidarra et al. [12], etc. The third architecture, including Begole et al. [24] and Inventor collaborative tool, supports the sharing and manipulation of services or modules of a system by other systems. Here is some developed software for retrieving a knowledge based system to reduce network ambiguity:-

Fibersim is a suite of software that supports all of the unique and complex design and manufacturing methodologies necessary for you to engineer innovative, durable and lightweight products and parts made of advanced composite materials.

NX is a suite of integrated, fully associative CAD/CAM/CAE applications. NX touches the full range of development processes in product design, manufacturing and simulation, allowing companies to encourage the use best practices by capturing and reusing product and process knowledge.
Seat Design Environment (SDE) is software that's fully integrated into commercial 3D CAD systems, for designing and manufacturing innovative transportation seat systems and interior components.

Solid Edge is a hybrid 2D/3D CAD system that uses synchronous technology for accelerated design, faster change, and improved imported re-use. With part and assembly modeling, drafting, transparent data management, and built-in finite element analysis (FEA), Solid Edge eases the growing complexity of product design.

Syncrofit is a family of specialized engineering products for designing and manufacturing complex assemblies and large aero structures. It allows you to author and manage the assembly interfaces and hundreds of thousands of fasteners that are typical in an airframe.

The following software components are used by CAD software developers as the foundation for their applications:

Parasolid is 3D geometric modeling component software, enabling users of Parasolid-based products to model complex parts and assemblies. It is used as the geometry engine in hundreds of different CAD, CAM and CAE applications.

D-Cubed Components are six software libraries that can be licensed by software developers for integration into their products. The capabilities they provide include parametric sketching, part and assembly design, motion simulation, collision detection, clearance measurement and hidden line visualization.

The design task can often be seen as an optimization problem in which the parameters or the structure describing the best quality design are sought. Genetic algorithms constitute a class of search algorithms especially suited to solving complex optimization problems. In addition to parameter optimization, genetic algorithms are also suggested for solving problems in creative design, such as combining components in a novel, creative way. Genetic algorithms transpose the notions of evolution in Nature to computers and imitate natural evolution. Liu [50] proposed a COM interface-based framework to wrap and expose API functions of CAD kernels/systems and process planning modules for remote invocations.
The concept of developing standard interface specifications, namely the common core interfaces, was proposed to encapsulate specific feature functions of different CAD kernels/systems to provide a generic and neutral application layer according to some international standards for features. The advantages of these two works include the straightforwardness of calling wrapped feature functions and the neutrality of CAD kernels/systems for different applications. However, considering the complexity and variation of features, the programming effort for implementation is quite huge and the add-on wrapping structures make the system quite heavy. In Jacquel and Salmon’s system [117], features from an ACIS modelling kernel are wrapped as services for remote design and manufacturing analysis. Gerhard et al. [86] proposed an event-based and agential framework to communicate design and manufacturing information through agent channels based on the Java RMI technology, and manufacturing analysis functions are enveloped as agents to support the establishment of an open and plug-in environment.

In principle, information storage and retrieval is simple. Suppose there is a store of documents and a person (use of the store) formulates a question (requestor query) to which the answer is a set of document satisfying the information need expressed by his question. He can obtain the set by reading all the document in the store, retaining the relevant document and discarding all the others. In a sense, this constitutes ‘perfect’ retrieval. This solution is obviously impracticable. A user either does not have the time or does not wish to spend the time reading the entire document collection, a part from the fact that it may be physically impossible for him to do so. The present proposed work deals with the aspect of computational linguistics and Quantitative Linguistics, from the point of view of logic programming. The connection between computational linguistics and Quantitative Linguistics via logic programming has both formal and utilitarian aspects. Quantitative Linguistics is a sub-discipline of general linguistics and, more specifically, of mathematical linguistics, which deals with language learning, language change, and application as well as structure of natural languages. For various methods and models used in quantitative linguistics, we can cite the book by Kohlar R. (Ed.) [136], Kohlar and Rieger (Eds.) [138] and Mohanty P. and Kohlar R. (Eds.) [182]. The book of Rasinger S. M. [213] presents a comprehensive introduction to the analysis and
examination of the fact that how quantitative techniques can be used by linguists to arrive at meaningful and accurate conclusion. Research work by Marusenko Mikhail et al. [178], focuses on developing a linguistic approach in the field of attribution of literary works using the material of plays written in verse.

A Knowledge-based system (KBS) is a computer program that reasons and uses a knowledge base to solve complex problems. The term is broad and is used to refer to many different kinds of systems. The one common theme that unites all knowledge based systems is an attempt to represent knowledge explicitly via tools such as ontologism and rules rather than implicitly via code the way a conventional computer program does.

Documents on the Internet are composed of several kinds of multimedia information when accessed for personal, entertainment, business, and scientific purposes. There are many specific content domains of interest to different communities of users. Extracting semantic relationships between entities from text documents is a challenging task in information extraction. By semantics for natural language in this connection, this paper understands not just the relating of a semantic representation language to natural language but the evaluation of natural language expressions with respect to databases.

Evaluating a declarative sentence (on a given reading) with respect to a database involves determining whether the sentence is true with respect to the data base, whether the sentence appropriately describes the database. Evaluating a question with respect to a database might determine what information in the database would lead to appropriate answers to the question, for this work in this thesis we developed a SVM technology which support maximum margin of errors in Networks.

Support vector machines are a set of algorithms that learn from data by creating models that maximize their margin of error. Support (SVMs) are a family of algorithms for classification, regression, transduction, novelty detection, and semi-supervised learning. In this thesis we choose SVM machine learning, optimization methods, and the intersection of machine learning with natural language processing and semantic modeling that maximizes the error margin of a training set. Our work covers a broad spectrum of research activities, from fundamental, theoretical machine learning to a wide variety of applied research projects, such as social network-based question answering.
Introduction and Brief outline of the program

The purpose of this thesis is to provide an introductory yet extensive tutorial on the basic ideas behind Support Vector Machines (SVMs). The books Vapnik [257], contain excellent descriptions of SVMs, but they leave room for an account whose purpose from the start is to teach.

By way of motivation, and to alert the reader to some of the literature, we summarize some recent applications and extensions of support vector machines. For the pattern recognition case, SVMs have been used for isolated handwritten digit recognition Cortes and Vapnik et al [22]; object recognition Blanz et al. [13], speaker identification chmidt [159], and text categorization Joachims [243]. For the regression estimation case, SVMs have been compared on benchmark time series prediction tests Muller et al [187]; Mukherjee et al. [222], the Boston housing problem Drucker et al. [38], In most of these cases, SVM generalization performance (i.e. error rates on test sets) either matches or is significantly better than that of competing methods. The use of SVMs for density estimation Weston et al., [85], and ANOVA decomposition Stitson et al., [158], has also been studied. Regarding extensions, the basic SVMs contain no prior knowledge of the problem (for example, a large class of SVMs for the image recognition problem would give the same results if the pixels were first permuted randomly (with each image suffering the same permutation), an act of vandalism that would leave the best performing neural networks severely handicapped) and much work has been done on incorporating prior knowledge into SVMs Scholkopf, Burges and Vapnik, 1996; Scholkopf et al., 1998a; Burges, 1998 [85]. The reduced set method Burges, 1996; Burges and Scholkopf, 1997 [99], was introduced to address the speed of support vector machines in test phase, and also starts with a trained SVM.

SVMs provide a new approach to the problem of pattern recognition (together with regression estimation and linear operator inversion) with clear connections to the
underlying statistical learning theory. They differ radically from comparable

approaches such as neural networks: SVM training always finds a global minimum and their simple geometric interpretation provides fertile ground for further investigation. An SVM is largely characterized by the choice of its kernel, and SVMs thus link the problems they are designed for with a large body of existing work on kernel based methods and for this purpose there are different software’s developed cited as :-

Estelar
SVM (struct), again by Joachism, ia an implementation that can model complex (multivariate) output data \( y \), such as trees, sequences, or sets. These complex output SVM can be applied to NLP parsing, sequence alignment in pattern recognition.

SVM (light), by Joachims, is one of the most widely used SVM classification and regression package, it has a fast optimization algorithm, can be applied to very large datasets, and has a very efficient implementation of the leave-one-out cross-validation.

SVM (pml) is an interactive object oriented framework for machine learning in Python. It contains a wrapper for LIBSVM, and procedures for optimizing a classifier: multi-class methods, descriptor selection, model selection, jury of classifiers, cross-validation, ROC curves.

LS-SVM (lab), by Suykens, is a MATLAB implementation of least squares support vector machines (LS-SVM) which reformulates the standard SVM leading to solving linear KKT systems. LS-SVM alike primal-dual formulations have been given to kernel PCA, kernel CCA and kernel PLS, thereby extending the class of primal-dual kernel machines.

In this Thesis, we focus on loop perforation (Program Transformation), which transforms loops to execute only a subset of their original iterations. Empirical results demonstrate the utility and effectiveness of loop perforation in reducing the amount of time (and/or other resources such as energy) that the application requires to produce a result while preserving acceptable accuracy, this work is cited also by some researcher H. Homann et al. [110]. While investigating the reasons behind these empirical results, we identified specific computations in our benchmark applications that interacted well with loop perforation. Inspired by these computations, in this paper we present four generalized computational patterns that interact well with loop perforation. We have previously proposed the use of Monte-Carlo simulation to explore how loop perforation changes the result that specific computational patterns produced by M. Rinard et al. [39].

In general, we do not understand the kind how to changes programmers make to software. The usual categories such as “bug fix” and “new feature” are too broad. In the past few years, software engineering researchers have been studying specific classes of bugs, such as null pointer violations and copy-and-paste errors, and have been using this
as a basis of tools for detecting and fixing bugs. This is important work that fits the theme of this paper well. However, not as much work has been done on adding features. Don Batory's [78] work on features is one good example of the kind of work that should be done. We need to better understand what we mean by "fix a bug" and "add a feature", which means to learn how to categorize them into more precise changes, learn when each kind of change is most appropriate, learn how to detect them when programmers make them, and (finally) learn when and how to automate them. Programs are often viewed as a set of modules. An alternate view is to think of a program as a sequence of program transformations. Composing sequences to form large sequences becomes a new kind of program composition, one that is well suited for feature-oriented programming designed by Dig, D [65]. Many program transformations are larger than a single program.

The CIP Project developed software via program transformation. CAD, Intuition-guided Programming, at the Technical University of Munich contributed much to the field of transformation [97]. The CIP system uses the approach of allowing the programmer to guide the transformer system [110]. The knowledge base is small but the fully interactive nature of the system allows a user to build new transformation upon the original transformation rules [95]. The CIP system represents programs as algebraic specifications [112].

For example, qmail is a mail transport agent that was designed to be more secure than send mail, its chief competitor, and has had a perfect security record since it was released in 1997. One of the reasons that it is so secure is that it is composed of over twenty small C programs, while send mail is one large C program Bravenboer, M et al. [158]. As on another part of Danny Dig’s dissertation and developed many using refactoring tools to ease the cost of component evolution, he developed a version control system that represented program history as a sequence of refactoring and program edits. Hafiz, et al.[82] This turned out to make merging much easier. It is probably possible to integrate other kind of program transformations with version control. This could make it easier to understand the history of the system. Making merging easier means that it is easier to apply a change done to one branch to another branch. Changes become components that can be moved from one branch of the version history to another. Programs are often viewed as a set of modules. An alternate view is to think of a
program as a sequence of program transformations. Composing sequences to form large sequences becomes a new kind of program composition, one that is well suited for feature-oriented programming developed by Delaware, B et al. [62].

In the present thesis an attempt is also being made to review the area of Information Retrieval (IR) from a computational semanticist's point of view. The work in ANGLABHARTI developed by R. Sinha, K. Sivaraman, A. Agrawal, R. Jain, R. Srivastava, A. Jain [3] represents a machine-aided translation methodology specifically designed for translating English to Indian languages. An excellent discussion of the origins of information retrieval has been given by Sandor Dominich, Mounia Lalma and C. J. Kiith van Rijsbergen [237]. Paul Thompson [200] has described the Maron and Kuhns article and the influence that it has had on the field of information retrieval.

An IR system can react in a variety of ways. For instance, it can present a list of documents presumably containing the information the user is looking for (ad-hoc retrieval), or it can directly answer a user’s question by generating natural language sentences or extracting sentences from the database (Question-Answering). The database itself is simply a collection of natural language documents. From an abstract point of view, a document \( d \) is relevant to a query \( q \) if \( d \) ‘is about’ \( q \). Defining ‘is about’ is a non-trivial task and several approaches have been proposed. An overview of this work can be found in the works of Baeza-Yates, R. and Ribeiro-Neto, B [17]. The work of L. Gore and N. Patil [161] demonstrates the use of Natural Language Processing in IR while D. Jurafsky and J. H. Martin [64] have recognized the content of a document in order to convert it into another language.

We can also cite the works of Jackendoff, R. [126] for using X-bar syntax for defining program transformation (phrase structure) structure; Gazdar G. et al [95] for developing some rules for defining phrase structure. In order to overcome this bottleneck, Sumita Eiichiro [263] proposes a new mechanism for lexical transfer, which is simple and suitable for learning from bilingual corpora. Culpeper J., [60] has worked in the stylistics of drama in learning to disambiguate words to be translated and on other hand Stockwell, P., [258] had considered more recent work in cognitive science on one hand. The article of Li Hang et al [170] proposes a new method for word translation disambiguation, one that uses a machine-learning technique called bilingual bootstrapping.
Abelson & Sussman [4] have developed computer program to determine part of speech and phrase structures. Gymnich Marion et al [98] for developing models for language description and for discussing their applications; Jose B., et al [136] describes in detail an n-gram approach to SMT (Statistical Machine Translation); Paolillo John C [198] for introducing the quantitative study of language variation and change; Lansey Jonathan C., et al [144] for studying the number of internet search results returned form multi-word queries based on the number of results returned. Where natural language processing (NLP) has long been considered necessary for further progress. Further this technique was also used by Farahat, A., et al. [77], and yielded a 10% improvement for new event detection of Machine Translation. Carroll and Fang, 2004 have demonstrated that for a given domain, using an HPSG (Head Driven Phrase Structure Grammar) enriched with detailed sub categorization information improves the parse success rate.

The automatic processing of written texts is being tackled by a variety of scientific disciplines. Within Computer Science the area of Natural Language Processing is deeply concerned with the problem of developing software systems that include language analysis functionalities to solve real problems and through this approach. This work can cite the paper of Martneernnde et al [90] in this arena. Natural language processing (NLP) techniques substantially enhance most phases of the information system lifecycle, starting with requirements analysis, specification and validation, and going up to conflict resolution, result processing and presentation. The work of Elisabeth Métais [70] can be cited for application of NLP in Enhancing information systems management. Program transformations can be classified according to various criteria such as amount of automation; improvement achieved, and subject language as is evident from the works of Feather [78], Partsch [108], Smaragdakis et al. [224].

query likelihood and a monolingual translation based model for the task of identifying restatements. John F. Vinsonhaler et al [135] have described a mixture model of query likelihood and a translation based model for successfully identifying similar word extraction techniques. Hongxia Xu, Li Zhang et al [113] extended traditional Earley’s algorithm for a two-dimensional relation structures. More sophisticated algorithms for top down parsing have been created by Frost R., et al [85].

The Retrieval of Sentences from Database allows users to retrieve sets of sentences fitting specified characteristics. The paper of Masao Fuketa, Jun-ichi Aoe [184] presents a technique for the storing of multi-stages of postings and retrieving them partly in order to compute efficiently the intersection between postings for the requested terms. From the simulation results it has been shown by them that the presented algorithm is 6 to 88 times faster than the traditional approach. Samuel W.K. Chan [235] has provided a novel quantitative model based on a shallow linguistic extraction technique for the creation of a summary by extracting a set of sentences that represent the most salient content of a text and the model.

According to the Mikhailov, A. I., A. I. Chernyi [192], Information retrieval is accomplished by means of an information retrieval system and is performed manually or with the use of mechanization or automation of database. The research paper of Douglas W. Oard, Daqing He, Jianqiang Wang [172] deals in Interactive Cross-Language Information Retrieval (CLIR), a process in which searcher and system collaborate to find documents that satisfy an information need regardless of the language in which those documents are written and calls for designs in which synergies between searcher and system can be leveraged so that the strengths of one can cover weaknesses of the other.

Human–computer information retrieval (HCIR) is the study of information retrieval techniques that bring human intelligence into the search process. This system is typically evaluated based on their mean average precision over a set of benchmark queries from organizations like the text Retrieval Conference (TREC). Marchionini, G. [164] works towards in HCIR available at http://www.asis.org/Bulletin/Jun-06/marchionini.html. Vaughn, M. W., Degen, H., Resnick, M. and Grebett, P [277] have defined that HCIR combines research from the fields of human-computer interaction (HCI) and IR. Ben Shneiderman et al. [28] have given their thoughts for Interactive systems for age and cultural differences which can contribute to acceptance or rejection of interface techniques in Interactive systems.
Bolstaed et al [34] have emphasized the effectiveness of the filtering of relevant documents applied to a Geographic Information Retrieval (GIR) system, instead of query expansion of information retrieval systems. Sandor Dominich et al [225] have worked for issues on model design, formulation and explanation in information retrieval using mathematics. Jonathan Ling, Paul van Schaik [137] in a paper published in International Journal of Human-Computer Studies, has discussed the influence of font type and line length on visual search and information retrieval in web pages. The research paper of Martijn Schuemie, Jan van den Berg [182] entitled “Associative Conceptual Space-based Information Retrieval Systems” published in 'Information Era' deals with the availability of large collections of books, articles, and journals.

Computer Program for IR is a sequence of instructions that a computer can interpret and execute. This term can refer to either the executable form that a computer can execute (executable code) or the human readable form (source code). A System of General Purpose Computer Programs for Information Retrieval in the Behavioral Sciences has been proposed by John F. Vinsonhaler [234]. The Google Search Appliance (GSA) makes the sea of lost and misplaced data on web servers, file servers, content management systems, relational databases and business applications instantly available from a single familiar search box.

Computational Design for Retrieval of Sentences has become crucial for several information retrieval tasks owing to the huge amount of information available in digital media. It has increased the demand for simple, language-independent extractive summarization strategies. Lucas Antiqueira, Osvaldo N. Oliveira Jr., Luciano da Fontoura Costa, Maria das Graças Volpe Nunes [175] have developed a paper by employing the concepts and metrics of complex networks to select sentences for an extractive summary.

As PROLOG is one of the most popular languages based on an inference mechanism and many expert systems are implemented in PROLOG. In the present work, we have selected this language to express the rules and a PROLOG interpreter and its uses as an inference mechanism, taking the initiative from the work of A Domenici et al. [1]. We have also utilized the work of Pascual Julián-Iranzo et al. [193], who have examined the extension of Prolog in order to be able to deal with similarity-based fuzzy unification and the work of R.J. Lucas [18], who has kept the options open to
implementers of Prolog interfaces to relational databases. Prolog programs are
nondeterministic, since several elements can be in a particular relation to a given element.
Because conventional languages are geared toward functions, that is, relations in which
one element is uniquely defined in terms of the others, computation proceeds
deterministically in such languages.

It is shown that multiple paradigms (PROLOG) can be incorporated without
disturbing logic programming language features and efficiency. It also introduces a new
programming paradigm called the relation-oriented paradigm. The research results are
reflected in the implementation of the Prolog-based knowledge programming system
PEACE (Prolog based Engineering Applications Environment), which is used to realize
an expert system in a diagnostic domain. PEACE provides a relation-oriented
programming paradigm, as well as previously discussed paradigms, such as object-
oriented, data-oriented, and rule-oriented paradigms. These paradigms are nicely
amalgamated in Prolog language and can be used intermixed.

Multiple knowledge programming paradigms are amalgamated in Prolog by the
technique called meta-programming Bowen [139]. Additionally, it describes a new
implementation of the Prolog based knowledge programming system called PEACE. It is
shown that PEACE efficiently supports a semantic network knowledge representation
realizing relation oriented programming as well as object-oriented programming, data-
oriented programming and rule-oriented programming amalgamated in Prolog. Another
way is to express a semantic network in a set of Horn clauses and to use the Prolog
interpreter as an inference mechanism Koyama [100].

When the characterization of a document is worked out, it should be such that
when the document it represents is relevant to a query, it will enable the document to be
retrieved in response to that query. Human indexers have traditionally characterized
document in this way when assigning index terms to document. The indexer attempts to
anticipate the kind of index terms a user would employ to retrieve each document whose
content he is about to describe. Implicitly he is constructing queries for which the
document is relevant. When the indexing is done automatically it is assumed that by
pushing the text of a document or query through the same automatic analysis, the output
will be a representation of the content, and if the document is relevant to the query, a
computational procedure will show this. Intellectually it is possible for a human to
establish the relevance of a document to a query. For a computer to do this we need to
construct a model within which relevance decisions can be quantified.
As logic programming applications grow in size, Prolog systems need to efficiently access larger and larger data sets and the need for any- and multi-argument indexing becomes more and more profound. The WAM [167] has mostly been a blessing but occasionally also a curse for Prolog systems.

Many Prolog systems still only support indexing on the main function symbol of the first argument. Some others, such as YAP version 4, can look inside some compound terms [29]. SICStus Prolog supports shallow backtracking [134]; choice points are fully populated only when it is certain that execution will enter the clause body. While shallow backtracking avoids some of the performance problems of unnecessary choice point creation, it does not offer the full benefits that indexing can provide. Other systems such as BIM-Prolog [74], SWI-Prolog [45] and XSB [169] allow for user-controlled multi-argument indexing. Notably, ilProlog [178] uses compile-time heuristics and generates code for multi-argument indexing automatically.

Recognizing the need for better indexing, researchers have proposed more flexible indexing mechanisms for Prolog. For example, Hickey and Mudambi proposed switching trees [88], which rely on the presence of mode information. Similar proposals were put forward by Van Roy, Demoen and Willems who investigated indexing on several arguments in the form of a selection tree [260] and by Zhou et al. [272] who implemented a matching tree oriented abstract machine for Prolog [110]. For static predicates, the XSB compiler offers support for unification factoring [112]; for asserted code, XSB can represent databases of facts using tries [127] which provide left-to-right multi-argument indexing. However, in XSB none of these mechanisms is used automatically; instead the user has to specify appropriate directives. Long ago, Kliger and Shapiro argued that such tree-based indexing schemes are not cost effective for the compilation of Prolog programs [133].

In a programming language such as Mercury [231] where modes are known the compiler can of course avoid this risk; indeed in Mercury modes (and types) are used to guide the compiler generate good indexing tables. However, the situation is different for a language like Prolog. Getting accurate information about the set of all possible modes of predicates requires a global static analyzer in the compiler — and most Prolog systems do not come with one. More importantly, it requires a lot of discipline from the programmer (e.g., those applications use the module system religiously and never bypass
As a result, most Prolog systems currently do not provide the type of indexing that applications require. Even in systems such as Ciao [109], which do come with a built-in static analyzer and more or less force such a discipline on the programmer, mode information is not used for multi-argument indexing.

A Knowledge Based Systems is on whose aim is merely to provide advice and guidance to the user and not autonomously to provide reasoned solutions is essentially a Knowledge Based System. Research on this topic is being done in numerous organizations all over the world, from higher education laboratories to research institutes and software development organizations. The decades of 80’s and 90’s saw great use of Knowledge Based Systems, but in last decade the use of Knowledge Based system has been less or that it is embedded with other technologies. From a strictly technical perspective, a KBS is: “a program for extending and/or querying a knowledge base. A knowledge base is a collection of knowledge expressed using some formal knowledge representation language. A knowledge base forms part of a knowledge-based system (KBS)” cited by Foldoc et. al. [109].

A definition which includes both finality and functionality is given by the Elsevier Knowledge-Based Systems journal. “Knowledge-Based Systems (the journal) focuses on systems that use knowledge-based techniques to support human decision-making, learning and action. Such systems are capable of cooperating with human users and so the quality of support given and the manner of its presentation are important issues.” [23]. Building an expert system is known as Knowledge Engineering and its practitioners are called Knowledge Engineers. The knowledge engineer must make sure that the computer has all the knowledge needed to solve a problem. The knowledge engineer must choose one or more forms of knowledge representation. He must also ensure that the computer can use the knowledge efficiently by selecting from a handful of reasoning methods.
Introduction and Brief outline of the program

In order to know of the previous research done in this direction, we have examined several studies dedicated to the topic or to broader topics including references to KBSs. Most of studies dealt with broader topics with references to KBSs. But just few of these were focused on KBSs in law, hence the reason for reviewing these studies. A literature survey was carried out to know about the Knowledge Based Systems in law over last two decades. It showed that the Expert Systems developed can be categorized into following types. In the Reading Guide on Knowledge-based Legal Applications by Russell Allen and Graham Greenleaf highlights same [53].

Andrew Stranieri, et. al [4] Opine that the evaluation strategies to assess the effectiveness of legal knowledge based systems enable strengths and limitations of systems to be accurately articulated. This facilitates efforts in the research community to develop systems and also promotes the adoption of research prototypes in the commercial world. However, evaluation strategies for systems that operate in a domain as complex as law are difficult to specify. Specific criteria for the evaluation of explanation facilities are also described. Bench-Capon, et.al. [20] Discusses the potential for providing knowledge based support for the task of formulating policy, and determining what legislation is required to implement the policy. Some examples of an alternative approach, based on hypertext, are discussed, and authors also come out with some proposals for overcoming the obstacles with a combination of the hypertext and knowledge based approaches.

Groendijk, C. [87] Opine that in most contemporary legal knowledge based systems, conclusions are reached by applying rules to case descriptions. A case description usually consists of a limited set of facts. In human judicial problem solving, the application of legal rules is not based on the facts directly, but on a structured interpretation of these raw data. A structured data interpretation serves as a guide through the problem space it enables the problem solver to ask context sensitive questions and to make plausible default assignments. A neural method to create structured data interpretations is advocated and a method to integrate these networks with a rule based system is presented.

Hage, J.C., et.al. [103]. The adoption by AI researchers specializing in law of new AI techniques, such as case based reasoning, neural networks, fuzzy logic, denotes logics and non-monotonic logics, may move closer to achieving an automation of legal reasoning. Unfortunately these approaches also suffer several drawbacks that will need to be overcome if this is to be achieved.
Chapter 1

Michael Aikenhead et al.[176] Opine that Computers have long been utilized in the legal environment. The main use of computers however, has merely been to automate office tasks. More exciting is the prospect of using artificial intelligence (AI) technology to create computers that can emulate the substantive legal jobs performed by lawyers to create computers that can autonomously reason with the law to determine legal solutions. Such attempts have not been successful.

Modeling the law and emulating the processes of legal reasoning have proved to be more complex and subtle than originally envisaged. Moles, R.N. [170] compare two different approaches to the development of expert systems in law: the 'law is rules' approach and the 'semiotic view'. The former is exemplified by some of the logic programmers (Kowalski, Sergot, Bench-Capon), the latter by a norm-based, information-systems methodology (Stamper). In looking at them from the point of view of a legal theorist, the authors are concerned more with their view of law than with the computational aspects of their work. P.Hassett, [192] Opine that legal expert systems use a knowledge base of legal rules to address legal issues. Such systems are said to be undesirable (a) because of the difficulty of replicating a multi-textured process such as legal reasoning, and (b) because law in its highest and best forms is humanistic rather than mechanistic. Both these objections seem particularly apt in the context of the exercise of judicial discretion. Author identifies some disadvantages of judicial discretion and some benefits of legal expert systems.

By reference to a prototype expert system for making bail recommendations, the authors also discuss how the advantages of expert systems can be used to improve the exercise of judicial discretion and how the perceived disadvantages of expert systems can be minimized or avoided. R.M. di Giorgi., et.al.,[208] Opine that at present, legal operators have at their disposal many legal data banks, documenting international and European Commission regulations, national and regional legislation, case law of the different courts and legal doctrine. This information is organized in non-standard documentary structures and has to be consulted in separate data banks utilizing different information retrieval systems. R. Santhanam.,et.al.,[209] Opine that research in knowledge-based systems (KBS) has become an important area of inquiry within decision sciences. In this thesis we determined frequency counts of papers and also performed a content analysis of the papers we surveyed.
The results indicate that there are a large number of studies informing of the design and development issues relating to KBS. However, there seems to be less research examining issues relating to the management and impact of KBS on individuals and organizations. Trevor Bench-Capon., et.al. [253] Opine that Legal knowledge based systems (KBSs) are, by definition, grounded on law. Very often the relevant law is subject to routine amendment and repeal, such changes occurring at irregular and unpredictable intervals. These systems are thus particularly affected by significant problems of adaptation as a result, a fact which has limited their practical take-up. If they are to be of more practical use the maintenance issues associated with these systems must be taken seriously. Stranieri., et.al.,[13] Opine that few automated legal reasoning systems have been developed in domains of law in which a judicial decision maker has extensive discretion in the exercise of his or her powers.

Research on Applying Prolog to Develop Distributed Database Systems has been conducted on a number of different distributed databases, given by Nuno P. Lopes et al. [186]. Implementation of an integrated multi-database for PROLOG systems have been covered as part of large systems by DA Bell et al. [15]. A table oriented database-prolog system approach given by S.M Kuo Pan et al. [19]. In this thesis, we investigate deductive inference for interiors and exteriors of Horn knowledge bases, where interiors and exteriors are those which were introduced by Makino et al [161] to study stability properties of knowledge bases for PROLOG systems. Knowledge-based systems are commonly used to store the sentences as our knowledge for the purpose of having automated reasoning such as deduction applied to them given by Brachman et al [13]. Deductive inference is a fundamental mode of reasoning, and is usually abstracted as follows, as given in the work of Makino and Ono et al. [162]

Karl Popper et al. [159] suggests that we make sincere efforts to refute hypotheses having great content; Lakatos et al. [146] applied the deductive model only within a research program, Even Clark Glymour [33], who rejects some aspects of the hypothetical-deductive method in his bootstrap technique, still holds to the general deductive model in many respects. Thomas Eiter et al. [245] investigate the expressive power and complexity of partial model semantics for disjunctive deductive databases. In particular, partial stable, regular model, maximal stable (M-stable), and least undefined stable (L-stable) semantics for function-free disjunctive logic programs are considered,
for which the expressiveness of queries based on possibility and certainty inference is determined. G Kyrchev et al. [56] work on a single deductive inference of Schwinger realization (interacting boson model—IBM), Holstein-Primakoff realization (truncated quadruple phonon model—TQM) and Dyson realization (finite quadruple phonon model—FQM) of dynamical SU(6) quadruple collective algebra (QCA) was presented with a full scope of their isomorphism on the level of representations. Again on the area we cited a work of Carlo Reverberi et al. [45], Elementary deduction is the ability of unreflectively drawing conclusions from explicit or implicit premises, on the basis of their logical forms. This ability is involved in many aspects of human cognition and interactions.

KB consists of knowledge about problem domain in the form of static and dynamic databases. Static knowledge consists of -Rules and facts which is compiled as a part of the system and does not change during execution of the system. Dynamic knowledge consists of facts related to a particular consultation of the system. At the beginning of the consultation, the dynamic knowledge base often called working memory is empty. As a consultation progresses, dynamic knowledge base grows and is used along with static knowledge in decision making. Working memory is deleted at the end of consultation of the system. Despite its centrality to logic, it is surprising how little foundational work has been done on the topic of deductive inference. Even though there is some work on the question of what an argument is and among these what makes an argument deductive, no similar work can be found when we turn from argument to inference, either directly or through a chain of other inferences, Harman 1973 et al.[104], Goldman 2011 et al. [80], Millar 1991 et al. [123], Mercier and Sperber 2011 et al. [174], second, in the process of acquiring a deductive/inference model. However, the relevant belief was not actually acquired in the process of association, it was remembered. It went from dispositional to occur rent Audi 1994 et al [9]. Knowledge (and probably other epistemological things like justified or warranted beliefs) can be extended through valid deduction cited by Kvanvig et al. [89]. The basic content of logical proposals is that whether or not an inference is deductive or not depends exclusively (almost) on the contents of the initial and acquired beliefs, and not on any accompanying thoughts or features of the underlying cognitive process or mechanism. Borrowing a distinction from Evans et al. 2010 [75]. However, in much of the cognitive science literature, the term “reasoning” is commonly reserved for a specific sort
of inferences, i.e. those where either the goal of the inference or at least some of the initial contents are conscious Sperber & Mercier et al. 2011 [174], Evans 1993. In some literature, the word “reasoning” is used to refer to the ability to perform inferences, so that humans are considered to possess reasoning because we have the capacity to infer Portoraro 2011. Thus, an inference can be deductively valid or deductively invalid, but not deductive itself proposed by Govier et al. [91]. On the other hand, besides the usual problems associated with the individuation of cognitive mechanisms and processes by Goldman2011 [90] to enhance the database query by PROLOG in systematic way and also used the deductive/ inference model.
1.2 Description of Terms and Functions used in Present Study

Corpus

In linguistics, a corpus (plural corpora) or text corpus is a large and structured set of texts (now usually electronically stored and processed). They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistics rules on a specific universe. A corpus may contain texts in a single language (monolingual corpus) or text data in multiple languages (multilingual corpus). Multilingual corpora that have been specially formatted for side-by-side comparison are called aligned parallel corpora. Corpora are the main knowledge base in corpus linguistics. The analysis and processing of various types of corpora is also the subject of much work in computational linguistics, speech recognition and machine translation, where they are often used to create hidden model of part of speech tagging and other purposes.

Parts of speech tagging (POS)

In grammar, a lexical category (also a word class, a lexical class, or in traditional grammar, a part of speech) is a linguistic category of words (or more precisely lexical items), which is generally defined by the syntactic or morphological categories include noun and verb, among others. There are open word classes, which constantly acquire new members, and closed word classes, which acquire new members infrequently if at all. Traditional English grammar is patterned with eight parts of speech.

In corpus linguistics, part-of-speech tagging (POS tagging or POST), also called grammatical tagging or word-category disambiguation, is the process of marking up the words in a text (corpus) as corresponding to a particular part of speech, based on both its definition, as well as its context, that is, Relationship with adjacent and related words in a phrase, sentence, or paragraph. Part-of-speech tagging is harder than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken.

PROLOG

Prolog is a general purpose logic programming language associated with artificial intelligence and computational linguistics. Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative, the
program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations.

Prolog was one of the first logic programming languages, and remains the most popular among such languages today, with many free and commercial implementations available. While initially aimed at natural language processing, the language has since then stretched far into other areas like theorem proving, expert systems, games, automated answering systems, ontology’s and sophisticated control systems. Modern Prolog environments support creating graphical user interfaces, as well as administrative and networked applications.

**Natural Language Processing (NLP)**

In the philosophy of language, a natural language (or ordinary language) is any language which arises in an unpremeditated fashion as the result of the innate facility for language possessed by the human intellect. A natural language is typically used for communication, and may be spoken, signed, or written. Natural language is distinguished from constructed languages and formal languages such as computer-programming languages or the "languages" used in the study of formal logic, especially mathematical logic.

Though the exact definition varies between scholars, natural language can broadly be defined in contrast on the one hand to artificial or constructed languages, such as computer programming languages like Python and international auxiliary languages like Esperanto, and on the other hand to other communication systems in nature, such as the waggle dance of bees. Although there are a variety of natural languages, any cognitively normal human infant is able to learn any natural language. By comparing the different natural languages, scholars hope to learn something about the nature of human intelligence and the innate biases and constraints that shape natural language, which are sometimes called universal grammar. The term "natural language" refers only a language that has developed naturally and hence to actual speech, rather than prescribed speech.
Inference Engine

It consists of inference mechanism and control strategy. Inference means search through knowledge base and derive new knowledge. It involves formal reasoning involving matching and unification similar to the one performed by human expert to solve problems in a specific area of knowledge. Inference operates by using modus ponens rule; control strategy determines the order in which rules are applied. There are mainly two types of control mechanism viz., forward chaining and backward chaining.

Advanced designing and algorithm (ADA)

This algorithm design is a specific method to create a mathematical process in solving problems. Applied algorithm design is algorithm engineering. Algorithm design is identified and incorporated into many solution theories of operation research, such as dynamic programming and divide-and-conquer. Techniques for designing and implementing algorithm designs are algorithm design patterns, such as template method pattern and decorator pattern, and uses of data structures, and name and sort lists. Some current day uses of algorithm design can be found in internet retrieval processes of web crawling, packet routing and caching. Mainframe programming languages such as ALGOL (for Algorithmic language), FORTRAN, COBOL, PL/I, SAIL, and SNOBOL are computing tools to implement an "algorithm design"... but, an "algorithm design" (a/d) is not a language. An a/d can be a hand written process, e.g. set of equations, a series of mechanical processes done by hand, an analog piece of equipment, or a digital process and/or processor.

Recursive function

Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested images that occur are a form of infinite recursion. The term has a variety of meanings specific to a variety of disciplines ranging from linguistics to image processing. The most common application of recursion is in mathematics and computer science, in which it refers to a method of defining functions in which the function being defined is applied within its own definition. Specifically this defines an infinite number of instances (function values), using a finite expression that for some instances may refer to other instances, but in such
a way that no loop or infinite chain of references can occur. The term is also used more generally to describe a process of repeating objects in a self-similar way.

**Computer Aided Design**

Computer-aided design (CAD) is the use of computer systems to assist in the creation, modification, analysis, or optimization of a design. CAD software is used to increase the productivity of the designer, improve the quality of design, improve communications through documentation, and to create a database for manufacturing. CAD output is often in the form of electronic files for print, machining, or other manufacturing operations. Computer-aided design is used in many fields. Its use in designing electronic systems is known as Electronic Design Automation, or ESA. In mechanical design it is known as Mechanical Design Automation (MDA) or computer-aided drafting (CAD), which includes the process of creating a technical drawing with the use of computer software. CAD software for mechanical design uses either vector-based graphics to depict the objects of traditional drafting, or may also produce raster graphics showing the overall appearance of designed objects.

**Program transformation**

A program transformation is any operation that takes a computer program and generates another program. In many cases the transformed program is required to be semantically equivalent to the original, relative to a particular formal semantics and in fewer cases the transformations result in programs that semantically differ from the original in predictable ways. While the transformations can be performed manually, it is often more practical to use a program transformation system (such as Coccinelle, Stratego/XT, TXL, DMS, ASF+SDF or Fermat), a tool that can accept and apply specifications of the required transformations.

**Vector machine**

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. The basic SVM takes a set of input data and predicts, for each given input, which of two
possible classes forms the output, making it a non-probabilistic binary linear classifier. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible.

**MATLAB**

MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran. Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the MuPAD symbolic engine, allowing access to symbolic computing capabilities. An additional package, Simulink, adds graphical multi-domain simulation and Model-Based Design for dynamic and embedded systems.

**Secondary Data Algorithm**

In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning. An algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

**Go tagger software**

Go tagger software is the process of adding geographical identification metadata to various media such as a go tagged or video, websites, SMS messages, QR Codes or RSS feeds and is a form of geospatial metadata. This data usually
Introduction and Brief outline of the program

consists of latitude and longitude coordinates, though they can also include altitude, bearing, distance, accuracy data, and place names. Go tagging can help users find a wide variety of location-specific information. For instance, one can find images taken near a given location by entering latitude and longitude coordinates into a suitable image search engine.

Database Management System (DBMS)

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. Database management systems (DBMSs) are specially designed applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose database management system (DBMS) is a software system designed to allow the definition, creation, querying, update, and administration of database.

Database Indexing

A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and the use of more storage space to maintain the extra copy of data. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records.

An index is a copy of select columns of data from a table that can be searched very efficiently that also includes a low level disk block address or direct link to the complete row of data it was copied from.

Knowledge based System

A Knowledge-based system (KBS) is a computer program that reasons and uses a knowledge base to solve complex problems. The term is broad and is used to refer to many different kinds of systems. The one common theme that unites all knowledge based systems is an attempt to represent knowledge explicitly via tools such as ontology’s and
rules rather than implicitly via code the way a conventional computer program does. A knowledge based system has at least one and usually two types of sub-systems: a knowledge base and an inference engine. The knowledge base represents facts about the world, often in some form of sub-sumption ontology. The inference engine represents logical assertions and conditions about the world, usually represented via IF-THEN rules.

**Deductive/Inference model**

Deductive/Inference model is the action or process of deriving logical conclusions from premises known or assumed to be true. The conclusion drawn is also called an idiomatic. The laws of valid inference are studied in the field of logic.

Alternatively, inference may be defined as the non-logical, but rational means, through observation of patterns of facts, to indirectly see new meanings and contexts for understanding. Of particular use to this application of inference are anomalies and symbols. Inference, in this sense, does not draw conclusions but opens new paths for inquiry. In another definition of Deductive/Inference model, there are two types of inference: inductive inference and deductive inference. Unlike the definition of inference in the first paragraph above, meaning of word meanings are not tested but meaningful relationships are articulated. Human inference (i.e. how humans draw conclusions) is traditionally studied within the field of cognitive psychology; artificial intelligence researchers develop automated inference systems to emulate human inference.

**Preference and Argumentation Model**

Preference model (This Model gives the preference to right verb among the entered wrong verb) that assigns polarity (positive, negative) and a weight to parameters, together with additional parameters among which the target sentence, and the type of verb, etc. We also explain a natural-language-based elicitation of the meeting request details and constraints, and outline the solving of the resulting constrained problem (with preferences model). An Argumentation model, also known as Generalization model is the one that considers the parameters as weighted arguments. It has been utilized in dealing with the problems of verbs starting or ending with capital letters.
Waterfall Model

The waterfall model is a sequential design process, often used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of Conception, Initiation, Analysis, Design, Construction, Testing, production/implementation and Maintenance. The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development.

Rapid Action Development Model

Rapid application development sometimes called fast excessive or RAD model, is a software development methodology that involves methods like iterative development and software prototyping. According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, in rapid application development, structured techniques and prototyping are especially used to define users’ requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in a combined business requirements and technical design statement to be used for constructing new systems. RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating application maintenance.

Knowledge Acquisition

Knowledge acquisition module allows system to acquire knowledge about the problem domain. Sources of Knowledge for ES – text books, reports, case studies, – empirical data and – domain expert experience. Updating of Knowledge can be done using Knowledge acquisition module of the system. – Insertion, – deletion and – updating of existing knowledge.
**Explanation module**

Most expert systems have explanation facilities that allow the user to ask the system why it asked some question, and how it reached to conclusion. It contains 'How' and 'Why' modules attached to it. The sub-module ‘How’ tells the user about the process through which system has reached to a particular solution. Why' sub-module tells that why is that particular solution offered. It explains user about the reasoning behind any particular problem solution. Questions are answered by referring to the system goals, the rules being used, and any existing problem data.

**Advanced Integrated Development Environment**

An integrated development environment (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development, An IDE normally consists of:

- A source code editor
- A Compiler and/or an Interpreter
- A Build automation tools
- A Debugger

IDEs are designed to maximize programmer productivity by providing tightly-knit components with similar user interface. This should mean that the programmer has to do less mode switching versus using discrete development programs. However, because an IDE is a complicated piece of software by its very nature, this higher productivity only occurs after a lengthy learning process. Typically an IDE is dedicated to a specific programming language, allowing a feature set that most closely matches the programming paradigms of the language.

**Flow Graph**

It is the management of data flow in a network to ensure that the receiver can handle all the incoming data. Flow–control mechanism, implemented in both hardware and software, prevent a sender of traffic from sending it faster it faster than the receiver can receive.
Instr, Left, Right Functions of VB

Instr is a powerful VB string function. It's the perfect way to search and test strings in robust Visual Basic 6.0 applications. InStr returns the first occurrence of a string inside another string.

\[
\text{InStr([start,] string1, string2 [, compare])}
\]

Left Returns a string containing a specified number of characters from the left side of a string.

\[
\text{Public Shared Function Left (ByVal str As String, ByVal Length As Integer) As String.}
\]

Right Returns a string containing a specified number of characters from the right side of a string.

\[
\text{Public Shared Function Right (ByVal str As String, ByVal Length As Integer) As String.}
\]

Truth Table

A truth table is a breakdown of a logic function by listing all possible values the function can attain. Such a table typically contains several rows and columns, with the top row representing the logical variables and combinations, in increasing complexity leading up to the final function. In a logic function, there are three basic operations: NOT (also called inversion or negation and symbolized \(-\)), OR (also called disjunction or addition and symbolized \(+\)), and AND (also called conjunction or multiplication and symbolized \(*\)). The values of the functions are normally assigned as logic 0 = false and logic 1 = true.

Boolean Operator

Boolean operators define the relationships between words or groups of words. There are five Boolean operators that can be used to manipulate TRUE/FALSE values. These operators have the meanings, where \(x\) and \(y\) represent values of TRUE or FALSE. The OR operator is often called an inclusive OR, whereas XOR is an exclusive OR. Boolean operators are used widely in programming and also in forming database queries.
Boolean algebra

An algebra in which elements have one of two values and the algebraic operations defined on the set are logical OR, a type of addition, and logical AND, a type of multiplication is known as Boolean algebra. It resembles the algebra of real numbers, but with the numeric operations of multiplication $xy$, addition $x + y$, and negation $-x$ replaced by the respective logical operations of conjunction $x*y$, disjunction $x/y$, and negation $\neg x$. The Boolean operations are these and all other operations that can be built from these, such as $x*(y/z)$.

Software Requirement Specification

A software requirements specification (SRS) is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations. Parameters such as operating speed, response time, availability, portability, maintainability, footprint, security and speed of recovery from adverse events are evaluated.

Scheduling Technique

Scheduling of a software project does not differ greatly from scheduling of any multitask engineering effort. Therefore, generalized project scheduling tools and technique can be applied with little modification to software projects.

Conditional Random Fields

Conditional random fields (CRFs) are a probabilistic framework for labeling and segmenting structured data, such as sequences, trees and lattices. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. The primary advantage of CRFs over hidden Markov models is their conditional nature, resulting in the relaxation of the independence assumptions required by related fields.
Introduction and Brief outline of the program

Word Frequency Counter

Word Frequency Counter (WFC) counts the frequency of words from a single file, multiple files or the clipboard. The many options make it a very useful word counting tool for language analysis and learning. Word Frequency Counter enables user to:
Define words. A word is made up of characters from an alphabet, but there are some characters * that one might or might not want to include in a word definition such as & or 
Define word separators. Word separators are used to divide language into individual words (text segmentation). The space character and punctuation (in the English language) are the most important word separators, but one also needs to decide whether he/she wants to use characters such as & or - as separators. Using word counting software one can add and remove characters which are used as word separators.

Monte Carlo Technique

Monte Carlo methods (or Monte Carlo experiments) are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used in simulating physical and mathematical systems. These methods are most suited to calculation by a computer and tend to be used when it is infeasible to compute an exact result with a deterministic algorithm. This method is also used to complement the theoretical derivations.
Monte Carlo methods are especially useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures.

System Development life cycle

Traditional development mythology called system development life cycle (SDLC) mythology consists of a set of development activities that have a prescribed order. Once a problem or opportunity for a new system is reorganized, a request for developing a new system is forwarded for approval. If approved, a study is conducted to ensure that the proposed system is feasible. If feasible, the system requirements are specified followed by phases of system design, system implementation, testing, conversion and evolution.
Project Planning and Scheduling

Planning is the first step which any Software Engineer follows in its initial position that what are the overall aims for a project, which is a declaration of what one would like to achieve at the objectives, which act as clearly defined stages that make up the project. It consist several points that what are the objectives, specific products to be delivered, activities and milestones resource requirements.

Specific products to be delivered: The products that will be delivered are: - The tested system and network, A robust data base management server, some other supported software, And other supported Hardware Requirements used.

Activity and milestones: The activities in the system, after including the provisions for security are :- Verification of the users, Validation of the data, Security of the system, Better for vast management of the data, Better for vast management of the specification.

Software Scope

The first activity in software project planning is the determination of scope function and performance allocated to software during system engineering should be assessed to establish a project scope that is unambiguous and understandable at the management and technical levels. A statement of software scope must be bounded.

System Design

Design is a meaningful engineering representation of something that is to be built. It can be traced to a customer’s requirements and of the same time assessed for quality against a set of predefined criteria for “good” design. In the software engineering context, design focuses on four major areas of concerns: data, architecture, interfaces, and components.

Software engineering design computer based system but the skills required at each level of design work are different. When the data is at architectural level, Design focuses on pattern as they apply to application to be built. At the interface level, human ergonomics often dictate our design approach level us to effective data and procedural designs.
Testing

Testing is a set of activities that can be planned in advance and conducted systematically. For this reason a template for software testing—a set of steps into which we can place specific test case design techniques and testing methods—should be defined for the software process. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. Strategies must provide guidance for the practitioner and a set of milestones for the manager.
1.3 Outline of the Program

The outline of the thesis is as follows:

First Chapter of this thesis as usual is introductory in nature. A brief historical survey of the work done and the explanations of the used terms and definitions are incorporated in this chapter. It also deals with the description of working of PROLOG systems, and NLP (Natural Language Programming), how PROLOG system and computer system works together with NLP analysis and its formation for English language rules, what are the different computer programs used in PROLOG system, Computer Program for Counting the Part of Speeches, Text Narrations by using Secondary Data Algorithm Techniques, Removal of network anomaly through KBS (Knowledge Based Systems) and making a PROLOG database systems of sentences from the large corpus of English grammar, and what are the computer program for PROLOG systems, NLP and removal of networks ambiguity through KBS (Knowledge Based Systems) and we also explain how database systems work to develop the PROLOG system, and explained the developed software and work which are already in used for the researchers.

Second Chapter deals with the establishment of relation between Computer Program for Counting the Part of Speeches, Text Narrations by using Secondary Data Algorithm Techniques. The second chapter is an attempt in the direction of generation of Primary data by refinement of secondary data available at www.GoTaggersoft.in. The Go tagger software only distinguishes the narrations in text but does not count them. To overcome this limitation, we have embedded our research work with Go tagger and have used portability technique for identifying the narrations. We used advanced designing and algorithm (ADA), recursive functions types of keywords to solve this problem. An application of dialogue systems for developing computer programs has been also described and discussed in this chapter.

Part of this chapter in the form of research paper has appeared in International Journal of Computer Applications, Volume 51– No.11, August 2012, pp. 37-42, ISSN-0975-8887.
Introduction and Brief outline of the program

Third Chapter is an attempt in the direction of Removal of Network Ambiguities through Knowledge based System. Documents on the Internet are composed of several kinds of multimedia information when accessed for personal, entertainment, business, and scientific purposes. There are many specific content domains of interest to different communities of users. Extracting semantic relationships between entities from text documents is challenging task in information extraction. By semantics for natural language in this connection, this chapter understands not just the relating of a semantic representation language to natural language but the evaluation of natural language expressions with respect to databases. Evaluating a declarative sentence (on a given reading) with respect to a database involves determining whether the sentence is true with respect to the data base, whether the sentence appropriately describes the database. Evaluating a question with respect to a database might determine what information in the database would lead to appropriate answers to the question. The implementation of a knowledge-based system that deals with the Very Large Scale requires the important consideration of several problems, including the complexity of the domain, the nature of information processing, and the automation requirements to this problem is the aim of this work. It addresses the incorporation of diverse lexical, syntactic and semantic knowledge in feature-based relation extraction using support vector machines. This chapter has used the base phrase chunking information for relation extraction and has also demonstrated the use of Word Net in feature-based relation extraction to further improve the performance. We used Computer Aided Design; Program transformation; Vector machine keywords to solve this problem. An application of dialogue systems for developing the algorithm has been described and discussed in this chapter.

Part of this chapter in the form of research paper has appeared in International Journal of Computer Application, Volume 57– No.14, November 2012, pp. 31-35, ISSN-0975-8887.

Last Chapter, the Fourth one is dedicated to the attempt in the direction of Development of Prolog for Database Management Systems. This chapter aims at the development of logical programming language (Prolog) for implementing database concepts in the form of establishment of links between logic programming and databases.
We also used Prolog mechanism, Indexing, Knowledge-based systems and Deductive/Inference model types of keywords in this chapter to resolve this problem. We have been able to implement a deductive/inference database management and Knowledge-based systems in Prolog, has been described and discussed in this chapter.

*Part of this chapter in the form of research paper has appeared in International Journal of Computer Applications Volume 71– No.24, June 2013, pp. 01-06, ISSN-0975-8887.*
Estelar
Estelar