Computational Linguistics and Artificial Intelligence: Implications for Computer Assisted Language Learning*

Ali Farghaly
Kuwait University

*Research on this paper was partially Supported by a Kuwait University Grant No. RMU 004

**Ph. D. Linguistics, The University of Texas at Austin, 1981.
Lecturer, English Department, Kuwait University.

Abstract

The first generation of Computer Assisted language Learning (CALL) Programs was developed primarily by programmers and was pedagogically very poor. Second generation CALL software is characterized by the involvement of language teaching professionals in developing ESL software. It is proposed that third generation CALL software should have more reasoning power. The programs should be able to solve the problems that are set to students (Bonnet 1984).

In this paper we present a description of Artificial Intelligence and Computational Linguistics tools that could be used in CALL such as parsing, knowledge representation and expert systems. Both the capabilities and limitations of such tools are discussed and it is proposed that a model for intelligent CALL programs (MICALI) must be developed as a natural language processing system and should employ tools and techniques used in these disciplines. However, these tools are both modified to meet pedagogical requirements and supplemented by a host of other programs (pedagogical component) for error detection, analysis and feedback.

With these tools the third generation of CALL has the advantage over second generation software in that it is highly interactive and communicative. First, it does not store answers to the questions posed to students. Therefor no limitations are imposed on the type of questions in CALL. Creativity of software developers are used to the full. Second, It is bi-directional in the sense that it can either initiate a conversation with a student by asking him questions on a topic chosen by the student or respond to questions posed by students on a previously defined specific domain of knowledge. Although the new generation of CALL can be very powerful, it is realized that, with the present state of the art, it can only operate in limited parsing and domain - specific knowledge representation.

Introduction

The advent of the personal computer in the late seventies and its availability at schools and colleges led educators to think of ways to make use of this new technology for instructional purposes.

Since educators did not know much about this new technology they had to rely on programmers who knew how to manipulate the machine. The first generation of Computer Assisted Instruction (CAI) was characterized, then, by two important properties. First it came into being as a result of the emerging new technology. The question educators had was: "We have a new technology, what can we do with it?" rather than "we have a problem, how can technology help us with it?". Second, almost all CAI software was developed by programmers who knew very little about the principles and techniques of teaching methodology.

Not surprisingly, many researchers and ESL educators expressed disappointment over the quality of most ESL Computer Assisted Language Instruction (CALI) software. The (1982) believes that "the most fundamental problem is that most educational software is written by programmers who know nothing about pedagogy". Stevens (1983) suggests that "professional educators must be integrally involved in its production". For this purpose, books have already appeared (Ahmed, Corbett, Rogers, and Sussex, 1985), (Higgins & Johns, 1984), (Hope, Taylor & Pusack, 1984), and (Underwood, 1984), (Jones & Fortescure 1987) to introduce ESL teachers to Educational software design and evaluation.

A second generation of CALI software came as a result of serious involvement of language teachers and educators in the production of CALI. Many teachers were involved in research over CALI software design and evaluation (Phinney 1987), (Both 1987), (Hill 1987) and (Stevens & Thrush 1987). This second generation was pedagogically more sound, and based on more communicative approaches to language learning. It also benefited from close observation and evaluation of students response to different types of CALI software.

Still most present CALI programs have built-in solutions to the problems presented to the learners and very limited reasoning power. Such programs cannot solve the problems themselves. In many cases the computer rejects students answers simply because they were not listed in the program among the acceptable ones. In a review of CALI software Johnson (1987) reports on his experience with "Essential Idioms" a good software package for teaching English idioms: "The only

problem I had with Essential Idioms is one that is common to much of the language arts software, and that is, the program's inflexibility in accepting meaningful responses". At advanced level, it is almost impossible to list all acceptable responses expected from students. Program developers had to restrict their questions to the type that have limited number of possible answers. This imposes severe restrictions on the type of questions that can be included in CALI programs. Most questions in these programs are objective questions which are more appropriate to testing rather than teaching. Open ended questions and communicative interaction are missing from most CALI software.

One of the main objectives in CAI is to produce intelligent software that can engage students in meaningful interaction. Such software should also be sensitive to errors in students performance, branching whenever appropriate to present remedial materials. Thus combining individualizaion of learning, answer judging and feedback, record keeping, and immediate remedial practice.

Third generation CALI software should be organized around expert systems (Bonnet 1984). So programs of this type should be able to solve the problems they give to students, evaluate and criticize students performance. In language learning that would mean that parsing natural language principles and strategies must be an integral part of CALI programs. The design of efficient parsers has been one of the central goals in Computational Linguistics. The parsing operation (Karttunen and Zwicky 1985) can be viewed as the application of language- specific rules of a sentence (like phrase structure rules S $\longrightarrow$ NP VP) in a way to obtain all grammatical descriptions of that sentence. Parsing a sentence with a grammar (Grishman 1986) means finding a derivation, usually represented as a parse tree of that sentence. If the sentence is ambiguous, there will be several derivations corresponding to the different interpretations of that sentence, unless the ambiguity of the sentence can be resolved by the context in which it occurs. The derivation of a sentence is obtained through the application of the rules in a given order. We may start from the string building up the tree performing lower applications before those higher up, applying reductions one a time (Bottom-Up parsing) until we reach the "S" node, alternatively we may begin with the start symbol working down from the top of the tree (Top-Down parsing). The specific rules are neutral as between these two approaches TD and BU as well as between analysis and generation. They can be viewed either as instructions for the generation of the sentence or as instructions for assigning a structural description to a given sentence.

A parser is a tool used by computational linguists and Artificial Intelligence scientists for a variety of purposes such as:

a - testing the adequacy of grammars
b - parsing the source text in a machine translation system
c - analyzing the input string in a man/machine interface for an advanced information retrieval system.

I will argue that AI tools and techniques such as parsing, knowledge representation and expert systems can improve the quality of existing Computer Assisted Language Learning software. However; to achieve this goal we need to recognize both the limitations and capabilities of these tools, and to define their function in CALI programs which may not be the same as in natural language processing systems. We will also need to develop the system that can take only what it needs from these tools and at the same time supplement them with modules that provide the learner with what he needs.

AI tools may help us overcome some of the major drawbacks in CALI software. Programs could be more interactive departing form the rigid objective questions and allowing for freer communication between students and the computer. It will be easier to introduce variety to the programs, thus they will attract students attention and interest and minimize their boredom. Teaching materials for CALI will be easier to write and material writers will be able to use their creativity to the full since they no longer have to write their materials in the strict format required for present CALI programs.

Computational Linguistics

Language may be viewed as a knowledge-based process (Winograd 1983), and mainly as a system for encoding and transmitting ideas (Kay 1985). This stems from the view of language as a communicative process based on knowledge since both speaker-hearer are processing information. In communication both the speaker and hearer resort to their knowledge of the language, the world and the situation. Computational Linguistics therefore, seeks to understand and model the structure of this knowledge in procedural terms and understand the processes used by the human mind in processing information. Since the computer exhibits the ability to manipulate symbols and make complex processes making decisions on the basis of stored knowledge, it is argued (Winograd 1983) that the view of language as knowledge-based process corresponds closely to the notions of data and program in Computer Science. Data contains the knowledge base on the basis of which a computer can make

decisions whereas computer programs correspond to the mental processes carried out by the human mind. It is a necessary condition for effective communication that participants in a speech event must share the system that pairs sound with meanings. In other words they have the knowledge of what sequences of sound constitute actual words and what these words stand for. A model of this knowledge incorporates linguistic rules at all levels i.e. phonological, morphological, syntactic, semantic ...etc.

This linguistic knowledge-base, though necessary, is not sufficient for communication to take place. Without the processes performed by the human mind on the linguistic knowledge, communication would be impossible. This is not a novel idea. Actual decoding and encoding of messages take place in the human mind as studies on aphasia and other psycholinguistic phenomena show. Damage to certain parts of the brain may cause the loss of one or more aspects of the language faculty (Kelso, Tuller and Harris 1983).

This can form the basis for building computational models of human communication. However; such models are not to be taken as representations of the real world. Many questions concerning the intricacies of human language comprehension remain the most challenging and baffling questions of science and human behavior (Moyne 1985). One obvious distinction between computer models of language processing and human processes is that the analysis of the computer of a sentence has to be done in sequence. That is it has to do lexical analysis before it builds up the syntactic tree. Semantic analysis normally follows syntactic and morphological analysis. Linguists have often identified levels of analysis (or components of grammar such as phonolgy, morphology, syntax and semantics) and assumed that the output of one component would be the input to another (Chomsky 1965). Others (Marslen-Wilson, 1973, 1975; Moyne 1980; Schank 1972, 1975; Wilks 1973; Winograd 1972) have proposed a parallel and interactive approach since all language components seem to integrate effectively in language processing.

Natural Language Parsing

The term "parsing natural languages" refers to computer programs that model the human process of analyzing an utterance and passing judgments of grammaticality on the basis of linguistic rules. Parsing programs, then, must have a linguistic knowledge base that is assumed to model the linguistic competence of native speakers. This means that a parser should have a grammar in the Chomskyan sense (Chomsky

1965), i.e. all internalized knowledge that underlies native speakers intuitions of his language. This is an impossible task in the present state of knowledge in linguistics. Linguists do not claim that they have arrived at a comprehensive account of what native speakers know of their language. In fact, there is no agreement yet among linguists on the form of grammar. However; progress has been made in understanding and explaining several phenomena in human language.

In addition to the grammar, the parser has to have a procedure such that within finite time and space the program must halt and either accepts the input string and outputs the structure (s) it assigns to the sentence, or rejects it for being ill-formed according to the grammar. One important difference between a grammar and a procedure is that the grammar offers a static description of the language without specifying how this description can be used by the machine, whereas the procedure, like an abstract machine, decides what operation to perform next on the basis of a given input and/or what production rules have been applied before. The procedure gives the computer mechanical instructions of a sequence of operations to perform, and each operation must be executable within a fixed amount of time and space (De Roeck 1983).

There are different parsing strategies, such as deterministic and non deterministic parsing, bottom up parse and top down parse, bottom up with top down filtering to reject the application of some rules discovered by the bottom up parser, first parse and all parse strategies, breadth first and depth first, left to right and right to left parse ...etc.; however, it is difficult to claim that one algorithm corresponds more closely to the actual processes of the human mind. Several factors enter into the selection of a parsing algorithm. A "suitable" algorithm for a particular application may not necessarily be "suitable" for another.

A Top-down parser, for example, cannot continue analysis after coming across an ungrammatical constituent in the sentence (Slocum 1981). Therefore it might not be suitable for an application allowing "a fail soft behavior". The choice of right-to-left vs. Left-to-right may be dictated by the nature of the language under analysis. English, for example, is claimed to be better suited to left-to-right parsing. The efficiency of a parser is judged by the validity of its judgments and the range of structures it covers. Parsers are also evaluated according to the memory space, computing time each takes in parsing similar sentences, disk access time, distribution of parse time with respect to sentence length and/or complexity (Slocum 1981).

Different parsers are designed for different grammatical formalisms.

For example a parser for a context free grammar is different from a parser for transformational grammar even if they attempt to cover the same structures. Recently parsers have become available for linguists attempting to develop grammars within a particular grammar formalism. ProGram (Evans 1985) is a development tool for linguists writing grammars in the Generalized Phrase Structure Grammar (GPSG) formalism. ProGram provides computational representations of GPSG grammars and allows the linguist to check the effect of the grammar on the analysis of the structures given by the parser. The parser provides several options (Evans & Gazdar 1984) in ProGram it can either find the first parse of a sentence or all parses. It can also operate in an automatic mode or an interactive mode where it will consult the user at all major decisions. It is developed to be relevant to both applied and theoretical work:

> The ProGram grammar development system is a computational tool to help overcome these problems. As such it can be of use both to the theoretical linguist who wishes to examine the behavior of a grammar, and to the applied computational linguist, who is concerned that the grammar to be incorporated in an language understanding system or a language production system is internally consistent and incapable of assigning spurious analyses. (Evans & Gazdar 1984).

Parsers have been written for other grammatical formalisms for example the LFG Grammar Writer's Workbench was developed at Xerox PARC and Stanford University to aid linguists in writing and testing grammars written in LFG framework. D-PATR is another development environment for Unification - Based Grammars (Karttunen 1986) which is more versatile for it is claimed to be suitable for encoding a wide variety of grammars ranging from Phrase structure Grammars without feature augmentation to Unification-based Categorial Grammars in which all syntactic information are encoded in the lexicon and phrasal building rules are expressed in terms of unification processes of functors with their arguments. Parsers written for a specific grammatical formalism require competence of that framework on the part of the linguist. This is not very desirable in CALI. In many cases such parsers are meant to show how this formalism can handle certain grammatical constructions from a theoretical viewpoint rather than to be of practical use with wide coverage.

PSG (a simple phrase structure parser) requires only competence in writing phrase structure grammars which almost every linguist masters, regardless of his theoretical orientation. A parser of this kind can be very

useful in CALI if both its capabilities and limitations are well understood and supplemented by other programs, as we will show below.

The following is a detailed account of PSG (Bear & Kartuen 1979) which is designed to parse a substantial fragment of English sentences including problematic syntactic structures like relative clauses, WH-Questions and existential sentences. PSG consists of four parts; Input Functions, Parse Functions, Act Functions and Output Functions.

The input functions represent the linguistic input to the PSG parser. This linguistic input which consists of the grammar and the lexicon, is stored in two spearate files. The grammar file contains rules as in (1) and (2).

(1) (S NP VP ((AGREE 2 3 'No)))
(2) (NP NP CON NP ((ASSIGN 1 'No PL)))

Each rule can be interpreted as having a left part and a right part. The left part contains category labels (syntactic categories), whereas the right part of the rules specifies conditions and actions. In (1) the category labels are "S NP VP" and in (2) they are "NP NP CON N.P." The first category label in each rule is the mother node whereas the rest are the daughters. In traditional PSG notation the left parts of (1) and (2) would be written as follows:

S $\longrightarrow$ NP VP

NP $\longrightarrow$ NP CON NP

(1) says that a sentence may consist of an NP followed by a VP. The right hand side of the rule states the condition for this rule to apply which is that the NP and VP agree in Number. In (2) the right hand side specifies an action: to assign the feature "plural" to the higher NP.

The second file that contains the lexicon has a list of entries each contains at least a label and a category as in (3).

(3) (FROM PREP)

Besides a label and a category, a lexical entry may contain other information as subcategorization and other syntactic features.
(4) is an example of a more complex lexical entry.

(4) (WRITES V TR (TENSE PRESENT) (NO SG) (PERSON 3RD))

Tense, No and Person are features that can have different values for different lexical entries.

The input to the parse functions is the list of words that are keyed in for the parser to assign the syntactic structures if the input was grammatical or to reject them if they turn out to be ill formed. The parser works its way through the sentenoe by looking each word up in the lexicon and getting all the information stored in the lexical entry of the word. The information will include the syntactic category of the word and the property list associated with the lexical entry. The parse functions have access to another type of information i.e. the PSRs which specify syntactic structures and conditions and actions for well formedness. The output of the parse functions for the sentence.

Mary Saw John.

will be as follows:

(NP MARY)
(V SAW)
(NP JOHN)
(VP (V SAW) (NP MARY))
(S (NP MARY) (VP (V SAW) (NP JOHN)))

The parser arrives at this structure by first going through the input list of words one by one from left to right. It reads the input word, then looks it up in the dictionary and builds a chart. This chart will have the lexical item and its syntactic category and the property list associated with the lexical item in question such as singular/plural in the case of nouns and transitive/intransitive the case of verbs. The parser then goes to the next word to do the same thing. As the parser adds new information to the chart, it also checks at each step the constituents against the syntactic rules. For example it combines the verb "saw" and the following NP "John" to form a higher constituent "VP". However, the ACT Functions will check the conditions on the rule of the VP. Although the VP rule specifies that a verb followed by an NP may constitute a VP, yet the conditions of the rule says that the verb must be transitive. So, if the sentence has an intransitive verb like "sleep" for example, the parser will reject it as being ill formed. This is where the property list assoiated with each entry becomes very important. At a later stage it combines the VP with the preceding NP "Mary" to from an S. Before accepting the sentence, the parser will check the agreement between the subject and

verb. The Output Functions make the syntactic structure of the sentence available to the user in two forms; list notations and tree diagrams. The user can choose which form the result of the parse would take. If he wants the computer to draw a tree for the input sentence "Mary saw John," the computer would display a tree diagram of the sentence.

If the user asks the computer to display a "flat" representation of the structur of the same sentence, the computer will respond by typing in:

(S (NP MARY) (VP (V SAW) (NP JOHN))

Parsing in Foreign Language Teaching

Educational needs dictate a different strategy in implementing parsing in CAI software. Parsing programs developed by linguists are meant to test theories and grammars of particular languages. For the linguist it is important to display the structure of the input whether in the form of a tree diagram or a list notation. A case in point is the difference in the syntactic relationship of the prepositional phrase in sentences like a and b below:

a - I saw the man with the red shirt.
b - I went to the party with the red shirt.

For the linguist it is of utmost importance that his parser would show that the Prepositional Phrase in (a) is a daughter of the NP whereas in (b) in is a daughter of the VP. The phrase "with the red shirt" can be used to identify the man in answer to the question:

Which man did you see?

but the same phrase cannot identify the party in answer to the question:

Which party did you go to?

The distinction of which constituent is the daughter or parent of which one does not even arise in the mind of the foreign language learner whose main objective is to be able to use the language correctly and appropriately. The emphasis here is not on the analysis of the language but rather on its use.

The language learner may get puzzled by the display of tree diagrams of the deep structure. What the learner needs is useful feedback and reinforcement. In contrast, this is the last thing a linguist expects or needs from a parsing program.

Parsing programs are expensive, difficult to build and very demanding on the memory of the computer. One can understand why most ESL software developers refrained from incorporating parsers in their programs. First, the needs of the foreign language learner, as shown above, are different from the needs of the linguists who originally developed such programs. Second, it is not clear yet how such programs can enhance foreign language learning. Third, incorporating parsing techniques in ESL software will increase the cost of the software and limit its distribution since it will require more advanced; therefore, more costly and less widely used hardware. On the other hand, a closer look at the direction towards the future would confirm that small machines with larger memory and more processing power are being increasingly available and less costly. It is also worth noting that new generation of programming languages like PLNLP (langendoen & Barnett, 1986), LISP and PROLOG that are more suited to natural language processing are being developed.

For example Definite Clause Grammar (DCG) formalism (Pereira 1983) is implemented in PROLOG using its inherent features of TOP DOWN and backtracking. DCG consists of a set of context free phrase structure Rules. Each rule is interpreted by PROLOG as consisting of a main goal (the term of the left hand side of the rule) and one or more subgoals on the right hand side. The following is an example of DCG rules:

a - sentence ———→ np, vp.

b - det ———→ [the].

(a) illustrates the grammar rules whereas (b) is an example of the lexial rules.

This formalism is more convenient for linguists with limited knowledge of programming to input the grammar and the lexicon. Moreover; computational tools for linguistic research (e.g. general purpose morphological analyzers and general parsers) are being developed and will be commercially available. This will drastically reduce the time spent on developing intelligent educational software which in turn will reduce its cost and make the development of more powerful systems much easier than before. Therefore the rewards we get from more powerful ESL software far outweighs the cost.

It is desirable to incorporate in CALI a mechanism that accepts or rejects students' responses by checking their wellformedness on the basis of grammar, and not by comparing each response to a finite set of

previously defined "acceptable" answers. Such a mechanism frees program developers from many restrictions and allows them to write more authentic and creative teaching materials.

Although parsers could make CALI software more powerful, they do not address all the problems in present software. Since our goal is to develop software that could engage learners in meaningful and appropriate interaction, it should be clear that parsing alone, though necessary, is not sufficient for achieving this goal. First, the grammar a parser uses is at best an account of native speaker's linguistic competence. Natural interaction between speaker-hearer is not always a true reflection of their competence. Performance can reflect the native speaker's competence only in idealized situation and in a homogeneous speech community (Chomsky 1965). In second language learning the situation is worse. Learners' performance reflects their imperfect knowledge of the target language, interference of their mother tongue and other factors. Parsers in CALI should be designed in a way to allow teachers to relax some of the rules. Second, meaningful interaction requires understanding of the input. So CALI software has to perform semantic analysis of the input. Most present parsers do not include semantic analysis partly because semantics is that part of grammar that linguists know least of and partly because understanding the meaning of the sentence requires not only linguistic knowlege but also acknowledge of the situation and of the world.

Parsers will not run in the foreground as is the case in a computational linguistics context, but will run in the background of ESL software. The learner will not see the tree diagram on the screen since he is not interested in the underlying structure of the sentence. CALI needs programs that can recognize if students input is grammatical or not. However; recognizers in CALI software will need to be supplemented by other programs to provide learners with appropriate and useful feedback if their input was ungrammatical. Such programs should be built after careful observation and analysis of the type of errors learners normally make.

A recognizer is used as a tool in an ESL software program. It frees the software developer from severe restrictions on the type of questions that can be asked by giving him the chance to include open ended questions and to present real communicative materials to students. Moreover; a parser can be modified in such a way to generate texts. Thus providing detailed and specific comments on students answers.

CRITIQUE (Jensen, Heidron, Richardson and Haas, 1986), a computer system developed at the Research Division of the IBM Thomas Watson

Research Center, demonstrates the power parsing can give to educational software.

CRITIQUE was originally developed as a writing - aid and a tool for improving business correspondence in an office environment. What distinguishes CRITIQUE from other writing systems is that it incorporates a parser (PEG) which covers a wide range of English syntactic structures. PEG has a set of about 235 syntactic rules (Jensen, Heidron, Richardson and Haas, 1986) as well as rules that handle structural ambiguity. PEG has also access to the vocabulary of the Webster's 7th Collegiate Dictionary and to the morphological rules of English. Thus PEG will automatically check both the spelling and morphological endings of words and suggest appropriate corrections. The most interesting part is that PEG will parse the input text and will present the user with a "critique" of the text pointing out to the type and source of errors. Such critiques normally include the following:

a - Grammar
e.g. types of disagreement, wrong pronoun case, incorrect verb form ....etc.

b - Style
e.g. nonparallel constructions, split infinitives, excessive noun modifications ...etc.

c - Words & Phrases
e.g. misspellings, awkward redundant and/or overused phrases

d - Summary Analysis
Total number of paragraphs, sentences, words, active and passive verbs, average number of syllables per word.
Average paragraph length in words and in sentences
Shortest paragraph and longest paragraph ...etc.

Figures (1) and (2) below show CRITIQUE's detection of a grammatical error while processing a text.

I am writing to recommend Susan Hayes, **who's** application you recently received.

Confusion of "who's" and "whose" whose
The word "who's" (which means "who is") and the word "whose" (which is possessive) cannot be interchanged.

Figure 1. Second level of Help includes name of error, suggested correction, and a brief explanation

Source: Richardson & Braden - Harder (1988)

Lest contemplate how a president is selected.
* Let's
In many cases the best candidate in the eyes of
MISSING COMMA
the public is the one who has the most exposure. This is no way to chose a president, but
* choose
unfortunately it is often true. The total package of a candidates political ideas don't really make
* doesn't
an impression on the public. His appearance
FRAGMENT
and mannerisms and the amount of exposure that make him successful.

Figure 2. Example of errors flagged by CRITIQUE

Source: Rechardson & Braden - Harder (1988)

CRITIQUE is a powerful system that can handle any English text. However; it requires a large memory and a powerful processor. It has been developed on the IBM mainframe which is a very expensive machine. It took several years of research to get to its present shape. The components of CRITIQUE include a lexical analyzer which has access to the huge on-line Webster's dictionary, a morphological analyzer which has the morphological rules of English, and the parser which incorporates the grammar including the disambiguating rules which determine in the case of ambiguous sentences which reading is most plausible, and finally rules that handle parsing failure. CRITIQUE also performs stylistic analysis and diagnoses potential style problems and generates statistical information on the lexical and syntactic errors in the text.

Although designing programs like CRITIQUE is very costly, yet it is a powerful and useful tool. It is not difficult to imagine several uses for its capabilities. Recently, The system has been made available to hundreds of users in three major areas of application: office environment, publications organizations and educational institutions. In educational institutions CRITIQUE had to deal with a wide range of ill formed text originating from classes in composition, business writing, technical writing and English as a Second Language. This proved very challenging

and resulted in joint studies with three universities to help test and refine CRITIQUE. (Richardson & Braden-Harder 1988/196).

The possible advantages of incorporating parsing techniques in developing CALI software can be summarized as follows:

1. When supplemented with other components, can provide an interactive mode in the foreign language where communication is authentic and meaningful. This is particularly important in teaching a foreign language rather than a second language.

2. Coupled with advanced graphics and speech capabilities it can simulate real life situations.

3. In foreign language situations, it creates a real communicative atmosphere. The computer becomes, almost like a native speaker, a communicable machine that can relax its requirements at the will of the teacher and can communicate efficiently and individually with all students 24 hours a day in a limited domain.

4. Types of exercises will be more communicative and creative. No need for storing the right answers. The emphasis will be on processing students response rather than on matching it to a previously stored "correct" answer.

5. It can offer informative and detailed feedback on students errors.

6. Reinforcement will be through students success in interacting with the computer and not through congratulating students for doing trivial things as is noticed in present software. Designing programs that incorporate parsers requires mastery of certain programming languages (Farghaly and Brownfield 1985) that are more suited to natural language processing. However; an investment in the required hardware and human experts will result in more meaningful student/computer interaction. The difficulty in designing large and complex programs will be rewarded by the creative use of language in the computer lab.

7. Efforts put into the design of the lexicon and the grammar are never duplicated. Once the parser is there in the computer, it can be used over and over again, even for different purposes, which frees the researchers and gives them the time to upgrade the system.

Limitations of Parsing in CALI

An understanding of the limitations of tools such as parsing is as important as appreciating its capabilities.

1. A parser is as good as the grammar it incorporates. Linguists have not written a complete grammar of any language. Therefore; no parsers exist that can handle general unrestricted texts. Most successful parsers are designed to process a restricted sample of a natural language. Howerver; in language teaching teachers always deal with restricted texts in terms of vocabulary and grammar. Age of students imposes another restriction on the subject matter of teaching materials. Selection of items to be taught and grading them in a particular order is an important component in language teaching methodology.

2. Most parsers are not designed to be error detectors. Parsers do not normally report on the cause of the failure of a parse. Finding out the cause of the failure is a hard, unsolved AI problem.

3. A parser is not sufficient for meaningful and communicative interaction. It only handles grammar, not meaning or inference. It will not reject an inappropiate response from a student if it was grammatical. This could be detrimental in teaching since it gives students the false illusion that they are doing well.

4. Parsers do not provide detailed and informative feedback which learners need most. A lot of work needs to be done in order to turn the failure of a parse into informative feedback to students. We need to identify which error occurred in parsing and associate each error with appropriate and useful feedback.

5. Parsers are not normally designed to process illformed input. In a teaching situation one would expect learners to make many trivial errors. in addition, of course, to serious ones. The problem that parsers cannot make a distinction between the two types of errors. This would lead to students frustration since for the most trivial mistake they make, they will be unable to communicate with the computer. Provision should be made to allow parsing ill formed sentences while generating only well formed sentences.

Artificial Intelligence and Language

The primary goal of Artificial intelligence is to make computers

smarter and more useful to man. To achieve this goal we need to understand what intelligence is. So far psychologists have not arrived yet at a precise definition of what intelligence is but it seems that, among other things, one can assume that intelligence involves perceiving patterns, organizing information, learning from experience and understanding relations. Language for Artificial Intelligence is very important since we learn, get and give information through language. We even think in language. We also acquire the experience of others through language both in written and spoken forms. In fact language plays crucial part in developing cognitive skills. It has been pointed out (Bernstein 1973) that children from working class families show poor progress at school not because they are inferior to Middle Class children in their cognitive abilities but mainly because school adopts the Middle class language and not the working class language. There is no way to account for human intelligence without reference to the human faculty which distinguishes humans from all other species i.e. Language.

We normally communicate with computers through either a programming language or a set of commands for what is called "Application Programs". One important property of programming languages and application programs is that they do not allow ambiguity. The user has to key in the commands literally. Any slight mistake or typing error will result in the failure of the computer to respond properly. A notorious example is what usually happens when a user wants to move from one drive to another in a PC. He is supposed to terminate his command with a colon. If, by mistake, he termiates his command with a semi colon instead of a colon, the computer will respond with an error message. This apparent stupidity of the computer turns many users off because they are frustrated all the time.

AI aims at making communication with computers as easy and flexible as is the case in human interaction in natural language. We can understand ungrammatical sentences and foreign accents. Furthermore; human communication allows for redundancy, ambiguity and poaraphrase. We can infer from a given text things that were not stated explicitly in the text. To get the computer to act similarly AI has to deal, among other things, with a number of challenging problems.

The Paraphrase Problem

It is possible in every language to say the same thing in different ways. Humans have no trouble understanding the paraphrase relation that holds between such utterances. Consider the following:

a - John is unmarried.

b - John is a bachelor.
d - John has never been married.

Although we have no problem in understanding that the above sentences mean the same thing, it is very hard for a computer or even for a formal grammar of English to recognize the paraphrase relation that ties the three sentences together. This problem adefinitely goes beyond syntactic parsing.

The Inference Problem

It seems that what we perceive from a text is much more than what is stated explicitly in the text. Consider the following sentence:

John ate dinner at a restaurant.

We could understand from this sentence that John is not hungry now, and that probably that he was hungry before that. We also understand that probably a waiter or a waitress served him dinner. We also assume that he did not have dinner at home, and that he probably had to pay for his dinner or that someone else paid for him. All these are possible inferences The challenge that AI has to face is how to get the computer to make the inferences that humans normally make from a given text.

The Ambiguity Problem

All natural languages exhibit some degree of lexical and structural ambiguity. Consider the ambiguity of the word "hand" (Schank 1984) in the following sentences:

a - John has a hand.
b - John had a hand in the cookie jar.
c - John had a hand in a robbery.
d - John is an old hand.
e - John gave Mary a hand.
f - John asked Mary for her hand.

It is difficult to define the precise meaning of the word "hand." It has a different meaning in each of the above sentences. Roger Schank (1972) devised a representation scheme that could do the analysis of such sentences. Schank's Conceptual Dependency is meant to represent the content of natural languages utterances in a precise way without any ambiguity. Once an English utterance was translated into this representation, it becomes easy for the computer to express the meaning of the utterance. Moreover; he devised a program that will do the translation automatically.

SAM (Cullingford 1978) is a program that can analyze newspaper stories, produce an English language summary and answer questions about things that were not stated explicitly in the text. For example, if SAM was told in a news story about an accident that John Miller was taken to hospital and was treated and released, SAM would respond to the question "Was anyone hurt?", saying "Yes, John Miller was slightly hurt". It is this kind of program that goes beyond what is stated explicitly to what is being implicated. It does not take the text for its literal meaning, which approximated more closely what we do in human interaction.

Pronoun Reference Resolution

The problem of identifying the right referent for pronouns and anaphors has been the subject of a great deal of research in the last few years (Winograd 1972), (Charnialk 1972) and (Hobbs 1986) and (Rich & LuperFoy 1988). Correct interpretation of pronouns is important for understanding natural language. While Government Binding Theory (Chomsky 1981, 1986) attempts to account for possible coreference and disjoint reference inside the sentence on the basis of syntactic relations like c-command, government, proper government and binding, most AI research in this area aims at developing algorithms for pronoun resolution in discourse on the basis of knowledge of the domain and of discourse (Grosz 1986).

AI research shows clearly that text understanding requires knowledge of the domain which is crucial for determining the meaning of various constituents of an utterance or a sentence. This information can be distinguished from the more immediate context of use which is determined by the previous sentences in the text. Pronouns and anaphors are among the phrases whose interpretation is constrained by context and non-linguistic factors. Consider the following example from (Hobbs 1986):

"There is a pile of inflammable trash next to your car. You will have to get rid of it."

The pronoun 'it' here agrees in number and gender with the antecedent noun phrase 'your car'. Howerver; the right referent of 'it' here is 'a pile of inflammable trash'. The hearer's knowledge of the world and the rules of human behavior makes him pick the right referent. First the whole utterance has the function of a warning or an advice because someone is concerned about the hearer's car. Assuming that the hearer also cares to keep his car out of danger, it follows that he would not get rid of his car, but rather get rid of the danger threatening his car. Second it is counter

intuitive that someone would get rid of something he owns because there is some danger threatening it. This is common sense knowledge. Thus resolving pronoun reference requires more than just looking for the most adjacent possible antecedent i.e. the noun phrase that agrees with the pronoun in number and gender.

Hobbs (1986) presents a system for the semantic analysis of English texts in which he develops an algorithm that out of all possible structures that could serve as an antecedent of a pronoun, it could pick the correct one. This proves Charniak's point (1972) that a complete account of pronoun resolution requires a total system for semantic analysis and that once this is done pronoun resolution comes free.

Realising that it is difficult to adopt one single theory for anaphora resolution, Rich and LuperFoy (1988) has employed a strategy of incorporating different partial theories that would interact to propose candidate antecedents and to evaluate the plausibility of each of the candidate. Figure (3) shows the architecture of anaphora resolution in their system.

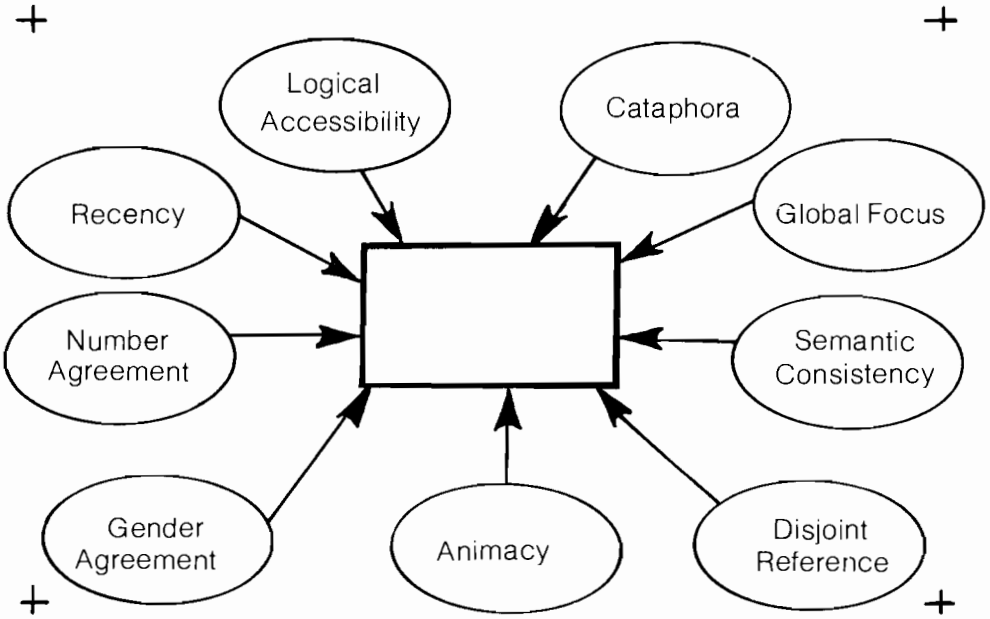

Figure 3: The Architecture of Anaphora Resolution

Source: Rich and LuperFoy (1988).

Knowledge Representation

AI researchers (Hayes-Roth, Waterman and Lenat 1983) emphasize the importance of knowledge representation and knowledge engineering in building expert systems. They believe that machines that lack knowledge are incapable of doing significant work, since the power to do serious work depends on knowledge. Human experts are people who have intellectual knowledge and the ability to apply it skillfully. To design an expert system, then, AI scientists have to find ways to transfer human knowledge and skill into the machine.

Knowledge representation involves feeding the computer with:

a - Data base which represents a set of definite assertions
b - an inference mechanism and a body of rules

Logic programming is widely used in AI programs as a tool for representing both the assertions and the inference mechanism. Consider the following sentence (Moore 1988):

Every man either loves a woman who doesn't love him or is loved by a woman who he doesn't love..

This can be presented in the following way:

(every x ((Man x) ⊃
(($\exists$ Y ((Women Y)
(Loves X Y)
(¬(LOves Y X))))
(($\exists$ Y ((Woman Y)
(Loves Y X)
(¬(Loves X Y))))))))

This May Read as Follows

for every X, if X is a man, there exists a woman (Y) such that the man (X) loves the woman (Y) and that the woman (Y) does not love the man (X); or there exists a woman (Y) such that the woman (Y) loves the man (X) and that it is not the case that the man (X) loves the woman (Y).

If somewhere in the data base, we had Jack and Mary for example, then the computer would know that either Jack loves Mary or that Mary loves jack.

This is a very simple example of how Logic programming is implemented by AI scientists to represent the semantics of natural

languages. Although general purpose systems for natural languages have not been very successful yet, yet those restricted to a limited domain, for example the LUNAR system show impressive results.

Implementing knowledge representation techniques in ESL software involves reducing the context or discourse and both the literal and implied meanings in a teaching lesson into a set of definite assertions. All knowledge related to the topic of the lesson should follow directly from the definite assertions or indirectly from the inference mechanism. Having done this successfully, the computer will be able to judge the validity of students response and comparing the inference the student makes to the inference generated by the inference mechanism of the system. Moreover; in logic programming, the computer can justify its inferences and make detailed statements on how it arrived at its judgment displaying both the facts in the database and the rules he used. This can be very useful as feedback to students which teaches them also clear thinking. Such programs are particularly useful in the teaching of reading as it was noted (Perkins and Jones 1985) that reading tests should reveal whether or not correct inferences from a reading passage have been drawn.

Conclusion

An examination of most present ESL software suggests that it does not fully utilize the capabilities of the computer as an interactive machine where real communication between students and the machine can take place. It is proposed that CALL software can benefit from results of research in Computational Linguistics and Artificial Intelligence and should be developed as a natural language processing system. This may enable ESL software developers design progr that simulare the competence of native speakers of English in a limited domain. Such facility when coupled with the use of advanced graphics and speech capabilities can offer a powerful tool for writing creative, communicative and stimulating ESL software.

Bibliography

Ahmed, K. Corbett, G., Rogers, M and Sussex, R **Computers Language Learining and Language Teaching.** Cambridge: Cambridge University Press. 1985.

Bear, J. and Karttunen, L. "A Simple Phrase Structure Parser," **Texas Linguistic Forum.** Department of Linguistics, The University of Texas at Austin. (1979) No. 15: 1-46.

Bernstein, B. **Class, Codes and Control.** St. Albans, Herts, Paladin. 1973.

Bonnet, A. **L'intelligence artificielle: Promosse's et realite's,** Paris: Inter Edition. 1984.

Both, J. "Communicative Approaches in Computer Assisted Language Instruction." Paper presented at **the TESOL's 21st Annual Convention Miami:** florida. 1987.

Chapelle, C. and Jamieson, J. "Computer Assisted Language Learning as a Predictor of Success in Acquiring ESL." **TESOL Quarterly** (1986). Volume 20, No. 1: 27-46.

Charniak, E. **Toward a Model of Children's Story Comprehension,** AI TR-226. Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1972.

Chomsky, N. Lectures on Government and Binding, Foris, Dordrecht. 1981.
Barriers, MIT Press, Cambridge: Massachusetts. 1986.

Colmerauer, A. "Un Sous-Ensemble Interessant du Francais," **Paper presented at the workshop on Logic and Database,** Toulouse. 1979.

DE Roeck, A. "An Underview of Parsing," in **Parsing Natural Language,** ed. by Margret King, Academic Press. 1983.

Evans, R. **Program: A Development Tool for GPSG Grammars, CSRP 036,** The University of Sussex. U.K. 1985.

Evan, R. and Gazdar, G. **The Program Manual, CSRP 035,** The University of Sussex, U.K. 1984.

Farghaly, A. and Brownsfield, B. "Towards More Intelligent Software," Proceedings of the **First National Symposium on Language Teaching in Kuwait,** Language Center, Kuwait University: 32-51. 1985.

Grishman, Ralph. **Computational Linguistics: An Introduction Studies in Natural Language Processing Series,** Cambridge University Press, Cambridge: 1986.

Hayes-Roth, F., Waterman, D. and Lenat, D. **An Overview of Expert Systems, Building Expert System** Eds. Hayes-Roth, F., Waterman, D. and Lenat, D. London: Addison Wesley: 3-29, 1983.

Higgins, J. and Johns, T. **Computers in Language Learning.** Reading, MA: Addison Wesley. 1984.

Hill, B. "Interactive Video in the Teaching of English." Paper Presented **at the TESOL's 21st Annual Convention,** Miami, Florida Hope, G.R., Taylor, H.F., and Pusack, J.P. 1984, Using Computers in Teaching Foreign Languages. Orlando: Fl: Harcourt Brace, Jovanovitch. 1987.

Hobbs, J.R. "Resolving Pronoun References." In **Natural Language Processing,** Ed. Barbara Grosz et al, Morgan Kaufman Publishers, Inc. Los Altos, California, California: 339 - 352, 1986.

Jensen, K. Heidron, G. Richardson, S. and Hass N. PLNLP, PEG, and CRITIQUE: "Three Contributions to Computing in the Humanities," Paper Presented at **the Conference on Computers and the Humanities,** Toronto, Canada, 1986.

Johnson, Brett. **Essential Idioms in English,** A Review. TESOL's CALL-IS Newsletter, Volume 4, December 1987.

Jones, C. and Fortescue, S. **Using Computers in the Language Classroom.** New York: Longman Publishers. 1987.

Karttunen, Lauri and Zwicky, Arnold **Natural Language Parsing,** Eds Dowty, D.R.: Karttunen, L. and Zwisky, A., Cambridge University Press, 1-25 1985.

Kay, Martin "Parsing in Functional Unification Grammar," **Natural language Parsing,** ed. Dowty, D.R. and Karttunen, L. and Zwicky, A Cambridge Cambridge University Press, 251-278, 1983.

Kelso, S., Tuller, B. and Harris, K. A "Dynamic Pattern," Perspective on the "Control and Coordination of Movement," in The **Production of Speech,** Ed. Peter Mac-Neilage, New York: Springer-Verlag, 137-173, 1983.

Marlsen-Wilson, W. "Linguistic Structure and Speech Shadowing at very short latencies. **"Nature,"** (1973) 244-522-523. "Sentence Perception as an interactive parallel process." **Science** (1975) 189: 226-228.

Moore, R. C. The Role of Logic in Representing Meaning and Knowledge, and ACL tutorial, **Second Conference on Applied Natural Language Processing,** Austin, Texas. 1988.

Moyne, John A. "Language Use: A Performance Model." **International**

Journal of Computer and Information Sciences, (1980) 9: 459-481.
Understanding Language: Man or Machine, New York, Plenum Press. 1985.

Pereira, F. — **Logic For Natural Language,** Technical Report 275, Artificial Intelligence Center, SRI International, 1983.

Perkins, K. and Jones, B. — "Measuring Passage Contribution in ESL Reading Comprehension," **"TESOL Quarterly,** (1985) 19 (i): 137-153.

Phinney, M. — "Writing and revising with Computers in ESL Composition," Paper Presented **at TESOL's 21st Annual Convention,** Miami, Florida. 1987.

Rich, E. and Susann LuperFoy. — "An Architecture for Anaphora Resolution," Proceedings of the Second **Conference on Applied Natural language Processing,** ACL, Austin, Texas, 1988.

Richardson, S. and Lisa Braden-Harder — "The Experience of Developing a Large Scale Language Text Processing System: CRITIQUE," in Proceedings of the Second **Conference on Applied Natural Language Processing, ACL,** Austin, Texas, 1988: 195 - 202.

Schank. R., — "Conceptual Dependency: A Theory of Natural Language Understanding," in **Cognitive Psycholgy,** (1972), 3 (4): 552-631.
Conceptual Information Processing. Amsterdam: North Holland. 1975.
"Intelligent Advisory Systems," in **The AI Business,** ed. P.Winston and K. Prendergast, Cambridge, MA; The MIT Press: (1984), 133-148.

Slocum. J. — **Parser Construction Techniques:** A Tutorial, ACL annual meeting. 1985.

Stevens. V. — "English Lessons on PLATO, A Review," **TESOL Quarterly,** (1983), 17 (2) 293-300.

Stevens. V. and Emily Thrush. — "Use of the Computer Now and in the Future for ESL." Paper presented at **the Software Evaluation Workshop, held at the TESOL's 21st Annual Convention,** Miami, Florida, 1987.

Thé, L. — **Educational Software for the Home, Display Structure,** (1982). (6) 48-52, 102-114.

Underwood. J.H. — **Linguistics, Computers and the Language Teacher,** Rowley, MA: Newbery House. 1984.

Wilks, Y. **Preference Semantics, In Formal Semantics of natural language** Ed. E. Keenan, Cambridge: Cambridge University Press, 1973.

Winograd, T. **Understanding Natural Language.** New York: Academic Press, 1972.
Language as a Cognitive System, Volume 1: Syntax, Reading, MA: Addison - Wesley. 1983.

Winston, P.H. and Horn, B.K.P. **LISP,** Second Edition. Reading, MA: Addison-Wesley. 1984.