Theory-Neutral Parser Engineering

Sylvain Delisle[1] , Elizabeth Scarlett[2] & Stan Szpakowicz[2]

[1] Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Québec, Canada, G9A 5H7
Sylvain_Delisle@uqtr.uquebec.ca

[2] School of Information Technology and Engineering
University of Ottawa
Ottawa, Ontario, Canada, K1N 6N5
{scarlett, szpak}@site.uottawa.ca

Abstract

Developing and, above all, maintaining a large, broad-coverage parser can be a serious exercise in software engineering. While a parser trained on an annotated corpus may be maintained by retraining on another corpus, such corpora are scarce enough for much parser-writing to require linguistic introspection. We argue that intuitiveness and readability of the underlying grammar is an essential property of a successful parser, and that it helps a lot if the results are also intuitive and readable. Given such intuitiveness, it is easier to achieve the parser's robustness, improve its coverage and its portability to new domains and applications, and generally to increase its usefulness.

---

1. Parser Engineering and Test Suites

In the last decade or so, language engineering has become an important area of natural language processing. Netter & Pianesi (1997) write:

> "With a growing number of NLP applications going beyond the status of simple research systems, there is also a more evident need for better methods, tools and environments to support the development and reuse of large scale linguistic resources and efficient processors. This new area of research, often referred to as Linguistic Engineering, is rapidly gaining interest besides the more traditional ones concerned with formalisms or algorithm studies and development. Aspects of linguistic engineering range from grammar development environments, through the construction and maintenance of large scale linguistic resources, to methodologies for quality assurance and evaluation."

In this paper, we express the view that this new area of NLP benefits from the more general insights of good software engineering practice. In particular, we argue that readability of a parser's linguistic core can be a determining factor in the development of a robust NLP system.

Parser engineering—the business of developing and maintaining large-scale, broad-coverage parsers—is a peculiar brand of software engineering. Although even non-trivial parsers may be small by the contemporary software engineering standards, they usually require highly specialised expertise and labour-intensive testing. They also are highly sensitive to minor changes in the underlying grammar, which can have a dramatic effect on the parser's output.

Related work on various aspects of NLP system testing is presented in Black *et al.* (1992), Boguraev *et al*. (1988), Li *et al*. (1998), Schmidt *et al*. (1996), Srinivas *et al.* (1995), and Volk (1992).

Parsers can be derived from an annotated corpus, though such a derivation does little more than recover the grammatical principles encoded by the annotators, along with all unintentional annotation errors. Claims have been made that only trainable, statistical, parsers are robust enough to be successfully fielded—see, for example, Hatzivassiloglou (1994), Magerman (1994), Zelle & Mooney (1994), Lawrence *et al.* (1996), Charniak (1997), and Collins (1997)—but robustness depends on the quality of the requisite "treebank". It would appear that the effort of a virtually error-free annotation is not much different than the effort of designing and hand-coding a parser based on linguistic introspection. Once such a parser has been constructed, its maintenance can be done according to good software engineering practices, especially if it has been developed using a parser writer's workbench—see, for example, Erbach (1992), Baldwin *et al.* (1997), Cunningham *et al.* (1996), and several papers in Estival *et al.* (1997).

A parser constructed using Machine Learning techniques may be impossible to verify manually, or its verification may require very advanced expertise. While the performance on test sets reasonably close to training sets is usually demonstrably adequate, such a parser may, in practice, be not maintainable. Hybrid methods that combine symbolic and statistical tools offer an interesting compromise, particularly when a parser based on linguistic foundations is being augmented by considering statistical data. See, for example, Voutilainen & Padró (1997) and "Balancing" (1994).

Test suites are fairly widely used to test and validate parsers. The TSNLP project produced well-known test suites intended to be broad-coverage, multi-purpose, multi-user, multi-lingual, and reusable (Lehmann, 1996). An important issue in the TSNLP design was to consider the benefits of using a test suite rather than a corpus. Some of the advantages are as follows (Balkan *et al.* 1994a, b).

Control over test data. Test suites allow for systematic construction of test data focused upon specific phenomena. The phenomena may be tested alone or in combination with others.

Systematic coverage of phenomena. Tests suites may provide test items that systematically test variations over a specific phenomenon.

Non-redundant representation of phenomena. Systematically constructed test suites test a phenomenon only once.

Negative examples. Testing negative examples may be useful in diagnostic evaluation of any NLP application. Ungrammatical test items may be included in a test suite.

Annotation of test items. Test suites may be annotated with items specifically relevant to diagnostic evaluation.

Balkan *et al.* (1994c) identified potential shortcomings of test suites: too little coverage; lack of systematicity; lack of documentation and annotation; too much system-specificity, which prevents reusability; too much difficulty in interpreting the testing results. The TSNLP method is designed to optimize *control over test data*, *progressivity*, and *systematicity* in order to construct a test suite that is adequately broad-coverage and offers the advantages over corpora stated above (Lehmann, *op. cit.*) while also avoiding the shortcomings listed. The TSNLP design requires linguistic phenomena to be tested in isolation, not in combination, and so each test item contains only one phenomenon that distinguishes it from other similar test items. The TSNLP data are designed to test exhaustively the linguistic phenomenon covered, including the treatment of closed class items in a language. The TSNLP suite is thus very fine-grained and pinpoints problems very precisely. The test items for the basic sentence phenomena covered (complementation, modification, diathesis, modality, tense and aspect, clause type, coordination, and negation) are exhaustively complete. Given such a robust and precise diagnostic tool, the difficulty in parser maintenance is to identify the places in the

parser responsible for errors in test runs. Grammar rules that encode intuitive relationships between syntactic units and employ meaningful names facilitate correction.

This paper presents a parsing system that we believe to be sufficiently natural and intuitive to be readily maintained and expanded. Our research group has used this system for over a decade, and it has been occasionally used by other researchers. It had been tested, adapted, and extended in non-trivial ways—see, for example, Delisle *et al*. (1994) and Barker (1998). It is still in use and has been very recently ported to LPA Prolog in Windows NT. It has been recently comprehensively retested; the findings have been incorporated in an even more accurate grammar. We briefly discuss the system in Section 2, and we show its representation of linguistic knowledge in Section 3. Section 4 argues that, as our work with the parser has demonstrated, intuitiveness and naturalness have a beneficial effect on improving coverage and performance. In Section 5, we offer a few simple conclusions.

2. The Text Analysis Project

The development of an AI system often requires the building of a new knowledge base. Knowledge bases have been constructed by a cycle of interviewing experts, building a prototype, testing, then re-interviewing the expert. Knowledge acquisition, therefore, remains a difficult, expensive, and labour-intensive task. This is commonly called the *knowledge acquisition bottleneck*—see (Buchanan & Wilkins, 1993). Applications that analyze text for the purpose of knowledge acquisition aim to automatically, or semi-automatically, extract knowledge contained in written texts and encode it in a knowledge base. The TANKA[i] project (Barker *et al*., 1998) seeks to build a model of a technical domain by semi-automatically processing written text that describes the domain. No other source of domain-specific knowledge is available.

DIPETT[ii] is a broad-coverage parser of English technical text (Delisle, 1994), and is used primarily in the TANKA project. Broad-coverage parsing is required to parse large-scale technical text. The accuracy and completeness of a semantic representation generated by TANKA is partly determined by the accuracy of DIPETT's syntactic analysis of the text. DIPETT is a *theory-neutral* parser, in that it does not presuppose any particular linguistic theory of language, and it has a broad surface-syntactic coverage of English. The core of DIPETT's grammar is based on Quirk *et al.* (1985), a standard academic grammar of English, and on Winograd (1983). DIPETT is particularly rich in "look-ahead" and heuristics for efficient parsing of complex syntactic structures.

3. Linguistic Knowledge and the DIPETT Parser

We consider DIPETT robust because of its broad coverage and its handling of extra-grammatical data. DIPETT almost never rejects a fully or nearly correct input in English, producing at least a partial analysis. We will look at DIPETT as a domain-independent stand-alone syntactic processor, unrelated to a specific application or performance task. We will present some high-level technical comments (section 3.1), the underlying linguistic knowledge (section 3.2), and the partial-parsing facility (section 3.3).

3.1 A Few Technical Details

DIPETT has been implemented in Prolog. Its underlying grammar, expressed by means of DCG rules, encompasses much of one of the standard academic grammars of English, described by Quirk *et al.* (1973, 1985). Its syntactic analyses employ their terminology.

---

[i] Text Analysis for Knowledge Acquisition

[ii] Domain Independent Parser of English Technical Text

DIPETT is a surface-syntactic parser, in that it does not account for many lexically motivated semantic restrictions on grammaticality, presented by Quirk *et al.* It recognizes most of the structural variety, but it overgenerates by accepting semantically anomalous constructions (an inevitable limitation of parsers not "primed" with rich semantic information, for example in an extended lexicon). DIPETT is a working computational model of a sizable subset of the English surface syntax.

A detailed classification of parsers has been proposed by M. A. Covington in 1993 (it was circulated in the 'comp.ai.nat-lang' Usenet newsgroup). The classification helps list, systematically, a parser's main technical and algorithmic aspects. DIPETT's characteristics are shown in bold.

CRITERIA RELATING TO GRAMMATICAL RELATIONS

Constituency or dependency?

Is the goal of parsing (a) to segment the input string into constituents, or (b) to link heads to their dependents (arguments)? Parsers based on X-bar theory of course do a combination of the two. **DIPETT: constituency.**

Continuous or discontinuous phrases?

Does the parser require every constituent to be continuous? Some free-word-order parsers do not. (Applicable to dependency as well as constituency grammars: in dependency grammar, a constituent consists of any head plus all its dependents, their dependents, and so on recursively.) **DIPETT: continuous.**

Complete, partial, or no linear order?

Does the parser require fixed linear order of the elements of each constituent (or dependents of each head)? **DIPETT: complete order.**

CRITERIA RELATING TO SEQUENCE OF EXECUTION

Rule invocation strategy: top-down, bottom-up, or mixed?

Does the parser choose grammar rules on the basis of what it is looking for or on the basis of what it has found in the input? Left-corner parsing is an example of a mixed strategy (invoke a rule bottom-up and finish it top-down). **DIPETT: top-down.**

Completion: incremental or all-at-once?

Given a rule such as A $\rightarrow$ B C D, does the parser complete the A constituent in a single step after parsing B, C, and D, or does it add B, C, and D to the A one by one? (The analogous question can be asked for dependency grammar.) **DIPETT: all-at-once.**

Access to the input string: left-to-right sequential, random access, or what?

Most parsers accept words sequentially from left to right. **DIPETT: left-to-right.**

CRITERIA RELATING TO NONDETERMINISM

Handling of nondeterminacy: backtracking? look-ahead? concurrency?

For example, Prolog DCGs backtrack as needed; Marcus's parser uses look-ahead; and Earley's algorithm pursues all alternatives concurrently. **DIPETT: backtracking and a few look-ahead mechanisms.**

Limits on nondeterminacy, if any?

Marcus's parser has a strict limit on look-ahead; most other parsers have no limits on (simulated) nondeterminism. **DIPETT: mixed, with limits for some mechanisms.**

Mechanism to prevent duplication of work, if any?

The choices are: no chart; a passive chart that records only completed substructures; or an active chart that also records structures under construction. **DIPETT: passive chart.**

Other operations or tests during parsing?

Examples include feature unification or ATN register operations. **DIPETT: Prolog's term unification (it can be treated as simplified feature unification).**

Another technical element worth mentioning is the solution selection strategy. DIPETT produces what we call a "first good parse"—a syntactically acceptable, and usually semantically adequate, analysis. This is based on DIPETT's linguistic knowledge, look-ahead mechanisms and other heuristics. In a trade-off between efficiency and completeness, we chose to have one result that need not, inevitably, be the best parse possible. DIPETT's fundamental assumption of knowledge-scant analysis makes exhaustive search for an optimal parse entirely unrealistic.

Left recursion—a problem for top-down parser—is handled in DIPETT via two special parameters in a relevant subset of its DCG rules. The first of these parameters ensures that the calling rule (the parent predicate) only refers to itself if a token has been consumed in the input string. The second parameter counts the number of times a rule has been called in succession.

Yet another DIPETT's mechanism helps control potentially infinite searches. A CPU time limit, set at the beginning of a parsing session, ensures that no parsing run will ever take more than that predetermined limit; a partial parse may result if the limit has been reached even if a full parse tree could have been found given more time.

3.2 Linguistic Knowledge

The account of the English grammar in Quirk *et al.* avoids commitment to any formal theory of syntax, and builds on a tradition of broadly intuitive constituency analysis of the English sentence. DIPETT inherits this attitude. Just as their grammar is accessible to an educated non-linguist, our parser can be fairly easily understood and customized by programmers with a little knowledge of formal languages but almost no linguistic background. DIPETT does have a solid theoretical basis, but—in contrast with parsers built to give credence to some formal theory of syntax—this basis is not the *raison d'être* of our system. The complete DIPETT system contains several thousand lines of Prolog code and a few fragments in C. The core of its grammar—DCG rules and look-ahead mechanisms—resides in a well separated fragment of the program, so that linguistic maintenance is simplified. The DCG rules are well commented and contain detailed references to Quirk *et al*.

No clear, well-established methodology exists for determining the linguistic coverage of a parser. DIPETT's purpose was to cover a substantial domain-independent subset of English focussing on expository technical texts. Grammars presented in the application-oriented NLP literature had been too narrow or too *ad hoc*, or too domain-dependent. Designing wide syntactic coverage is a demanding enterprise. The first author had invested a year in devising the parser's core, relying on several general and NLP-oriented English grammars, especially Quirk *et al*., and on a few relatively large, syntactically varied, documents used for testing.

The reliance on Quirk *et al.* was an important design choice right from the earliest stages of DIPETT's development. Here is an example, derived from Quirk *et al.* (1973; 9.19):

> "There are three common correlative pairs: *either..or*, where *either* anticipates the alternative introduced by *or*; *both..and*, where *both* anticipates the addition introduced by *and*; and *neither..nor*, where *neither* negates the first clause and anticipates the additional negation introduced by *nor*."

Our example shows DIPETT's rule for the correlative conjunction of noun phrases using *both..and*. DIPETT's rule shows that for noun phrases, the conjunction *both* is followed by a noun phrase nucleus and its modifiers, which may be singular or plural. The conjunction *and* is anticipated as described above by Quirk *et al.*, as is the final noun phrase nucleus and modifiers. The *number* of the resulting noun phrase is calculated by the action *plur_rule_n*. For the conjunction *and*, as shown in the example below, the resulting conjuncted noun phrase is necessarily plural.

```
either_or__np(POS,conj_nps(both_and,NUM,[Np_e, Np_o]),NUM) -->
      conj(both), np_nucleus_and_mod(POS, Np_e, Num_e),
      conj(and), np_nucleus_and_mod(POS, Np_o, Num_o),
      {plur_rule_n(Num_e, Num_o, and, NUM)}.
```

The example sentences given by Quirk *et al.* (1973; 9.19) for noun phrases conjuncted by the correlative *both..and* are "*He smoked both cigars and cigarettes*." and "*Both Bob and Peter damaged the furniture.*" For the example noun phrase "*both cigars and cigarettes*" the data structure generated for the parse tree represents a noun phrase that consists of two separate noun phrases conjoined by the correlative *both_and*:

```
conj_nps(both_and, pl, [cigars, cigarettes])
```

where ***cigars*** and ***cigarettes*** are the parse trees for these (very simple) noun phrases. The whole parse tree for the sentence "*He smoked both cigars and cigarettes*." shows this single conjoined noun phrase as the object of the verb ***smoke***, as shown in Figure 1 (next page). DIPETT's parse tree output identifies the complete sentence structure, in this case a single declarative main clause with an SVO clause structure.

DIPETT builds parse trees that are highly detailed, and the constituents are all labelled with terms derived directly from Quirk et al.'s terminology. Parse trees show the subject, predicate, clause structure, complements and modifiers of each clause, as well as the head of each constituent. Also shown are the values of grammatical parameters, such as verb tense, modality and voice, and noun and pronoun person and number. DIPETT's parse trees display such linguistic phenomena as complex sentences, subordination, relative clauses, nominal clauses in a clear and easily readable manner.

Intuitiveness and, for what it is worth, "naturalness" of a syntactic description embodied in a parser is, in our opinion, advantageous from the software engineering point of view. A parser based on an esoteric syntactic theory, especially one with few very powerful rules and many locally enforced restrictions, makes it difficult to apply the principles of software development and maintenance.

A case in point is a recent evaluation and ensuing adaptation of DIPETT, which has been carried out as an exercise in software maintenance. A non-trivial test suite has been derived from Quirk *et al.*, which was treated as a high-level specification of the parser's behaviour. Another test suite has been acquired from the TSNLP project to provide an alternative set of test data.

```
parse_tree__decla_or_imper
  simple_sentence
    structure
      single_main_clause
        declarative
          statement
            subj
              entity
                ref
                  pers_pron  he
                number  sg3
            predicate
              regular
                verb  smoke
                  tense  past_simple
                  neg  yes
                  trans  tr_intr
              voice  active
              complement
                svo
                  conj_nps  both_and  pl
                    entity
                      head_noun
                        noun
                          n  cigar  countnoun
                      number  pl
                    entity
                      head_noun
                        noun
                          n  cigarette  countnoun
                      number  pl

end_of_input  period
```

Figure 1. DIPETT's parse tree for the sentence "*He smoked both cigars and cigarettes*."

3.3 Partial Parsing

DIPETT's robustness includes its capability of handling extra-grammatical inputs. While those are usually ungrammatical, data taken from free text may not always be covered by DIPETT's rather strict, though broad, grammar. DIPETT does not attempt parse fitting or any other kind of global repair. Instead, partial parsing (also called *fragmentary* parsing) is activated after full parsing has failed or timed out. In the partial-parsing mode, DIPETT uses a subset of its grammar rules, trying to recognize either a simple sentence or a sequence of major phrasal constituents: verb phrases, noun phrases, prepositional phrases, adjectival and adverbial phrases. The parser skips tokens for which it cannot find a grammatical role. Partial parsing identifies parsable fragments that, not surprisingly, play a significant role in the semantic analysis stages of the TANKA system. It is also interesting that DIPETT's partial parsing strategy occasionally boosts full parsing: a rule for simple sentences, out of reach during full parsing, may succeeds in this mode. The fewer fragments DIPETT produces, the more likely their usefulness. Figure 2 (next page) shows an example of a partial parse, given a low time limit.

```
parse_tree__frag_series

  sentence_fragment
    entity
      determinatives
        postdeter
          closed
            quant  most
      head_noun
        noun
          n  owner  countnoun
          noun_modifiers
            pre_modif
              n  car  countnoun
      number  pl

  sentence_fragment
    predicate
      regular
        verb  follow
          tense  imp
          neg  yes
          advs  carefully
            tokens  carefully
          trans  tr_intr
      voice  active

  sentence_fragment
    entity
      determinatives
        deter  the
        postdeter
          genitive  maker_s
      head_noun
        noun
          n  suggestion  countnoun
      number  pl
      np_postmodifiers
        pp  for
          entity
            determinatives
              deter  the
            attributes
              attrs
                adj  proper  pos
              tokens  proper
            head_noun
              noun
                n  care  countnoun
            number  sg3
            np_postmodifiers
              pp  of
                entity
                  determinatives
                    deter  their
                  head_noun
                    noun
                      n  car  countnoun
                  number  pl

  end_of_input  period
```

Figure 2. DIPETT's partial parse for the sentence "*Most car owners carefully follow the maker's suggestions for the proper care of their cars*." consists of 3 fragments.

4. Improvements and Evaluations

The goal of our diagnostic evaluation exercise (Scarlett, 2000; Scarlett & Szpakowicz, 2000) was to produce an *improved* DIPETT, that is, to extend DIPETT's coverage and improve even more the quality of the parse trees generated. The TSNLP group has provided an excellent test suite for such purposes, and certainly a broad-coverage parser should be able to parse all of the TSNLP test sentences. A diagnostic evaluation would be incomplete if the TSNLP data were not considered. The TSNLP suite may be useful for highlighting deficiencies in a grammar, but it lacks the benefits of parsing corpus data, and the coverage of the downloaded *tsdb(1)*[iii] database test items cannot be considered truly broad unless it were extended as envisaged by its design. To augment the coverage provided by the TSNLP suite, we proposed to design a collection of test sentences that was annotated and organized according to a formal English grammar description. Quirk *et al.*'s grammar was chosen as the source. The grammatical and starred example sentences were extracted and organized by topic, then used to test DIPETT's performance on the distribution of syntactic structures.

The Quirk test sentences were extracted from Quirk *et al*. (1985) chapters 2 and 3. The chapter 2 sentences cover a general outline of English grammar and of its major concepts and categories. The chapter 3 sentences cover the grammar of verb phrases. These sentences were augmented with selections from chapter 5 of Quirk *et al*. (1973) that illustrate the basic constituents of a noun phrase. DIPETT version 3 produced full parse trees for 79% of the grammatical Quirk sentences but only 65% of the grammatical TSNLP sentences. This result was unexpected because the TSNLP sentences are shorter, simpler, and narrower in their coverage of phenomena and phenomena interaction than are the Quirk sentences.

We modified DIPETT's grammar to optimize performance on the TSNLP suite. Modifications to DIPETT were made as unobtrusively as possible to minimize the risk of introducing errors and side effects. The following modifications were made to DIPETT's grammar to produce DIPETT version 4:

- Extended correlative conjunctions for noun phrases to include "*both..and*".
- Corrected parsing of passive sentences and allowed for *nil* passive subject.
- Added the progressive form of the auxiliary verb *be*.
- Completed the tense table (modal, aspect, and tense combinations).
- Added missing negative contractions.
- Fixed the parsing of "*used to*" in the passive, stative, and in aspect and tense combinations.
- Relative clauses, allowed comma delimiters, and relative clauses marked by a preposition followed by a wh-word.
- Added support for the mandative subjunctive.
- Allowed declarative statements terminated by a question mark (for the TSNLP S_Types-Questions-Y/N_questions-Non_inverted-Non-tagged test items).

[iii] The TSNLP database at http://tsnlp.dfki.uni-sb.de/tsnlp/tsdb/tsdb.cgi?language=english

We also tested how the improved version 4 of DIPETT performed in comparison with version 3. In particular we looked at the percentage of full parses (as opposed to partial or fragmentary parses). Table 1 presents the results.

ENGLISH TECHNICAL TEXT	% full parses, VERSION 3	% full parses, VERSION 4
QUIRK (578 grammatical sentences; rich in syntactic structure; phenomena interaction not controlled)	79.07%	85.12%
TSNLP (1143 grammatical sentences; basic sentence phenomena; controlled phenomena interaction)	65.18%	91.43%
CLOUDS (513 sentences; junior science reader on weather phenomena; simple syntax)	74.5%	71.2%
ENGINES (967 sentences; small engines mechanic manual; moderate to difficult syntax)	58.2%	58.7%
QUIZ (766 sentences; user guide on Cognos' PowerHouse Quiz report generator software; difficult syntax)	44.9%	42.4%
TAXES (304 sentences; tax declaration guide; difficult syntax)	45.1%	46.1%
"Home-made" Test Suite (150 sentences; wide variety of syntactic phenomena, from trivial to very difficult)	83.3%	82%

Table 1. Two versions of DIPETT compared.

These results confirm that we managed to improve the syntactic coverage—more syntactically accurate parse trees and fewer spurious trees—without losing any syntactic phenomena. This was achieved painlessly, following the approach outlined in this paper.

The percentage of full parses is still relatively low for the ENGINES text, and particularly for the QUIZ and TAXES texts. That is because these texts, syntactically quite involved, test DIPETT to the limit, especially in view of the fact that it is a surface parser with no semantic means of improving performance. For such texts, however, the percentage of partial parses

increases and makes it possible to recover a major part of the structure of syntactically difficult sentences. In general, DIPETT manages to parse at least 95% of any English text, just with varying proportion of full parses. For the last five texts from Table 1, partial parsing performed in addition to the full parses listed, gives the results shown in Table 2. That is, DIPETT parsed between 91% and 99% of these texts. Depending on syntactic complexity, unedited input text may get between 40% and 70% of full, usually semantically acceptable, parses.

ENGLISH TECHNICAL TEXT	% full parses	% partial parses	Total %
CLOUDS	71.2%	27.5%	98.7%
ENGINES	58.7%	38.6%	97.3%
QUIZ	42.4%	52.7%	95.1%
TAXES	46.1%	45.1%	91.2%
"Home-made" Test Suite	82%	17.3%	99.3%

Table 2. Full and partial parsing (VERSION 4).

5. Conclusions

Armed with our experience with a robust broad-coverage parser, we have argued for the importance of a clear, intuitive linguistic basis in parser engineering. We have explained why such linguistic "naturalness" is also beneficial to software engineering practice, which in turn plays a vital role in the development of robust natural language processing systems. We realize that quantitative support for our arguments would be hard to provide, but software engineering (and, specifically, parser engineering) is still not a completely objective business.

Parsing engineering has not lost importance with the advent of the Internet. On the contrary, there is a myriad of applications in which text plays a crucial role. Processors of natural languages have never been more needed. Text miners, key phrase extractor, translator and summarizers developed in the recent years are essentially based on shallow, text skimming strategies. The next generation of such tools should operate at a deeper level of linguistic processing. We are convinced that linguistic knowledge will be built into such tools with the invaluable help of parser engineering tools and methodologies.

Acknowledgments

This research is supported by the Natural Sciences and Engineering Research Council of Canada.

References

"Balancing" (1994). *Proceedings of the ACL Workshop "The Balancing Act" (Combining Symbolic and Statistical Approaches to Language)*, July 1994, Las Cruces, NM, USA.

BALDWIN, B., C. DORAN, J. REYNAR, B. SRINIVAS, M.NIV & M. WASSON (1997). EAGLE: An Extensible Architecture for General Linguistic Engineering. *Proceedings of the RIAO-97 Conference*, pp.271-283.

BALKAN, L., MEIJER, S., ARNOLD, D., DAUPHIN, E., ESTIVAL, D., FALKEDAL, K., LEHMANN, S., REGNIER-PROST, S. (1994a). Test Suite Design Guidelines and Methodology. Report to LRE 62-089 (D-WP2.1). University of Essex, UK.

BALKAN, L., MEIJER, S., ARNOLD, D., DAUPHIN, E., ESTIVAL, D., FALKEDAL, K., LEHMANN, S., NETTER, K., REGNIER-PROST, S. (1994b). Issues in Test Suite Design. Report to LRE 62-089 (D-WP2.1). University of Essex, UK.

BALKAN, L., NETTER, K., ARNOLD, D., MEIJER, S. (1994c). TSNLP: Test Suites for Natural Language Processing. *Proceedings of the Language Engineering Convention*, ELSNET, Centre for Cognitive Science, University of Edinburgh, pp. 17-22.

BARKER, K. (1998). Semi-Automatic Recognition of Semantic Relationships in English Technical Texts. PhD thesis, Department of Computer Science, University of Ottawa.

BARKER K., S. DELISLE & S. SZPAKOWICZ (1998). Test-Driving TANKA : Evaluating a Semi-automatic System of Text Analysis for Knowledge Acquisition. *Proceedings of the 12th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (CAI-1998)*, Lecture Notes in Artificial Intelligence #1418, Springer, pp.60-71.

BLACK, E., J. LAFFERTY & S. ROUKOS (1992). Development and Evaluation of a Broad-coverage Probabilistic Grammar of English-language Computer Manuals. *Proceedings of the 30th ACL Conference*, pp.185-192.

BOGURAEV, B., J. CARROLL, E. BRISCOE & C. GROVER (1988). Software Support for Practical Grammar Development. *Proceedings of the COLING-88 Conference*, pp.54-58.

BUCHANAN, B.G. & D.C. WILKINS (1993). *Readings in Knowledge Acquisition and Learning (Automating the Construction and Improvement of Expert Systems)*, Morgan Kaufmann Publishers.

CHARNIAK, E. (1997). Statistical Parsing with a Context-free Grammar and Word Statistics. *Proceedings of the AAAI-97 Conference*, pp.598-603.

COLLINS, M. (1997). Three Generative, Lexicalised Models for Statistical Parsing. *Proceedings of the EACL-97 Conference*, pp.16-23.

CUNNINGHAM, H., Y. WILKS & R.J. GAIZAUSKAS (1996). GATE — a General Architecture for Text Engineering. *Proceedings of the COLING-96 Conference*, pp.1057-1060.

DELISLE, S. (1994). Text Processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic Analysis for Incremental Knowledge Acquisition, Ph.D. thesis, Department of Computer Science, University of Ottawa.

DELISLE, S., K. BARKER, J.-F. DELANNOY, S. MATWIN & S. SZPAKOWICZ (1994). From Text to Horn Clauses: Combining Linguistic Analysis and Machine Learning, *Proceedings of the 10th Canadian Artificial Intelligence Conference — CAI-94*, pp.9-16.

ERBACH, G. (1992). A Tool for Grammar Engineering. *Proceedings of the 3rd ANLP Conference*, pp.243-244.

ESTIVAL, D., A. LAVELLI, K. NETTER & F. PIANESI, eds (1997). *Proceedings of the Computational Environments for Grammar Development and Linguistic Engineering Workshop of the ACL-EACL 1997 Conference*.

HATZIVASSILOGLOU, V. (1994). Do we Need Linguistics when we Have Statistics? (A Comparative Analysis of the Contributions of Linguistic Cues to a Statistical Word Grouping System. *Proceedings of the Combining Symbolic and Statistical Approaches to Language Workshop (the Balancing Act)*, pp.43-52.

LAWRENCE, S., S. FONG & C.L. GILES (1996). Natural Language Grammatical Inference: A Comparison of Recurrent Neural Networks and Machine Learning Methods. Symbolic, Connectionist, and Statistical Approaches to Learning for Natural Language Processing. *Lecture Notes in AI*, edited by S. Wermter, E. Riloff & G. Scheler, Springer Verlag, pp.33-47.

LEHMANN, S., OOPEN, S. REGNIER-PROST, S., NETTER, K., LUX, V., KLEIN, J., FALKEDAL, L., FOUVRY, F., ESTIVAL, D., DAUPHIN, E., COMPAGNION, H., BAUR, J., BALKAN, L. & ARNOLD, D. (1996). TSNLP – Test Suites for Natural Language Processing. *Proceedings of the COLING-96 Conference*, pp. 711-716.

LI, L., D.A. DAHL, L.M. NORTON, M.C. LINEBARGER & D. CHEN (1998). A Test Environment for Natural Language Understanding Systems. *Proceedings of the COLING-ACL'98 Conference*, pp.763-767.

MAGERMAN, D.M. (1994). Natural Language Parsing as Statistical Pattern Recognition, Ph.D. thesis, Stanford University.

NETTER, K. & F. PIANESI (1997). The preface to Estival *et al*. (1997), pp.iii-v.

QUIRK, R. & GREENBAUM, S. (1973). *A University Grammar of English*, Longman.

QUIRK, R., GREENBAUM, S., LEECH, G., & SVARTVIK, J. (1985). *A Comprehensive Grammar of the English Language*, Longman.

SCARLETT, E. & S. SZPAKOWICZ (2000). The Power of the TSNLP: Lessons from a Diagnostic Evaluation of a Broad-Coverage Parser. *Proceedings of the 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence (CAI-2000)*, Lecture Notes in Artificial Intelligence #1822, Springer, pp.138-150.

SCARLETT, E. (2000). An Evaluation of a Rule-Based Parser of English Sentences. Master's thesis, Department of Computer Science, University of Ottawa.

SCHMIDT, P., S. RIEDER, A. THEOFILIDIS & T. DECLERCK (1996). Lean Formalisms, Linguistic Theory, and Applications, Grammar Development in ALEP. *Proceedings of COLING-96 Conference*, pp.286-291.

SRINIVAS, B., C. DORAN & S. KULICK (1995). Heuristics and Parse Ranking. *Proceedings of the ACL/SIGPARSE 4th International Workshop on Parsing Technologies*.

VOLK, M. (1992). The Role of Testing in Grammar Engineering. *Proceedings of the 3rd ANLP Conference*, pp.257-258.

VOUTILAINEN, A. & L. PADRÓ. (1997). Developing a Hybrid NP Parser. *Proceedings of the 5th Conference on Applied Natural Language Processing*, pp.80-87.

WINOGRAD, T. (1983). *Language as a Cognitive Process* (Volume 1: Syntax), Addison-Wesley.

ZELLE J.M. & R.J. MOONEY (1994). Inducing Deterministic Prolog Parsers from Treebanks: A Machine Learning Approach. *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-1994)*, pp.748-753.