

Test-Driving TANKA: Evaluating a Semi-Automatic System of Text Analysis for Knowledge Acquisition

Ken Barker¹, Sylvain Delisle² and Stan Szpakowicz¹

¹School of Information Technology and Engineering
University of Ottawa
Ottawa, Canada K1N 6N5
{kbarker, szpak}@site.uottawa.ca

²Département de mathématiques et d'informatique
Université du Québec à Trois-Rivières
Trois-Rivières, Canada G9A 5H7
Sylvain_Delisle@uqtr.quebec.ca

Abstract. The evaluation of a large implemented natural language processing system involves more than its application to a common performance task. Such tasks have been used in the message understanding conferences (MUCs), text retrieval conferences (TRECs) as well as in speech technology and machine translation workshops. It is useful to compare the performance of different systems in a predefined application, but a detailed evaluation must take into account the specificity of the system.

We have carried out a systematic performance evaluation of our text analysis system TANKA. Since it is a semi-automatic, trainable system, we had to measure the user's participation (with a view to decreasing it gradually) and the rate at which the system learns from preceding analyses. This paper discusses the premises, the design and the execution of an evaluation of TANKA. The results confirm the basic assumptions of our supervised text analysis procedures, namely, that the system learns to make better analyses, that knowledge acquisition is possible even from erroneous or fragmentary parses and that the process is not too onerous for the user.

1 Introduction

This paper describes the evaluation of the trainable semi-automatic text analysis system TANKA (Text ANALYSIS for Knowledge Acquisition). TANKA aims to build interactively a semantic representation of a technical text. Throughout the development of TANKA we have performed diagnostic evaluations of the system's components. Now that these components are complete, the whole system must be tested on new texts in different domains.

There has been increased interest in the evaluation of natural language processing tools over the last few years. The message understanding conferences (MUCs) are a direct reaction to a lack of standardization in the evaluation of such tools. The MUC competitions compare the performance of different systems on some uniform, predetermined text processing task. The motivation for and background of the MUC competitions are described by Grishman & Sundheim [13].

Although these competitions have successfully emphasized the importance of evaluation for the NLP community, they have also revealed certain limitations. In

particular, researchers have noted that the predefined tasks in evaluation competitions result in applications that are designed to score well, but are not portable [16]. Furthermore, little attention has been paid to the evaluation of interactive systems and the role of users [14].

To address these issues, Sparck Jones and Galliers ([18], [19]) offer more general strategies for evaluating generic NLP systems. They note that “there is far too much variety in the situations and subjects of evaluation to come up with a definite [evaluation] scenario” [19: 193].

Hirschman & Thompson [14] note that different kinds of evaluation make different assumptions about the distribution of test data. While it is important to test a system on all types of input it is designed to handle, such a *diagnostic evaluation* does not reflect the performance of the system on actual texts. The distribution of linguistic phenomena in the test suite of a diagnostic evaluation is almost certainly not the same as the distribution of phenomena in complete real texts, which form the test suite for a *performance evaluation*. For performance evaluations, it is important to identify *criteria* (what is being evaluated), *measures* (what properties of performance reflect the criteria) and *methods* (how the measures are analyzed to arrive at an evaluation of the criteria).

The evaluation methodology described in this paper is necessarily specific to TANKA. Nonetheless, we have attempted to adhere to the precepts of a more rigorous approach to NLP system evaluation.

2 The TANKA System

Analysis in TANKA consists of recognizing semantic relationships signalled by surface linguistic phenomena. The system uses as little *a priori* semantic knowledge as possible. Instead, it performs detailed syntactic analysis using publicly available part-of-speech lists and lexicons and produces a tentative semantic analysis. This analysis is proposed to a participating user who usually approves the system’s proposal. In the case of an incorrect or incomplete analysis, the user may also be required to supply elements of the semantic interpretation. Delisle *et al.* [12] offer a detailed description of TANKA.

2.1 The DIPETT Parser

Syntactic analysis in TANKA is performed by DIPETT (Domain Independent Parser of English Technical Texts), a broad coverage Definite Clause Grammar parser whose rules are based primarily on Quirk *et al.* [17].

DIPETT takes an unedited, untagged text and automatically produces a single initial parse of each sentence. This first good parse tree is a detailed representation of the constituent structure of a sentence. If DIPETT is unable to produce a single complete parse of a sentence within a time limit imposed by the user, it will attempt to produce parses for fragments within the sentence (such as if possible, clauses and phrases). Delisle [10] and Delisle & Szpakowicz [11] present a complete discussion of DIPETT and related parsing issues. A WWW version of the parser with a simplified interface will be released soon.

2.2 The HAIKU Semantic Analyzer

Semantic Relationships. Given the parse trees produced by DIPETT, the HAIKU semantic analyzer [12] identifies the semantic relationships expressed by related syntactic constituents. The semantic relationships are expressed at three levels: between connected clauses, within clauses (between a verb and each of its arguments) and within noun phrases (between a head and each of its modifiers). The semantic relationships that HAIKU assigns to syntactic structures at each of these levels appear in Table 1. The *clause level relationships (CLRs)* are assigned to connected clauses, the *cases* are assigned to verb-argument pairs and the *noun modifier relationships (NMRs)* are assigned to modifier-noun pairs.

There are several observations to make about these semantic relationships. The lists are amalgams of similar lists used by researchers in discourse analysis, case and valency theory and noun phrase analysis. For each of our three lists, we identified an initial set of relationships and then did an extensive survey of the lexical items that mark them. This survey identified several omissions and redundancies in the lists. We further validated the relationships by checking their coverage on real English texts. Details of the construction process and validation appear in our publications [7, 2], as well as comparisons of our relationships to other lists [6, 3].

The next observation is that HAIKU does not depend on these particular lists of relationships. The techniques it uses at each level of analysis would work with any other closed list of semantic relationships.

Finally, several relationships appear in more than one list, reflecting the fact that the same semantic relationship may be realized at different levels of syntax. Merging the relationships into a single unified list for all three levels is a consideration for future work.

Semantic Analysis. HAIKU tries to assign semantic relationships with a minimum of *a priori* hand coded semantic knowledge. In the absence of such precoded semantics HAIKU enlists the help of a cooperating user who oversees decisions during semantic

<i>CLRs</i>	<i>Cases</i>		<i>NMRs</i>	
Causation	Accompaniment	Location_to	Agent	Object
Conjunction	Agent	Manner	Beneficiary	Possessor
Cooccurrence	Beneficiary	Material	Cause	Product
Detraction	Cause	Measure	Container	Property
Disjunction	Content	Object	Content	Purpose
Enablement	Direction	Opposition	Destination	Result
Entailment	Effect	Order	Equative	Source
Precedence	Exclusion	Orientation	Instrument	State
Prevention	Experiencer	Purpose	Located	Time
	Frequency	Recipient	Location	Topic
	Instrument	Time_at	Material	
	Location_at	Time_from		
	Location_from	Time_through		
	Location_through	Time_to		

Table 1. Semantic relationships in HAIKU

analysis. To lessen the burden on the user, the system first attempts automatic analysis. It compares input structures to similar structures in the text for which semantic analyses have been stored. Since it does not have access to a large body of pre-analyzed text, HAIKU starts processing from scratch for a text (or a collection of texts) and acquires the needed data incrementally.¹

Clause level relationships are assigned whenever there are two or more connected finite clauses in a sentence. For each clause, DIPETT provides a complete syntactic analysis including tense, modality and polarity (positive/negative). It gives us the connective (usually a conjunction) and the type of syntactic relationship between the clauses: coordinate, subordinate or correlative.

The CLR analyzer looks up the connective in a dictionary that maps each connective to the CLRs that it might mark. Since the connectives are a small closed class, the construction of such a marker dictionary is not a large knowledge engineering task. Once constructed, it can be used for any text. Using the subset of CLRs, HAIKU holds competitions between each pair of relationships based on the syntactic features of the clauses. The CLR with the most points after all competitions is the one suggested to the user for approval. The CLRs and the CLR analyzer are described by Barker & Szpakowicz [6].

Within a clause, the parser identifies the main verb and its arguments: subject, direct and indirect objects, adverbials and prepositional phrases. From this information, the case analyzer builds a *case marker pattern* (CMP) made of the symbols `psubj`, `pobj`, `piobj`, `adv` and any prepositions attached to the verb. To assign cases to the arguments of a given verb, the system compares the given verb+CMP to other verb+CMP instances already analyzed. It chooses the most similar previous verb+CMP instances and suggests previously assigned cases for this verb+CMP. Delisle *et al.* [12] and Barker *et al.* [7] describe case analysis and the cases in detail.

Within noun phrases, the parser identifies a flat list of premodifiers and any postmodifying prepositional phrases and appositives. The NMR analyzer first brackets the flat list of premodifiers into modifier-modificand pairs [4], and then assigns NMRs to each pair. NMRs are also assigned to the relationships between the head noun of the noun phrase and each postmodifying phrase. To pick the best NMR, the system first finds the most similar modifier-modificand instances previously analyzed. Next, it finds the NMRs previously assigned to the most similar instances and selects one of these relationships to present to the user for approval.

3 The Evaluation Exercise

TANKA is intended to analyze an unedited English technical text. Technical texts usually lack humour, intentional ambiguity, and other devices that would make

¹ An alternative to starting analysis from scratch would be to accumulate the semantic analyses from session to session. The extent to which the acquired knowledge from one text (or domain) would be useful in the analysis of a different text is a consideration for future work.

analysis more difficult. Copeck *et al.* [9] describe a study to investigate the character of texts typically considered technical. So in order to evaluate TANKA we ran it on a real, unedited technical text.

Individual components of DIPETT and HAIKU have been tested on a variety of technical texts including an income tax guide, a fourth generation programming language manual, a building code regulatory text and a computer installation guide.

The most thorough previous test of the system was reported by Barker & Delisle [5]. For that experiment, we chose a text [15] with fairly simple syntax to ensure a high number of successful parses. That text describes a technical domain (weather phenomena) using simple language. We refer to this book as the *clouds* text and the experiment as the *clouds* experiment. We chose a simpler text in the belief that the higher parse success rate would allow the system to acquire more knowledge (since semantic analysis is based on the results of syntactic analysis). What we found instead was that the simpler text used very general terminology and carried less content. We also noted that even if a sentence was parsed incorrectly or in fragments, much of the knowledge in the sentence could still be acquired, either from the incorrect or incomplete parse tree, or from other sentences containing the same or similar concepts (we refer to this as *conceptual redundancy*).

For the evaluation described in this paper, we chose a more syntactically complex text [1]. The text has more complex constructions, but it also uses more specific terminology and contains more information and less “fluff”. We refer to this book as the *small engines* text and the experiment as the *small engines* experiment.

3.1 The Evaluation Methodology

The *small engines* experiment was held at l’Université du Québec à Trois-Rivières over five days. Just over 15 hours were spent during timed semantic analysis interaction. One of the authors drove the system while another timed the interaction on each sentence and recorded details of the interaction. All interaction decisions were made collectively, adding time to the interactions for discussions, but hopefully also eliminating personal biases. The system was run in Quintus Prolog 3.2 on a Sparc-20 with a 120 second CPU time limit for parsing per sentence.

The first step in running DIPETT and HAIKU on a text is to build a lexicon for the text. DIPETT has a built-in part of speech labeller. The labeller uses the Collins wordlist to create a lexicon containing all parts of speech of all the words in the text. The resulting lexicon contains extraneous entries, since not every part of speech of every word appears in the text. We used a concordance tool to check the lexicon for extraneous entries. We should note that the lexicon construction stage is not as much work as tagging the text with part of speech information; nor does it give parsing as big a headstart. Many words appear more than once as different parts of speech in the text. The lexicon may contain several entries for each word, even after deleting the extraneous entries. The construction of the lexicon required a short day’s work for 1200 words.

For the analysis of each sentence, we started a stopwatch as soon as parsing was complete. We then examined the parse and recorded details of parse completeness and correctness. Interaction then proceeded through clause level relationship analysis, case analysis and noun modifier relationship analysis. For each of these levels of

analysis, we recorded the degree of difficulty of the interaction. After all interactions in semantic analysis were complete, we stopped the stopwatch and recorded the total interaction time for the sentence.

The degree of difficulty of interaction (what we refer to as *onus*) was recorded as an integer from 0 to 3. 0 means that the interaction is trivial. 1 is assigned to an interaction that requires a few moments of reflection. 2 rates an interaction as requiring serious thought, but eventually a semantic relationship is assigned. 3 means that even after much contemplation no relationship is deemed appropriate for the given input.

3.2 Examples of Interaction

To understand the criteria of our evaluation better, it is useful to show the reproductions of the different kinds of interactions with HAIKU. The user's input is shown bold and underlined

Figure 1 shows the interaction for the CLR assignment for sentence 119 from the *small engines* text. The sentence has two finite clauses connected by the subordinator *as*. From the parse tree, the system identifies the connective, the main clause and the subordinate clause. It looks up *as* in the marker dictionary and finds that it may mark Cooccurrence (ctmp), Causation (caus), Enablement (enab), Entailment (entl) or Prevention (prev). The system holds competitions between each pair of relationships (ten competitions for this example) based on the syntactic features of the clauses. The CLR analyzer chooses Causation as the most appropriate for the sentence, since it finishes with the most points.

Figure 2 shows the case interaction for sentence 164 from the *small engines* text.

String #119: **As the crankshaft rotates, its gear turns the gear on the camshaft and causes it to rotate, too.**

```

There is a Clause-Level Relationship marked by 'as':

  its gear turns the gear on the camshaft and causes it to
  rotate, too
    as
  the crankshaft rotates

CLR competitions among [ctmp,caus,enab,entl,prev]
Results (maximum is 8):
  ctmp   caus   enab   entl   prev
+-----+-----+-----+-----+
   2     8     1     6     3
The CLR Analyzer's best suggestion(s) for this input:
(1) Causation (caus)
> Please enter a number between 1 and 1: 1

Your CLR assignment will be stored as:
  the crankshaft rotates
    <causes>
  its gear turns the gear on the camshaft and causes it to
  rotate , too

```

Figure 1. Clause level relationship analysis interaction for sentence #119

String #164: *Electric starter motors **crank** an engine by spinning the flywheel.*

```

CURRENT SUBJ : electric starter motors
CURRENT VERB : crank
CURRENT COMPL : an engine by spinning the flywheel

CMP found by HAIKU: psubj-pobj-by
> Do you wish to overrule this CMP? N

S4: new verb with known CMP; new CP (relative to this verb).
Candidate CPs with example sentences:
(1) agt-obj-inst the carburetor controls the crankshaft speed
    by the amount of fuel/air mixture it allows
    into the cylinder.
(2) agt-obj-manr they do this by seeing that the oil is
    changed at the proper time and that greasing
    and tune-ups are carried out regularly.

if appropriate, enter a number between 1 and 2: 1
CMP & CP will be paired as: psubj/agt pobj/obj by/inst

```

Figure 2. Case analysis interaction for sentence #164

This sentence has one clause with the main verb *crank*. The case analyzer constructs the case marker pattern *psubj-pobj-by* based on information in the parse tree. In the previous sentences, the verb *crank* has never occurred. However, the system has seen the CMP twice: once with the verb *control* and once with the verb *do*. In those two instances the corresponding case patterns were Agent-Object-Instrument and Agent-Object-Manner. The system suggests these two case patterns to the user, who chooses the first, since “spinning the flywheel” is the Instrument (or *method* used) to “crank an engine”.

For the noun phrase *the threaded part of the plug* in sentence 142, the NMR analyzer will assign two NMRs: one between *threaded* and *part* and one between *part* and *plug*. The interaction in Figure 3 shows the interaction for *threaded part* only. The system has never seen *threaded* as a modifier of *part*, but it has seen *threaded* as a modifier of other modificands and *part* as a modificand with other modifiers. NMRA calculates a score for each of the NMRs that have been previously assigned when *threaded* was a modifier and when *part* was a modificand. Property receives the highest score for this pair.

String #142: *The lower electrode is fastened to the **threaded part** of the plug.*

```

HAIKU: Noun-Modifier Relationship Analysis of current input ...

Match type 3: (threaded, _, nil) or (_, part, nil)
[prop]: 4.66667

For the phrase 'threaded part'
> Do you accept the assignment:
  Property (prop): threaded_part is threaded

[n/a/<nmr>/Y] Y

```

Figure 3. Noun modifier relationship analysis interaction for sentence #142

4 The Evaluation

Three main criteria guided the evaluation of our system. First, we wanted to evaluate the ability of HAIKU to learn to make better suggestions to the user. Since the system starts with no prior analyses on which to base its suggestions, the user is responsible for supplying the semantic relationships at the beginning of a session. To measure HAIKU's learning, we compared the cumulative number of assignments required from the user to the cumulative number of correct assignments suggested to the user by the system over the course of analyzing the *small engines* text.

Second, we wanted to compare the number of relationships that HAIKU analyzed with the total number of such relationships in a text (*i.e.*, the number it *should have* analyzed). Since it would not be feasible to count by hand the total number of relationships even in a text of several hundred sentences, we sampled relationships at random from the text to find the proportion that HAIKU analyzed. This proportion can be thought of as a measure of *Recall*² of the kind of knowledge HAIKU has been designed to acquire.

Third, we wanted to evaluate the burden that semi-automatic analysis places on the user. Since burden is a fairly subjective criterion, we looked at the amount of time spent by the user on each sentence as well as the onus rating described in section 3.1.

4.1 Evaluating the Parser

In order to evaluate the performance of HAIKU, it is important to look at the results of parsing, the input to HAIKU.

Of the 557 sentences that we analyzed, 55% received a complete parse. That does not necessarily mean that the parses were correct. In fact, only 31% of the 557 parses were perfect. Another 9% were good enough not to affect semantic analysis. That means that for 60% of the sentences, parse errors were serious enough to affect semantic analysis. Most often, these errors meant that HAIKU was unable to make a semantic relationship assignment for a given input. Fortunately, due to the *conceptual redundancy* mentioned in section 3, much of the knowledge in these difficult sentences was acquired by the system anyway.

4.2 Evaluating CLR Analysis

The HAIKU module most affected by parse errors is the clause level relationship analyzer. In order to make an assignment, CLRA needs a complete and correct parse at the top-most level. A second problem with CLR analysis is that CLR data is the sparse: every sentence usually has several noun phrases and at least one finite clause, but relatively few have multiple connected finite clauses. Note however that because

² All semantic relationship assignments are either accepted by the user if correct or supplied by the user if the system's assignments are incorrect. Therefore, *Recall's* partner *Precision* is a meaningless measure in HAIKU (*i.e.*, it is always 100%).

the CLR data is so sparse, the proportion of relationships analyzed by HAIKU can be measured directly without sampling.

In the *clouds* experiment, the system correctly determined 73% of the CLRs assigned, while the user assigned the other 27%. In the *small engines* experiment, the system determined 76% of CLRs automatically. For the *clouds* text, there were 76 connected clauses requiring CLR assignments, 67% of which received analysis. For the *small engines* text, however, only 28% of the clause pairs received analysis. The small number of clause level relationships actually captured by HAIKU for the *small engines* text is a direct result of the error rate in complete parses of structurally complex sentences.

4.3 Evaluating Case Analysis

For case analysis we first compare the number of case patterns that the system determined correctly with the number the user had to supply directly. 584 clauses in the *small engines* text received a case analysis. The system is considered to have determined the case pattern correctly if the correct case pattern was among those suggested to the user. The system made an average 4.4 suggestions for clauses for which it could make any suggestions at all.

Figure 4 plots the cumulative number of assignments made by the user against the cumulative number of correct assignments among the system's suggestions. After fewer than a hundred clauses, the number of correct system assignments overtook the user assignments for good. By the end of the experiment, the system had suggested a correct case pattern for 66% of the assignments.

In order to evaluate the proportion of clauses in the text that received a case analysis, we took 100 verbs at random from the text, determined their case marker pattern manually and found that for 97 of them there was a corresponding case pattern in HAIKU's output. Therefore, with 95% confidence we claim that HAIKU extracted between 91.5% and 98.9% of the case patterns in the text. This high coverage is due

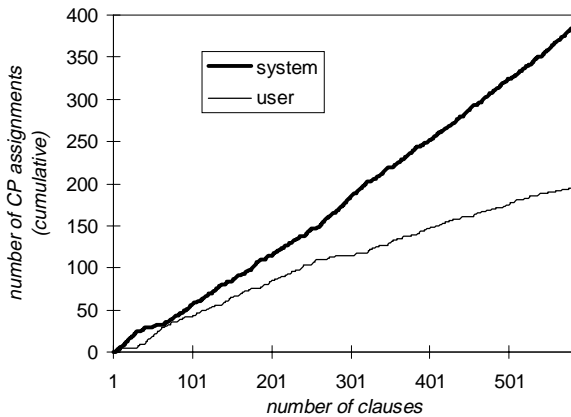


Figure 4. Case pattern assignments

to the fact that the user is given the opportunity to correct case marker patterns that are incorrect because of misparses.

4.4 Evaluating NMR Analysis

In the *small engines* experiment, 886 modifier-noun pairs were assigned an NMR. As with case analysis, we consider the system's assignment correct when the correct label is among its suggestions to the user. The system made an average 1.1 suggestions (when it could make any suggestions at all).

Figure 5 shows the cumulative number of NMR assignments supplied by the user versus those determined correctly by the system. Again, after about 100 assignments, the system was able to make the majority of assignments automatically. By the end of the experiment, the system had correctly assigned 69% of the NMRs.

To evaluate the number of modifier-noun relationships that HAIKU captured, we took 100 modifier-noun pairs at random from the text and found that 87 of them appeared in HAIKU's output. At the 95% confidence level, we can say that HAIKU extracted between 79.0% and 92.2% of the modifier-noun relationships in the *small engines* text.

4.5 User Burden

We said in section 3 that the *clouds* text was chosen for its relatively simple syntax to ensure a large number of successful parses. Correct parses guarantee the maximum amount of semantic analysis (and therefore user interaction). Making the user as active as possible would allow us to observe trends in user time and burden over the course of a session with TANKA. The main trends observed were that user time was directly related to sentence length and complexity, and that the average user time (adjusted for sentence length) decreased over the course of analyzing the text. As it turns out, we observed the same trends in the *small engines* experiment.

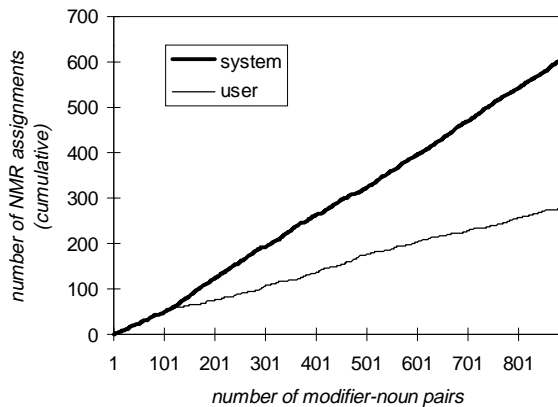


Figure 5. Noun modifier relationship assignments

The average number of tokens (words or punctuation) in the *small engines* sentences was 15.4. The average number of CLRs per sentence was 0.05, while the average number of case patterns was 1.2 and the average number of NMRs was 1.7. On average, we spent 1 minute, 49 seconds on each sentence. By coincidence, the average user time for the *clouds* experiment was also 1 minute, 49 seconds for an average of 0.1 CLRs and 0.9 case patterns per sentence (NMRs were not evaluated in the *clouds* experiment).

The *onus* numbers for each level of semantic analysis in the *small engines* experiment appear in Table 2.

<i>onus</i>	<i>0</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>average</i>
CLRA	18	4	0	0	0.19
CA	480	89	15	0	0.20
NMRA	808	71	7	0	0.10

Table 2. Onus ratings for semantic analysis

5 Conclusions

Three main conclusions can be drawn from the evaluation, corresponding to the three criteria presented in section 4. The first is that the system learns. By using partial matching on a growing set of semantic patterns, the system can come up with analyses of inputs that it has never seen before.

The second conclusion is that the system can acquire knowledge from the text even with fragmentary parses and misparses, although case analysis and NMR analysis are less sensitive to parse errors than CLR analysis. As mentioned earlier, this result is due to two things. First, many of the key concepts in a text are repeated frequently. Second, the fragments in fragmentary parses are often large enough to cover one or more phrases or even whole clauses. This conclusion is significant, since one of the main problems of systems that rely on detailed parsing is the parse error rate, due to the brittleness of grammars. Our evaluation shows that the amount of knowledge acquired is not necessarily limited by the proportion of perfect parses.

Finally, the evaluation showed that our system is not too onerous for the user. We extracted fairly complete knowledge from more than 500 sentences in a technical domain in just a few days. We also showed in section 4 that after a brief training period the system made the majority of semantic relationship assignments, with the user merely accepting the analyses. Finally, we observed that the average user time decreased over the course of the experiment.

Acknowledgments

This research is supported by the Natural Sciences and Engineering Research Council of Canada and by le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche.

References

1. Atkinson, Henry F. (1990). *Mechanics of Small Engines*. New York: Gregg Division, McGraw-Hill.
2. Barker, Ken (1996). "The Assessment of Semantic Cases Using English Positional, Prepositional and Adverbial Case Markers." TR-96-08, Department of Computer Science, University of Ottawa.
3. Barker, Ken (1997). "Noun Modifier Relationship Analysis in the TANKA System." TR-97-02, Department of Computer Science, University of Ottawa.
4. Barker, Ken (1998). "A Trainable Bracketeer for Noun Modifiers." *Proceedings of the Twelfth Canadian Conference on Artificial Intelligence*, Vancouver.
5. Barker, Ken & Sylvain Delisle (1996). "Experimental Validation of a Semi-Automatic Text Analyzer." TR-96-01, Department of Computer Science, University of Ottawa.
6. Barker, Ken & Stan Szpakowicz (1995). "Interactive Semantic analysis of Clause-Level Relationships." *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, Brisbane, 22-30.
7. Barker, Ken, Terry Copeck, Sylvain Delisle & Stan Szpakowicz (1997). "Systematic Construction of a Versatile Case System." *Journal of Natural Language Engineering* (in print).
8. Cole, Ronald A., Joseph Mariani, Hans Uszkoreit, Annie Zaenen & Victor Zue (1996). *Survey of the State of the Art in Human Language Technology*.
<http://www.cse.ogi.edu/CSLU/HLTSurvey/>
9. Copeck, Terry, Ken Barker, Sylvain Delisle, Stan Szpakowicz & Jean-François Delannoy (1997). "What is Technical Text?" *Language Sciences* 19(4), 391-424.
10. Delisle, Sylvain (1994). "Text processing without A-Priori Domain Knowledge: Semi-Automatic Linguistic analysis for Incremental Knowledge Acquisition." Ph.D. thesis, TR-94-02, Department of Computer Science, University of Ottawa.
11. Delisle, Sylvain & Stan Szpakowicz (1995). "Realistic Parsing: Practical Solutions of Difficult Problems." *Proceedings of the Second Conference of the Pacific Association for Computational Linguistics*, Brisbane, 59-68.
12. Delisle, Sylvain, Ken Barker, Terry Copeck & Stan Szpakowicz (1996). "Interactive Semantic analysis of Technical Texts." *Computational Intelligence* 12(2), May, 1996, 273-306.
13. Grishman R & B. Sundheim (1996). "Message Understanding Conference - 6: A Brief History." *Proceedings of COLING-96*, 466-471.
14. Hirschman, Lynette & Henry S. Thompson (1996). "Overview of Evaluation in Speech and Natural Language Processing." in [8].
15. Larrick, Nancy. (1961). *Junior Science Book of Rain, Hail, Sleet & Snow*. Champaign: Garrard Publishing Company.
16. MUC-6 (1996). *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann.
17. Quirk, Randolph, Sidney Greenbaum, Geoffrey Leech & Jan Svartvik (1985). *A Comprehensive Grammar of the English Language*. London: Longman.
18. Sparck Jones, Karen (1994). "Towards Better NLP System Evaluation." *Proceedings of the Human Language Technology Workshop, 1994*, San Francisco: Morgan Kaufmann, 102-107.
19. Sparck Jones, Karen & Julia R. Galliers (1996). "Evaluating Natural Language Processing Systems: An Analysis and Review." *Lecture Notes in Artificial Intelligence* 1083, New York: Springer-Verlag.