ABSTRACT
Starting from first principles, we have designed a rule-based first-order logic (FOL) knowledge-based system (KBS), called EXPARS, and coupled it with an original inductive learning module for the purpose of grammatical inference. While we stress the flexibility of the KBS approach as applied to parsing, we also raise the issues imposed by such a coupling. The interaction between EXPARS and the learning module contributes to the learning of one grammar from positive examples. The proposed approach represents a contribution towards the design of future intelligent parsers.

Categories and Subject Descriptors
D.3.3 [Programming Languages]: Language Constructs and Features. F.2.2 Nonnumerical Algorithms and Problems. F.4 [Mathematical logic and formal languages]; F.4.2 Grammars and Other Rewriting Systems, Parsing. F.4.3 Formal Languages, Classes defined by grammars or automata. I.2.5 Programming Languages and Software, Expert system tools and techniques

General Terms
Algorithms, Design, Languages

Keywords
Expert systems, Inductive learning, Grammatical inference, Parsing

1. INTRODUCTION

Knowledge-based systems (KBSs) are applied to parsing, one of the basic building-blocks of grammatical inference (GI). GI is mainly concerned with the construction of a language from positive and eventually negative examples of sentences [9]. Parsing a sentence involves the use of linguistic knowledge of a language to discover the sentence structure. Parsing represents a discipline in its own right. As of 2007, 1755 bibliographical references have been compiled of which 1100 with notations in [4] and made available over the Web via the link: ftp://ftp.cs.vu.nl/pub/dick/PTAPG_2nd_Edition/index.html. We here focus on problems concerning knowledge-based sentence parsing from design to ordinary usage. KBS area of research is strongly motivated by the deluge of information in the World Wide Web (WWW) for which effective and autonomous information processing tools are still lacking. In order to achieve high-level intelligent information processing, many most challenging research problems in the area of knowledge acquisition, knowledge representation, and knowledge utilization have to be addressed. We expect that future enhancement of knowledge bases will be carried out automatically by using the established and yet to be developed processing technologies to extract new knowledge from the WWW and from various text sources such as XML documents and tagged corpus. This issue requires dynamic knowledge bases that have to be automatically adapted to different languages. Thus “intelligent” parsing, as defined in the sequel.

The fundamental purpose of inductive learning is to offer the possibility of predicting unseen observations (or examples) from given and known evidence. The field of grammatical induction, or grammatical inference (GI), is based on this fundamental idea and can be stated as follows: starting from a set of correct sentences, called positive instances (or examples), i.e. those belonging to a given language, and eventually incorrect ones, called negative instances, automatically generate one grammar of that language. It is a challenging task to attempt since it has been proven difficult to construct a grammar manually. Indeed, it has been established, more than four decades ago, that the space of possible grammars is infinite [3]. On the other hand, verifying that a given grammar generates a set of correct sentences is computationally expensive. Therefore there is a constant need to develop novel methods to deal with issues addressed by GI, given that examples are available for free over the Web for ready use. In the present attempt, our primary interest is to study KBS as a support for GI. We concentrate on positive data, following [10], and specifically in the direction of learnable subclasses of CFGs [5]. The issues addressed in the paper are aimed to have a direct impact not only on knowledge-based systems (KBSs) destined to the processing of formal languages, as related to GI, but also on other distant fields such as control systems [7] and robotics [8]. The design of knowledge-based language processing systems utilizes various forms of knowledge to parse conceptual structures of sentences. Moreover, knowledge bases are evolving under the impetus of automatic knowledge extraction methods as applied to new language processing systems. In order to fill the gap between parsing and KBSs, an original system is proposed, namely EXPARS standing for EXPert systems-based PARSer.

The article is structured as follows. In Section 2, we specify the refined objectives of the proposed system and describe the basic characteristics of EXPARS. In Section 3, we formulate our solution by introducing a three-layer methodology for parsing.
Section 4 reports relevant results. Finally, lessons learned are drawn from the actual results and proposals are highlighted in the conclusion; pointing towards the improvement of the actual work.

2. EXPARS CHARACTERISTICS

Our contribution falls at the intersection of two major fields of research namely GI as a subfield of machine learning, and knowledge based systems (KBSs) considered here within the general field of inductive logic programming. We know that, although naturally related, these fields of research historically evolved independently. Indeed, each of these areas has always had its own scientific community with its ad hoc periodicals, its scientific meetings and its specialized conferences. The specific goal of this paper is to use first-order logic (FOL) as a reasoning scientific meetings and its specialized conferences. The specific goal of this paper is to use first-order logic (FOL) as a reasoning mechanism to infer one possible grammar, in conjunction with a learning module, and thus constructing one grammar, on the basis of a set of positive examples at each iteration of the learning process.

2.1 Refined Objectives

The refined objective is to design an FOL rule-based KBS, called EXPARS, capable of reasoning in forward chaining on assertions related to an unknown grammar to be induced. EXPARS is coupled to a learning module based on the learning algorithm for grammatical inference (LAGI), an adapted version of the so-called Inductive Learning System for Grammatical Inference ILSGInf [6]. Moreover, EXPARS is a system developed from scratch, and as such is easier to update and to adapt for special applications such as the one we are dealing with, i.e. GI. The main characteristics of EXPARS are:

2.1.1 Stand-alone inferences capability

EXPARS is a system based on FOL that can infer knowledge for general-purpose application. In this respect, EXPARS can be compared to standard systems available over the Web, e.g. NASA CLIPS Rule-Based Language, described in [http://www.siliconvalleyone.com/clips.htm].

2.1.2 Ordinary parsing

EXPARS can be used as an Earley-like parser [2] for any context-free language (CFL).

2.1.3 Enhanced parsing

EXPARS can be used to parse parts of sentences instead of entire sentences. As such, EXPARS detects all recognizable syntactic patterns within larger patterns. Perhaps not of great use in parsing per se, enhanced parsing is important in other fields like Bioinformatics where the issue is to identify only parts of very large strings. In our case, enhanced parsing is necessary for learning as shown in the results.

2.1.4 "Intelligent" parsing

EXPARS can infer one unknown linear CFG from positive examples, in conjunction with LAGI. Thus, we propose three different nested layers of parsing offered by EXPARS, i.e. ordinary, enhanced and intelligent layers.

2.2 Inferential Characteristics

The central process in any FOL-based intelligent system is inference, defined here as the ability to add new valid facts to a knowledge base (KB) or to derive the truth of propositions not explicitly contained within the knowledge base. EXPARS has the inferential and complementary characteristics described below.

2.2.1 Rule-based system

Knowledge is rule-based i.e. it is represented by production rules.

2.2.2 First-order, predicate logic

Reasoning is based on FOL (or predicate logic).

2.2.3 Variables

Use of variables is allowed. These are instantiated (or bound) by constants from the fact base. In our case, variables are represented using the notation ?x.

2.2.4 Closed world assumption

Like many logic programming languages (e.g. Prolog), EXPARS works under the closed world assumption, i.e. a goal that is not explicitly expressed in the fact base, or that cannot be inferred from it, is considered false. This assumption does not reduce the capabilities of our system since the grammar contains all information concerning the language under consideration. Indeed, any grammar generates all the instances of the corresponding language. The main difficulty resides in inferring one grammar, not in using it.

2.2.5 Backtrack

In the case of failure, search for a new solution is done by returning to the state preceding actual failure.

2.2.6 Resolution principle

The system does not use the Robinson’s resolution principle. Therefore, it can easily be understood and adapted by a non-specialist.

2.2.7 Forward chaining and backward chaining

The system uses both forward and backward chaining for deriving or proving new knowledge. Only forward chaining is used and described in this paper because of its relevance to the GI task.

2.3 Complementary Characteristics

EXPARS has the following complementary characteristics:

2.3.1 Parsing

As far as parsing is concerned, EXPARS represents an adapted version of Earley algorithm [2]. Detailed accounts on parsing are given below.

2.3.2 Learning

The learning algorithm for grammatical inference (LAGI) is designed in order to undertake the GI process proper.

2.3.3 Integration/Modularity

Taken apart, both EXPARS and LAGI can be used as stand-alone algorithms/systems. When integrated, these are used as a complete GI system [5].

2.4 Notations used in Parsing

We use the standard notations available for formal languages description [1]. A grammar is a quadruplet G = (N, T, P, S) where:

- N is a finite set of non-terminal symbols called variables. Elements of N are used for the generation of sentences of the language.
3. THREE-LAYER PARSING

3.1 First Layer: Ordinary Parsing

The first layer of EXPARS is based on Earley algorithm, an example of chart parser class. Cocke-Younger-Kasami algorithm (CYK) is another example [12]. These algorithms are both based on dynamic programming. The choice of Earley algorithm is dictated by considerations related to complexity and simplicity of implementation. The time complexity of both algorithms is \(O(n^3)\) where \(n\) is the length of the sentence. However, Earley algorithm performs better in most situations. Indeed, it reaches \(O(n^2)\) for unambiguous grammars and \(O(n)\) for LR(k). For the space complexity, Earley algorithm consumes \(O(n^2)\), while CYK needs \(O(n^3)\). Earley algorithm can parse all CFGs, but CYK parses only a finite subset of the Cartesian product \(N \times (N \cup T)^*\) i.e. \(P\) represents the set of rules whose left-hand-side contains one non-terminal symbol and the right-hand-side is any string in \((N \cup T)^*\). Any production or generation rule is represented in the form \(a \rightarrow b\). These productions describe how the sentences of the language are generated. In the sequel, capital letters are used for non-terminals and small letters for terminals.

- \(S\) is a special symbol in \(N\) called initial symbol. The language defined by a grammar is the set of all productions, exclusively composed of terminal symbols and derived from \(S\) by rules in \(P\).

3.2 Second Layer: Enhanced Parsing

In ordinary parsing represented by the first layer above, a sentence is either recognized or refused. In other words, parsing is stopped, perhaps at the outset, due to the first unrecognized character - with no further search, [13]. This limitation characterizes all existing methods like Earley algorithm or its offshoots [11]. In a learning context, as the one we are ultimately considering here, this limitation is a truly severe drawback. Indeed, we want for example to know whether at least some parts of the sentence are correct without getting ejected by the first unrecognized character. Therefore, one of the objectives is to devise a method capable of parsing all that is parsable. In this way, we are able to draw maximum syntactic knowledge from the sentence under consideration. In order to address this issue, we introduce the concept of partial parsing and its corresponding algorithm, the so-called partial derivations constructor algorithm (PDCA). This latter is based on a declarative form of Earley algorithm. In order to describe PDCA as the algorithm responsible for the second layer of EXPARS, we need to introduce some additional definitions.

3.2.1 Additional Definitions

Let \(C\) be a global sentence defined as a sequence of characters in any artificial language. We adopt the following definitions.

- A sub-sentence of a given global sentence \(C\) is any subset of sequence of characters in this global sentence.
- A partial derivation (PaDe) of \(C\) is the parse sub-tree of any sub-sentence.
- Any parsing generating PaDe’s is termed partial parsing, obtained through the so-called partial derivations constructor algorithm (PDCA).
- A list (resp. sub-list) is the result of parsing of a global sentence (resp. sub-sentence).
- Most general grammar: a grammar that, when used with the parser, can parse any new sentence.
- Degree of generality: we say that a PaDet(i) is more general than PaDet(j) if there are less terminals in PaDet(i) than in PaDet(j). If the number of terminals is the same in both PaDe’s, the most general is the one that has the least number of non-terminals and if these are equal then the most general is the one that has the deeper corresponding parse tree.

3.2.2 Declarative Form of Earley algorithm

Our adapted declarative Earley algorithm is described below. Detailed dotted notations are given in [2].
ADD (I_p [?symbol1 → &part1 ?a •&part2, ?q])
DELETE (string ?a &string_remainder)
ADD (string & string_remainder)

RULE 5 /* Filling list l */
IF (I_p [?symbol1 → &part1 ?a •&part2, ?q])
   (I_q [?symbol2 → &part1 •?symbol1 &part2, ?k])
THEN ADD (I_p [?symbol2 → &part1•?symbol1 &part2, ?k])

RULE 6 /* Filling list l */
IF (I_p [?symbol1 → &-•?symbol2 &-, ?q])
   (RULE?symbol2 &part)
THEN ADD (I_p [?symbol2 → •&part, ?p])

RULE 7 /* Parsing of complete string */
IF (string)
   (length ?n)
   (I_n [?symbol → &part•, 0])
   (initial_symbol ?symbol)
THEN ADD (write ("parsing is successfully achieved"))
DELETE (string)

3.2.3 Steps of Partial Derivations Constructor Algorithm (PDCA)

// * Partial derivations constructor algorithm (PDCA) *//
PDCA_Parse
Input: a sentence, and a grammar
Output:
   if sentence can be successfully parsed using grammar,
      then nothing is done;
   else sort in decreasing order of generality all partial derivations (PaDe’s) for all parsable parts of sentence as output;

Begin
parse all parsable parts of sentence using declarative Earley algorithm
   if the sentence is parsable (accepted)
      then return true
   else sort PaDe’s in decreasing order of generality
end.

3.2.4 Example of PDCA Use
Consider the initially recognized global sentence: a*b
Use PDCA to recognize any part of the new global sentence given by: (a*b)
- First layer of EXPARS: result of ordinary parsing is that the new global sentence is refused because of the first unrecognized character “(”.
- Second layer of EXPARS: result of enhanced parsing, construct the following PaDe’s:
  PaDel is (PaDe2 is [PaDe3 is a*b PaDe4 is ] PaDe5 is )
Therefore PDCA recognizes PaDe3 and rejects all other PaDe’s.
- Third layer of EXPARS: or “Intelligent” parsing layer.
  In this context, the so-called learning algorithm for grammatical inference (LAGI) is coupled with EXPARS as explained below.

3.3 Third Layer: Intelligent Parsing
The previously-described layers mentioned ordinary parsing and partial parsing. Emphasis is now made on the learning process through the algorithm (LAGI).

3.3.1 Informal Description of LAGI
The learning algorithm for grammatical inference (LAGI) is based on the exact learning paradigm, based on positive examples. For many practical purposes, only positive examples are needed for grammar inference [10], [14]. It is an incremental learning algorithm; examples are treated one by one. The inferred grammar is written in Chomsky normal form (CNF). Initially, a grammar G_0 is constructed based on the first sentence and written from left to right. Each new sentence is then parsed by the current grammar. Two cases can occur:
- Firstly, if the sentence is successfully parsed by the current grammar, nothing is done.
- Secondly, when only parts of the sentence are either totally parsed, i.e. the root of the parse tree is the initial symbol, or partially parsed, i.e. the root of the parse tree is not the initial symbol, we need a generalization of the current grammar. The generalization process is performed by adding new rules based on the most general partial derivations (PaDe’s), constructed using the partial derivation construction algorithm (PDCA).

3.3.2 Steps of LAGI
// * Learning Algorithm for Grammatical Inference (LAGI) *//
Input: a set of positive examples initially labeled unseen.
Output: one grammar written in CNF, which is either the most general or can generate at least the set of the input examples;

Begin
sentence ← next unseen example
label sentence as seen
current_grammar ← Generate_first_grammar(sentence)
While (the set of unseen example is not empty) do
   sentence ← next unseen example
4. EXPARS RESULTS

For example, describe the behavior of the learning layer of EXPARS as a parser for the following Context-Free Language (CFL)

\[ L = \{ w = a^n b^n, n > 0 \} \]

Third layer behavior: Intelligent parser

Figure 2 describes the use EXPARS as “intelligent” parser for the CFL above. Here, the complete system EXPARS/LAGI is used for learning an unknown language. A set of positive sentences are received and the learning algorithm LAGI generates a grammar for the language based on the input samples.

5. CONCLUSION

To contribute to the stringent issue of grammatical inference from positive examples, on the one hand, and to the enhancement of a knowledge contained in the global sentence. In this way, the system avoids the construction of redundant rules and thus improves the quality of the inferred grammar. The present work represents an early contribution towards truly intelligent parsers especially as learned languages become more and more complex.

6. REFERENCES


Figure 1: EXPARS as a three-layered system

First Layer of EXPARS: global sentence recognizer
- EXPARS used as KBS and acting as ordinary parser

Second Layer of EXPARS: sub-sentence recognizer
- EXPARS used as KBS and acting as enhanced parser

Third Layer of EXPARS: inferring an unknown linear grammar
- EXPARS used as knowledge-based System (KBS) and acting as intelligent parser

Sequence of positive sentences
By Human Expert (Teacher)

LAGI

Inferred Grammar

Facts Rules

EXPARS as sentence recognizer
First Layer Use: ordinary parser

EXPARS as declarative Earley parser
Second Layer Use: enhanced parser

EXPARS contributes in GI
Third Layer: intelligent parser
Figure 2: **EXPARS** behavior for learning the language $L = \{w = a^n b^n, n > 0\}$

**EXPARS** used as “Intelligent Parser”:

**EXPARS** receives a set of positive sentences and the learning algorithm **LAGI** generates a grammar for the language based on the input sample.

The cooperation **EXPARS/LAGI** is used for learning an unknown language.