Realistic Parsing: Practical Solutions of Difficult Problems

SYLVAIN DELISLE [a] & STAN SZPAKOWICZ [b]

[a] Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Québec, Canada G9A 5H7
email: Sylvain_Delisle@uqtr.uquebec.ca, phone +1 819 376 5125; fax +1 819 376 5185

[b] Department of Computer Science
University of Ottawa
Ottawa, Ontario, Canada K1N 6N5
email: szpak@csi.uottawa.ca, phone +1 613 562 5800 ext. 6687; fax +1 613 562 5187

Abstract

This paper describes work on the linguistic analysis of texts within a project devoted to knowledge acquisition from text. We focus on syntactic processing and present some key elements of the project's parser that allow it to deal successfully with technical texts. The parser is fully implemented and tested on a variety of real texts; improvements and enhancements are in progress. Because our knowledge acquisition method assumes no *a priori* model of the domain of the source text, the parser relies as much as possible on lexical and syntactic clues. That is why it strives for full syntactic analysis rather than some form of text skimming. We present a practical approach to four acknowledged difficult problems which to date have no generally acceptable answers: phrase attachment; time constraints for problematic input (how to avoid long and unproductive computation); parsing conjoined structures (how to preserve broad coverage without losing control of the parsing process); and the treatment of fragmentary input or fragments that are a by-product of a fallback parsing strategy. We review recent related work and conclude by listing several future work items.

Key Words

Text processing, knowledge acquisition from text, broad-coverage parsing, parsing conjoined structures.

1. Introduction

The work described here applies natural language processing techniques to knowledge acquisition from actual, unsimplified texts (minimal editing may be required to remove formatting information). The goal is to construct semi-automatically a preliminary conceptual model of the domain described by a given text. A vicious circle must be avoided: acquisition only works well if much knowledge is available. Meaning is therefore gleaned from syntactic clues, which are the only observable aspect of a text. An interactive semantic analyzer suggests interpretations that are next approved or complemented by a knowledgeable user (Delisle *et al.,* to appear). The parser described here does not *by design* rely on domain-specific knowledge prepared in advance. It must function adequately despite the incompleteness of semantic information to hand. Dealing with this self-imposed constraint determines the character of our work.

We now present those technical aspects of DIPETT (Domain-Independent Parser for English Technical Texts) that influence its syntactic coverage and overall performance, and significantly affect practical parsing of

real texts. DIPETT is a fully implemented parser, part of the linguistic subsystem of the TANKA system (Szpakowicz, 1990; Copeck *et al.*, 1992) and the MaLTe system (Matwin and Szpakowicz, 1993; Delisle *et al.*, 1994).

The parser accepts a majority of sentences found in what is commonly considered technical text. With such a broad-coverage parser, acquisition of semantic information from text can be reasonably complete. That is because syntax is the main support for meaning if a rich semantic model is not available. DIPETT (Delisle and Szpakowicz, 1991; Copeck *et al.*, 1992; Delisle, 1994) may handle, fully or partially (see 2.4), as much as 95% of the sentences in actual unsimplified texts. The parser has been used to analyze a variety of actual texts, in particular: an introductory manual of the non-procedural data access language QUIZ, the Canadian Tax Guide, an acoustics textbook and a grade school science reader.

Linguistic theories of syntax (for example, GPSG, HPSG, LFG) usually apply feature structures to the encoding of linguistic objects; the form and content of these structures depend on the underlying theory. DIPETT is not committed to any particular linguistic theory. Its DCG grammar formalism is theory-neutral (as are other formalisms, for example, PATR-II). Most of the grammar rules are based on Quirk *et al.* (1985). DIPETT may be considered as a functional grammar, one in which syntactic analysis is based on syntactic roles instead of only word order.

In addition to standard parsing functions, the parser contains a few non-standard components: a simple tagger, a dynamic dictionary expansion facility, a passive chart, and an error explanation mechanism. A batch mode allows parsing long texts without the user's supervision.

The parser is implemented in Quintus Prolog 3.1 and in SICStus Prolog 2.1/#9. It runs on Sun workstations.

2. Syntactic coverage: practical solutions to difficult issues

2.1 Issues in phrase attachment

DIPETT produces the first parse tree it can find. We assume that it is preferable to show the user a single parse tree instead of "flooding" her with all possible analyses — a strategy used, for example, in the ANLT parser which can produce several thousand different parse trees for one input (see Carroll and Briscoe (1994) and Keenan and Evett (1994)). If DIPETT's analysis is not satisfactory to the user, a post-parsing interactive reattachment module will allow an easy reorganization of the tree before continuing with semantic analysis. This reattachment module, fully designed, will be implemented in the near future. We find that this approach to the parsing of structurally complicated, ambiguous data is less frustrating for the user and saves processing time.

Naturally, the first tree is not always correct. For the QUIZ language manual, on which the parser was tested with particular care, we established by hand that 60-70% of DIPETT's results were accurate: the parser correctly identified syntactic constituents, though it may have misattached constituents such as prepositional phrases. This, then, is syntactic rather than semantic accuracy, which — within our constraints — is inevitably lower than the 60-70% range. DIPETT can be characterized as a noncommittal surface-syntactic parser similar in spirit to what is described by Jacobs and Rau (1993). It finds the first reasonable parse tree, based on its heuristics (and biases) but *no* semantics. It uses a few simple phrase attachment heuristics. For example, prepositional phrase (PP) attachment is controlled by this rule: "if a PP cannot be attached with confidence to a preceding noun (as would, say, a PP introduced by *of*), then attach it to the preceding verb". That is, DIPETT follows the principle of Minimal Attachment (a constituent tends to attach to an existing nonterminal using the fewest additional nodes), unless there is a clue, such as the preposition *of*, for Right Association (a constituent tends to attach to another constituent immediately to its right).

It is particularly important for the interactive semantic analysis immediately following DIPETT (Delisle, 1994; Delisle *et al.*, to appear) to minimize the number of PPs that may be incorrectly buried in an embedded structure and thus overlooked during the Case marker identification phase. In other words, in case of doubt, they should rather be attached to the verb than to the noun. This is one of the parsing biases in DIPETT.

Despite such heuristics — however good they are — misattachment caused by structural ambiguity is unavoidable. Without a rich semantic model of the text under consideration, DIPETT will not always produce an accurate analysis. Even if such a model is available, some ambiguities (for example, "She controlled the computer *with the mouse*") can only be resolved by context or by human intervention. We propose a practical, realistic solution: a post-parsing processing module that will allow the user to revise a parse easily and to correct misattachment when necessary.

It is worth noting that some researchers even question whether it makes good sense to always strive for a parse tree with no misattachment. According to Jacobs (1992: p. 316): "Yet phrase attachment does not always contribute very much to the results of text analysis, particularly in data extraction. [...] The result is that a typical parser spends more time deliberating where to attach phrases whose attachment *doesn't* help semantic interpretation than it does on phrases whose attachment does matter, and parsing is fundamentally misdirected.".

2.2 Time constraints for problematic data

Sooner or later, a *problematic* input will be given to the parser and will engage it in an unacceptably lengthy computation. Our experience shows that such data are either not covered by DIPETT's grammar or are highly ambiguous: the parser is forced to explore a huge search space. A combinatorial effect may arise: two sentences that are parsed quickly may require an unacceptably long parsing time when conjoined in a single input string. DIPETT's grammar is also very large and (due to its lack of semantics) seemingly permissive, so it is sometimes thrown off track in an attempt to parse the sentence somehow, at any cost.

We must handle such extreme situations because our knowledge acquisition method requires finding a sensible parse tree for any sentence or fragment, if at all possible. We have decided to consider *time constraints*: the parser may be timed out. If the maximum time that the parser is allowed to spend on any input is reached before finding a parse, full parsing will be abandoned and parsing into a series of fragments (see 2.4) will be attempted. (There are other approaches to timing-out syntactic analysis, such as Strzalkowski (1992).)

The combination of the CPU time limit, DIPETT's batch parsing mode, and the fragmentary parsing strategy is very useful in practice. Since our knowledge acquisition methodology has no inflexible requirements as far as parsing time is concerned, we have found this combination handy, especially for the parsing of large volumes of text. Only the knowledge acquisition tasks, starting from parse trees (preferably produced ahead of time), must be done interactively. So far, we have dealt with texts of about a thousand sentences that we process in some depth. Batch parsing of a text can be done overnight, with a generous CPU time limit, say 180 seconds on a Sparc-2 for sentences averaging between 15 and 25 words. All the results (including failures) are saved for further processing by the interactive semantic analyzer (Delisle, 1994; Delisle *et al*., to appear).

2.3 Parsing conjoined structures

In the absence of *a priori* semantics, the parsing of conjoined constituents is a difficult problem whose general and efficient solution remains elusive and may never be found unless massive parallelism is employed. We present conjoined structures recognized by DIPETT and explain how it works in the most difficult case of conjoined clauses and verb phrases.

2.3.1 Conjoined structures recognized by DIPETT

The parser deals with seven types of conjoined structures. The grammar does not limit the number of conjoined items in coordinate constructions. (All examples are accepted by DIPETT.)

- main clauses: multiple coordination (*Bob printed a long file, Beth printed a short personal letter,* ***and*** *Tom printed several annual reports*.), correlative coordination (***Not only*** *Sam destroyed his first document* ***but*** *he also destroyed all the subsequent ones.*), and simple coordination (*David could not use his car yesterday* ***and*** *the new battery he had bought was dead.*).

- verb phrases (*Lucy typed her letter, printed it on the laser printer,* ***and*** *sent it to her mother.*).

- noun phrases and nominal clauses: regular conjunction with commas, *and*, *or* (*They accidentally deleted their system file, their new applications,* ***and*** *all their backup files.*), plus the *either-or* and *neither-nor* bipartite constructions (*James hopes* ***either*** *to win the tournament* ***or*** *to finish in second place.*).

- prepositional phrases: regular conjunction (*She sent an application form to the ACME company **and** to the EMCA corporation.*), bipartite constructions (*Bob will go **neither** to the meeting **nor** to the business party.*).

- modifiers: modifiers of verb phrases and sentences that can be joined by conjunctions *and*, *or* (*From my perspective **and** from your point of view, we surprisingly came to the very same conclusion.*), or form a sequence without conjunctions *(Amazingly, suddenly, they decided to launch the operation without any further delay.).*

- adjectives: adjectives that precede the head noun of a noun phrase and can be optionally joined by commas, *and*, *or* (*We went to a long **and** boring presentation.*) or form a sequence without conjunctions (*You now have the useful, new computer data file.*).

- verbs and adverbs: as adjectives just above (*Frank washed **and** dried the dishes.*).

2.3.2 Parsing coordinated clauses with look-ahead

Conjoined sentences and conjoined verb phrases tend to be most expensive to parse. To reduce the cost, we have devised two look-ahead mechanisms that check the lexical categories of tokens ahead of the parser's current position in the input string. The parser uses this information to identify potential joining points for conjoined sentences and to avoid applying rules that cannot succeed.

When the rule for coordinated main clauses, simple or correlative, is invoked, the parser checks whether the input contains potential conjoined verb phrases. One look-ahead mechanism seeks a conjunction in a place that would separate the input into two clauses; the other mechanism checks whether each part contains at least one potential verb. If not, the joining point suggested by the first look-ahead is rejected and another is sought; when none is found, the parser will not attempt to recognize conjoined sentences. If the second look-ahead mechanism finds a potential verb in each part, then and only then does the parser actually attempt to parse conjoined sentences.

The parser also handles a phenomenon specific to conjoined verb phrases: ellipsis of subjects and complements. In a typical situation, illustrated by the sentence *I have **printed and read** my messages*, the parser must recognize three facts: the implicit subject of the second verb phrase is *I*, the elided complement of the first verb phrase is *my messages*, the verb sequence of the second verb phrase is *have read* (with the elided auxiliary *have*). Scoping of negation and adverbs in conjoined verbs is handled, too. For example, based on our heuristics, the text

*John **did not accidentally print and read** my personal messages.*

is analyzed as

[[John] [[did not accidentally print] and [did not accidentally read]] [my personal messages]].

DIPETT deals with all that and represents the missing constituents explicitly in the parse tree.

2.3.3 Top-level parsing heuristics

When DIPETT is given an input string, it must first decide which top-level rules it should attempt and in what order. Ideally, the parser would find an optimal order that minimizes parsing time and maximizes parsing accuracy by first selecting the most promising rules. DIPETT has three top-level rules for declarative statements: 1) MULT_COOR for multiple coordinated sentences; 2) COMPOUND for compound sentences, that is, correlative and simple coordination; 3) NON_COMPOUND for simple and complex sentences, that is, a single main clause with zero or more subordinate clauses (see 2.3.1.).

The simplest strategy would be to apply all three top-level rules in a pre-determined, immutable order. For instance, for any given input string, always apply the top-level rules in the order MULT_COOR, COMPOUND, and NON_COMPOUND — we experimented with this strategy, as well as others, and report on the results at the end of this sub-section. When conjunctions accepted by the parser were numerous, much time was wasted on failed attempts with rules doomed right from the beginning. For example, there is no need to try MULT_COOR and COMPOUND when the data contains only one verb. We have devised a heuristic-based

selection mechanism for top-level rules. (For other examples of the use of heuristics based on syntactic clues see Huyck and Lytinen (1993) and Liu and Soo (1993).)

Top-level rule selection

The heuristic is based on three parameters, obtained after lexical analysis and before parsing: the number of potential verbs in the data, the presence of potential coordinators in the data, and verb density (roughly speaking, it indicates how potential verbs are distributed). For instance, a low density means that verbs are scattered throughout the interval; a high density means that the verbs appear close to each other in the interval, as in a conjunction of verbs such as "$Verb_1$ and $Verb_2$ and $Verb_3$". The latter parameter, computed by a pre-parsing processor, deserves further discussion. (Note: A "potential" verb may actually turn out to be, say, a noun, but only parsing can tell us how such a lexical ambiguity has been resolved. If the input were pre-processed by a tagger, such as those described by Church (1988) or Cutting *et al.* (1992), the ambiguity might disappear and our approach would become deterministic.)

Verb density is evaluated by considering intervals (substrings) between verbs. An interval is represented by four attributes: two integers (the beginning and the end of the substring), the substring's list of tokens and a flag. The flag is true if and only if every token in the interval is *and*, *or*, *but*, *not* or a comma; verbs are said to have a high density in such an interval, typically due to the presence of conjoined verbs. The computation takes into account the number of auxiliaries. But it is non-auxiliary verbs that most influence the result.

There are N+1 intervals in an input string containing N potential verbs:

$$int_0\ \ Auxiliaries_1\ Verb_1\ \ int_1\ \ Auxiliaries_2\ Verb_2\ \ int_2\ \ ...\ Auxiliaries_N\ Verb_N\ \ int_N.$$

For example, the text *Beth and Ruth went to the mall this morning after they had sent off the mail, and they bought several new things for the house* would be segmented as follows (the density flag is *false* for all of the four intervals):

int_0: ['Beth', and, 'Ruth'],

$Verb_1$: went,

int_1: [to, the, mall, this, morning, after, they],

$Auxiliary_2$: had,

$Verb_2$: sent,

int_2: [off, the, mail, and, they],

$Verb_3$: bought,

int_3: [several, new, things, for, the, house].

The algorithm for top-level rule selection is shown in Figure 1. Given the input string's features we have just discussed, it returns a list of up to 3 of the rules COMPOUND, NON_COMPOUND, and MULT_COOR to be attempted when parsing this string. (All verbs here are *potential* verbs. Intervals int_2, ..., int_{N-1} are referred to as *middle* intervals.)

Experiments with rule selection heuristics

We conducted comparative testing of DIPETT's performance with all six permutations of the top-level rules and then with our heuristics. We used three texts, 1664 sentences in all. We found no significant speed difference between the heuristics and the *average* of six fixed orderings. Heuristics may help more if lexical ambiguity is low (e.g., when an accurate tagger is available), in particular to tell verbs from nouns. Heuristics slightly improve parsing accuracy. In one test, 20 of 600 sentences were parsed differently with heuristics than with the best of fixed orderings, 17 of them more appropriately. In another test, 19 of 300 results differed and 7 of them were preferable. Large-scale testing may show, however, that the parser is successful enough with the fixed order: NON_COMPOUND, COMPOUND, and MULT_COOR, perhaps with separate time limits allotted to each of the rules. Preparation for such tests is in progress.

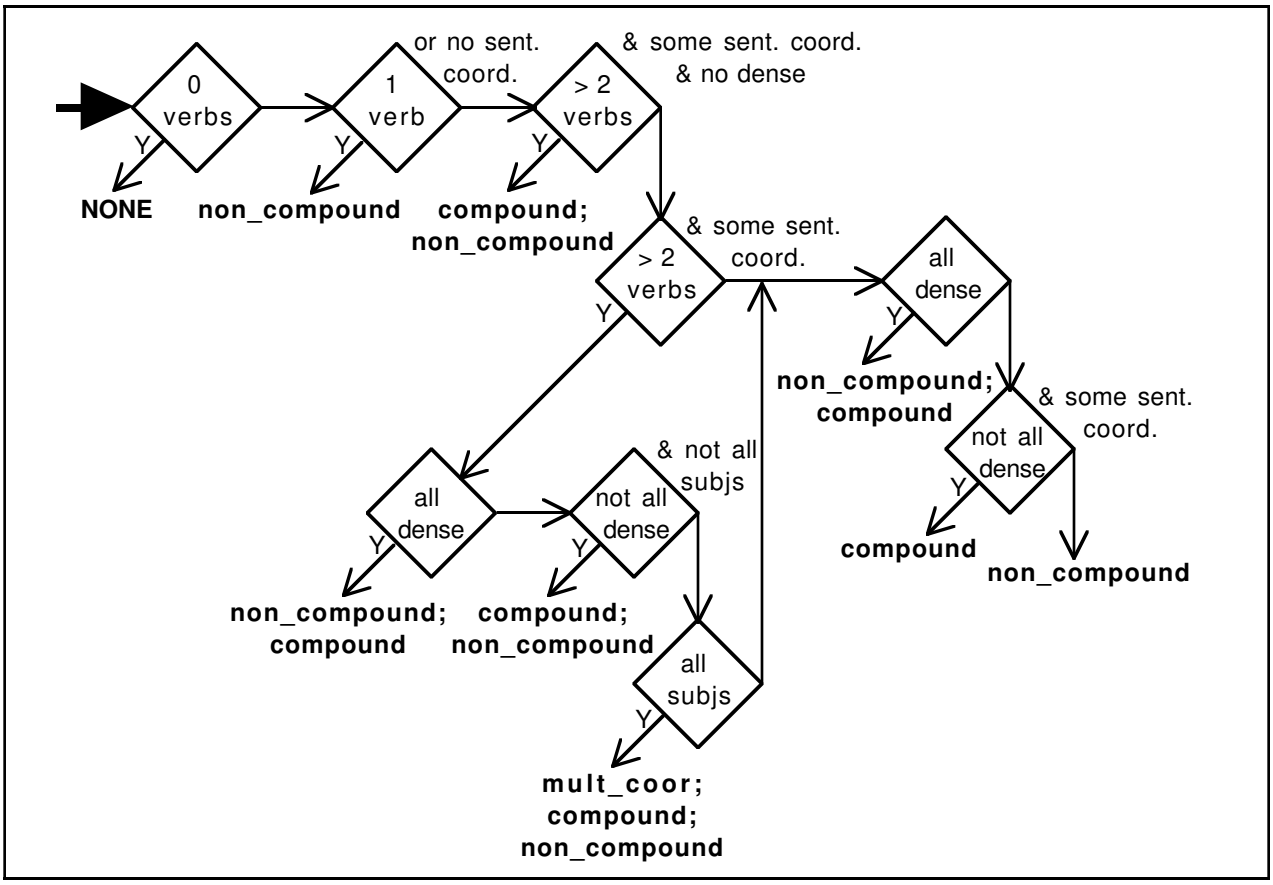

Fig. 1. The algorithm for top-level rule selection.

Legend	
0 verbs	No verbs on input.
1 verb or no sent. coord.	One verb or no sentential coordinators.
> 2 verbs & some sent. coord. & no dense	More than two verbs, no dense intervals, a coordinator in a middle interval.
> 2 verbs & some sent. coord.	More than two verbs, a coordinator in a middle interval.
all dense	All intervals dense.
not all dense & not all subjs	Not all intervals dense, not all intervals have subjects.
all subjs	All intervals have subjects.
not all dense & some sent. coord.	Not all intervals dense, a coordinator in a middle interval.

2.4 Parsing fragments

2.4.1 Fragments in actual texts

It is a long-term goal of our project on knowledge acquisition from text to process free-form technical texts. We assume that non-textual elements (such as graphics or tables) are skipped. A "hole" left by such an element *between sentences* does not affect the structure of the text, but some holes cause fragments to appear. A fragment is an input string that represents a constituent of a grammatical sentence. It may be an "almost complete" sentence or verb phrase, a noun phrase or a prepositional phrase. Fragments — especially noun phrases — also occur in itemized lists.

Since Case analysis (Fillmore, 1968; Somers, 1987) is an important element of semantic processing in the TANKA system, fragments with verbs ("almost complete" sentences and verb phrases) are of particular interest to knowledge acquisition. A non-textual element may play the role of a noun phrase. One strategy we tried was to fill the hole with a dummy noun phrase marker that doubles as a legal noun phrase at this location. This substitution makes the input fully parsable.

2.4.2 Parsing an input as a series of fragments

Fragments matter in other practical considerations. The notion of *fragmentary*, *partial* or *segmentation* parsing has recently gained importance in natural language processing (Hobbs *et al.*, 1993; Chinchor *et al.*, 1993). Full parsing tends to be too time-consuming for certain types of actual, unsimplified data. This observation comes as no surprise, because it has recently become more clearly established and accepted that natural language is NP-complete (for example, see Ristad's complexity thesis (1993)).

In DIPETT, fragmentary parsing — in combination with the time-out mechanism (see 2.2) — is essentially a last-resort parsing strategy. The rationale is this: if a complete analysis cannot be found for a given input (due to a time-up or extra-grammaticality of the input), a fragmentary analysis is still better than no analysis at all. This is also justified from the knowledge acquisition perspective, because even a modicum of new semantic information may turn out to be useful.

Fragmentary parsing in DIPETT produces a list of major constituents. In the following example the time limit was 5 seconds (on a Sparc-2 station). The fragmentary parser shows the time required, the number of fragments, the number of tokens that had to be skipped and the percentage of the input string recognized. The input was analyzed as a series of five substructures: a prepositional phrase (`pp`), a simple sentence (`statement`), an adjective (`attrs`), a nominal clause (`nominal_clause`), and an adverb (`advs`).

```
String #2: In the next few pages you can
learn how to ask the right questions even
if you've never used a computer before .
Process this string (Y/n), try another
Version, or Exit [Y/n/v/e]?

PARSING STOPPED for the following reason:
CPU TIME LIMIT EXCEEDED in ...
verb_clause2 ;
 > the current CPU time limit is: 5.0
second(s).
At that moment, the input string was:
[ask,the,right,questions,even,if,you,have
,never,used,a,computer,before,.]

** Sorry dear user, UNABLE TO PARSE THE
INPUT **

** Reparse as a series of fragments
[n/Y]?
fragmentary parsing  (cpu time) : 2.3
second(s).
** Number of fragments          : 5
** Number of tokens not parsed  : 0
** % of input string recognized : 100.0%

parse_tree__frag_series
  sentence_fragment
    pp  in
{ ... }
  sentence_fragment
    statement
{ ... }
  sentence_fragment
    attrs
{ ... }
  sentence_fragment
    nominal_clause
{ ... }
  sentence_fragment
    advs
{ ... }
  end_of_input  period
```

One final remark on fragmentary parsing: it is our experience that almost every input is parsed quite rapidly, at most in a few seconds on a Sparc-2 workstation, though an exceedingly long chunk of input may still contain a constituent too taxing even for the fragmentary parser (since fragmentary parsing attempts to recognize the longest possible substrings).

3. Recent related work

Phrase attachment, especially prepositional phrase (PP) attachment, is a problem well documented in the literature. For instance, Whittemore *et al.* (1990) evaluate a number of attachment *predictors* (for example, Right Association, Minimal Attachment, Referential Success) on a corpus of text. Based on the evaluation, they define an algorithm which predicts PP attachment. It includes tests such as "associate adjectives with locative (and possibly temporal) qualities with the nouns they modify". It seems quite difficult to describe what constitutes a *locative*, a *temporal*, and a *quality* without assuming extensive *a priori* (ontological, commonsense, domain) knowledge or even human intervention. Such discussions and proposals on PP attachment are common in the literature (Agarwal, 1990; Boggess et al., 1991; Hindle and Rooth, 1991; Hobbs and Bear, 1990). See also Brill and Resnik (1994) who describe a system that automatically learns simple rules based on statistics collected during training on a corpus and predicts correct PP attachment based on contextual (surface-syntactic) information.

Timing-out mechanisms are used in a variety of modern NLP systems (for example, the TTP system (Strzalkowski, 1992)). They are particularly helpful in large applications dealing with complicated texts. For example, the PRC system, participating in the MUC-3 competition, has such a mechanism: "The parser also used a time-out mechanism: for sentences taking more than a threshold time (approximately 50% of the corpus), the parser simply returned the longest parsed substring or a run-on sentence analysis." (Chinchor *et al.*, 1993: p. 428). Another option is the use of anytime algorithms (Görz and Kesseler, 1994) in which a useful but not necessarily maximally fine-grained result must be produced under a given time constraint.

One of the most difficult problems in parsing, the treatment of conjoined elements, has not yet been satisfactorily solved. Coordination is difficult because it encompasses structural ambiguity, elided elements and the handling of fragments. A stock example is Woods' *John drove the car through and completely demolished a window* (*John* is elided from the second conjunct and *a window* from the first). Lavelli and Stock (1990) use an expectation-based algorithm that deals with intersentential ellipsis and coordination in a chart-based parsing framework, Agarwal and Boggess (1992) propose a deterministic and domain-independent algorithm for conjunct identification applicable to texts pre-labelled with syntactic and semantic tags, whereas Kempen (1991) presents an inheritance-based approach to the problem of clause-level coordination in Dutch, based on the psycholinguistic model of self-correction. Similarly to what we described above in 2.3.2, (Kurohashi and Nagao, 1994) present an approach in which they first find candidate conjoined components before actually parsing the complete input. What differs considerably is the detection process itself which is much more complicated than ours and their scheme also appears to be quite specific to Japanese.

Fragment processing and "partial parsing" have recently gained importance, especially as the basis of a fallback approach to the parsing of complicated data (including ungrammatical and extra-grammatical strings). When an input cannot be analyzed in a pre-determined time, the processor reconsiders it as a series of fragments, producing a series of simpler parse trees instead of a much larger unique sentential parse: this is exactly what DIPETT does. In this vein, Grishman *et al.* (1992) discuss fitted parse, Jacobs (1990; 1992) text skimming and text segmentation, Hobbs *et al.* (1993) fragments and terminal substring parsing, McDonald (1992) and Lavie (1994) partial parsing, and Li *et al.* (1990) segment parsing.

Partial parsing is considered essential for the robustness of parsing: "For the syntax-driven systems, only those systems that provided a robust parsing capability performed well. Among the top-performing systems, there were two successful approaches to robust parsing: use of partial parsing and use of heuristics added to a standard full-sentence parser that allowed recovery of parsed substrings." (Chinchor *et al.*, 1993: p. 443).

4. Conclusions and future work

The parser has been validated on a comprehensive test suite and on a few sizable texts. Several students have thoroughly tested the parser on a few larger, virtually unedited texts. Areas that require tuning or upgrading have been identified; certain parser biases have turned out to be inappropriate. At the same time, we will further extend DIPETT's syntactic coverage by making it

recognize correctly or more accurately, among others, reflexive pronouns, genitives with implicit conjunction, some unusual punctuation, itemized lists, phrasal verbs, sentential relative clauses, inserted optional modifiers, and broader range of questions.

The parser has recently been ported to SICStus Prolog in order to use a built-in Prolog execution profiler for a detailed analysis of DIPETT's performance. The lessons learned will be used to change or eliminate some of the parser's biased. The implementation of the reattachment module will begin during the coming year.

The main accomplishment of our work — of which the solutions reported in this paper are an element — is the creation of a satisfactorily robust, broad-coverage parser capable of tackling unsimplified texts. Its results are now being used in semi-automatic knowledge acquisition from text, a challenging application that crucially relies on syntax for success.

Acknowledgments

This work has been supported by the Natural Sciences and Engineering Research Council of Canada. Our thanks to the members of the TANKA team for all the cooperation, and to Jean-François Delannoy in particular for comments.

References

R. Agarwal and L. Boggess, "A Simple but Useful Approach to Conjunct Identification", *Proc. of 30th Annual Meeting of the ACL*, 1992, 15-21.

R. Agarwal, "Disambiguation of Prepositional Phrase Attachments in English Sentences Using Case Grammar Analysis", M.Sc. Thesis, Department of Computer Science, Mississippi State University, 1990.

L. Boggess, R. Agarwal and R. Davis, "Disambiguation of Prepositional Phrases in Automatically Labelled Technical Text", *Proc. of AAAI-91*, 1991, 155-159.

E. Brill and P. Resnik, "A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation", *Proc. of COLING-94*, Kyoto (Japan), 1994, 1198-1204.

J. Carroll and T. Briscoe, "Integrating Probabilistic Knowledge-based Approaches to Corpus Parsing", *Proc. of Workshop on Computational Linguistics for Speech and Handwriting Recognition* (Leeds University, England), April 12th 1994.

N. Chinchor, L. Hirschman and D. D. Lewis, "Evaluating Message Understanding Systems: An Analysis of the Third Message Understanding Conference (MUC-3)", *Computational Linguistics*, 19(3), 1993, 409-449.

K. W. Church, "A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text", *Proc. of 2nd Conf. on Applied Natural Language Processing* (Austin, Texas), 1988, 136-143.

T. Copeck, S. Delisle and S. Szpakowicz, "Parsing and Case Analysis in TANKA", *Proc. of COLING-92* (Nantes, France), 1992, 1008-1012.

D. Cutting, J. Kupiec, J. Pedersen and P. Sibun, "A Practical Part-of-Speech Tagger", *Proc. of 3rd Conf. on Applied Natural Language Processing* (Trento, Italy), 1992, 133-140.

S. Delisle and S. Szpakowicz, "A Broad-Coverage Parser for Knowledge Acquisition from Technical Texts", *Proc. of 5th International Conf. on Symbolic and Logical Computing — ICEBOL5* , Madison, S. D., USA, April 1991, 169-183.

S. Delisle, K. Barker, J.-F. Delannoy, S. Matwin and S. Szpakowicz, "From Text to Horn Clauses: Combining Linguistic Analysis and Machine Learning", *Proc. of 10th Canadian Artificial Intelligence Conf.*, Banff, May 16-20, 1994, 9-16.

S. Delisle, K. Barker, T. Copeck and S. Szpakowicz, "Interactive Semantic Analysis of Technical Texts", to appear in *Computational Intelligence*.

S. Delisle, "Text Processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic Analysis for Incremental Knowledge Acquisition", Ph.D. thesis, Department of Computer Science — Ottawa-Carleton Institute for Computer Science, TR-94-02, University of Ottawa, 1994.

C. Fillmore, "The Case for Case", in E. Bach and R.T. Harms (eds.), *Universals in Linguistic Theory*, Holt, Rinehart and Winston, 1968.

G. Görz and M. Kesseler , "Anytime Algorithms for Speech Parsing?", *Proc. of COLING-94*, Kyoto (Japan), 1994, 997-1001.

R. Grishman, C. Macleod and J. Sterling, "Evaluating Parsing Strategies Using Standardized Parse Files", *Proc. of 3rd Conf. on Applied Natural Language Processing*, 1992, 156-161.

D. Hindle and M. Rooth, "Structural Ambiguity and Lexical Relations", *Proc. of 29th Annual Meeting of the ACL*, 1991, 229-236.

J. R. Hobbs and J. Bear, "Two Principles of Parse Preference", *Proc. of COLING-90*, 1990, 162-167.

J. R. Hobbs, D. E. Appelt, J. Bear and M. Tyson, "Robust Processing of Real-World Natural-Language Texts", *Proc. of 3rd Conf. on Applied Natural Language Processing*, 1992, 186-192.

C. R. Huyck and S. L. Lytinen, "Efficient Heuristic Natural Language Parsing", *Proc. of AAAI-93*, 1993, 386-391.

P. S. Jacobs and L. F. Rau, "Innovations in Text Interpretation", *AI Journal* 63(1-2) (Special Issue on Natural Language Processing), October 1993, 143-191.

P. S. Jacobs, "To Parse or Not to Parse: Relation-Driven Text Skimming", *Proc. of COLING-90*, 1990, 194-198.

P. S. Jacobs, "Parsing Run Amok: Relation-Driven Control for Text Analysis", *Proc. of AAAI-92*, 1992, 315-321.

F. Keenan and L. Evett, "Applying Syntactic Information to Text Recognition", *Proc. of Workshop on Computational Linguistics for Speech and Handwriting Recognition* (Leeds University, England), April 12th 1994.

G. Kempen, "Conjunction reduction and gapping in clause-level coordination: an inheritance-based approach", *Computational Intelligence*, 7(4), 1991, 357-360.

S. Kurohashi and M. Nagao, "A Syntactic Analysis Method of Long Japanese Sentences Based on the Detection of Conjunctive Structures", *Computational Linguistics*, 20(4), 1994, 507-534.

A. Lavelli and O. Stock, "When Something Is Missing: Ellipsis, Coordination and the Chart", *Proc. of COLING-90*, 1990, 184-189.

A. Lavie, "An Integrated Heuristic Scheme for Partial Parse Evaluation", *Proc. of 32nd Annual Meeting of the ACL*, Lac Cruces (NM), 1994, 316-318.

W.-C. Li, T. Pei, B.-H. Lee and C.-F. Chiou, "Parsing Long English Sentences with Pattern Rules", *Proc. of COLING-90*, 1990, 410-412.

R.-L. Liu and V.-W. Soo, "An Empirical Study on Thematic Knowledge Acquisition based on Syntactic Clues and Heuristics", *Proc. of 31st Annual Meeting of the ACL*, Columbus, Ohio, 1993, 243-250.

S. Matwin and S. Szpakowicz, "Text Analysis: How Can Machine Learning Help?", *Proc. of First Conf. of the Pacific Association for Computational Linguistics (PACLING)*, Vancouver, Canada, 1993, 33-42.

D. D. McDonald, "An Efficient Chart-based Algorithm for Partial-Parsing of Unrestricted Texts", *Proc. of 3rd Conf. on Applied Natural Language Processing*, 1992, 193-200.

R. Quirk, S. Greenbaum, G. Leech and J. Svartvik, *A Comprehensive Grammar of the English Language*, Longman, 1985.

E. S. Ristad, *The Language Complexity Game*, The MIT Press, 1993.

H.L. Somers, *Valency and Case in Computational Linguistics*, Edinburgh University Press, 1987.

T. Strzalkowski, "TTP: A Fast and Robust Parser for Natural Language", *Proc. of COLING-92* (Nantes, France), 1992, 198-204.

S. Szpakowicz, "Semi-Automatic Acquisition of Conceptual Structure from Technical Texts", *International Journal of Man-Machine Studies*, 33, 1990, 385-397.

G. Whittemore, K. Ferrara and H. Brunner, "Empirical Study of Predictive Powers of Simple Attachment Schemes for Post-modifier Prepositional Phrases", *Proc. of 28th Annual Meeting of the ACL*, 1990, 23-30.