Evaluation of TTP Parser: A Preliminary Report

Tomek Strzalkowski* and Peter G. N. Scheyen‡

* Courant Institute of Mathematical Sciences, New York University
715 Broadway, rm 704, New York, NY 10003
email: tomek@cs.nyu.edu

‡ Department of Computer Science, The University of Western Ontario
London, Ontario N6A 5B7
email: scheyen@csd.uwo.ca

Abstract

TTP (Tagged Text Parser) is a fast and robust natural language parser specifically designed to process vast quantities of unrestricted text. TTP can analyze written text at the speed of approximately 0.3 sec/sentence, or 73 words per second. An important novel feature of TTP parser is that it is equipped with a skip-and-fit recovery mechanism that allows for fast closing of more difficult sub-constituents after a preset amount of time has elapsed without producing a parse. Although a complete analysis is attempted for each sentence, the parser may occasionally ignore fragments of input to resume "normal" processing after skipping a few words. These fragments are later analyzed separately and attached as incomplete constituents to the main parse tree. TTP has recently been evaluated against several leading parsers. While no formal numbers were released (a formal evaluation is planned later this year), TTP has performed surprisingly well. The main argument of this paper is that TTP can provide a substantial gain in parsing speed giving up relatively little in terms of the quality of output it produces. This property allows TTP to be used effectively in parsing large volumes of text.

1 Overview of This Paper

Recently, there has been a growing demand for fast and reliable natural language processing tools, capable of performing reasonably accurate syntactic analysis of large volumes of text within an acceptable time. A full sentential parser that produces complete analysis of input, may be considered reasonably fast if the average parsing time per sentence falls anywhere between 2 and 10 seconds. A large volume of text, perhaps a gigabyte or more, would contain as many as 7 million sentences. At the speed of say, 6 sec/sentence, this much text would require well over a year to parse. While 7 million sentences is a lot of text, this much may easily be contained in a fair-sized text database. Therefore, the parsing speed would have to be increased by at least a factor of 10 to make such a task manageable.

In this paper we describe TTP, a fast and robust natural language parser that can analyze written text and generate regularized parse structures at a speed of below 1 second per sentence. In the experiments conducted on variety of natural language texts, including technical prose, news messages, and newspaper articles, the average parsing time varied between 0.3 sec/sentence and 0.5 sec/sentence, or between 2500 and 4300 words per minute, as we tried to find an acceptable compromise between parser's speed and precision (these results were obtained on a Sun SparcStation 2). Original experiments were performed within an information retrieval system with the recall/precision statistics used to measure effectiveness of the parser.

In the second part of the paper, the linguistic accuracy of TTP is discussed based on the partial results of a quantitative evaluation of its output using the Parseval method (Black et al, 1991). This method calculates three scores of "closeness" as it compares the bracketed parse structures returned by the parser against a pre-selected standard. These scores are: the crossings rate which indicates how many constituents in the candidate parse are incompatible with those in the standard; recall which is the percentage of candidate constituents found in the standard; and precision which specifies the percentage of standard constituents in the candidate parse. Parseval may also be used to compare the performance of different parsers. In comparison with NYU's Proteus parser, for example, which is on average two levels of magnitude slower than TTP, the crossing score was only 6 to 27% higher for TTP, with recall 13% lower, and approximately the same precision for both parsers.

In addition we discuss the relationships between allotted parse time per sentence, the average parsing time, crossings rate, and recall and precision scores.

2 Introduction to Tagged Text Parser

It has long been assumed that in order to gain speed, one may have to trade in some of the parser's accuracy. For example, we may have to settle for partial parsing that would recognize only selected grammatical structures (e.g. noun phrases; Ruge et al., 1991), or would avoid making difficult decisions (e.g. pp-attachment; Hindle, 1983). Much of the overhead and inefficiency comes from the fact that the lexical and structural ambiguity of natural language input can only be dealt with using limited context information available to the parser. Partial parsing techniques have been used with a considerable success in processing large volumes of text, for example AT&T's Fidditch (Hindle and Rooth, 1991) parsed 13 million words of Associated Press news messages, while MIT's parser (de Marcken, 1990) was used to process the 1 million word Lancaster/Oslo/Bergen (LOB) corpus. In both cases, the parsers were designed to do partial processing only, that is, they would never attempt a complete analysis of certain constructions, such as the attachment of pp-adjuncts, subordinate clauses, or coordinations. This kind of partial analysis may be sufficient in some applications because of a relatively high precision of identifying correct syntactic dependencies, for example Church and Hanks (1990) used partial parses generated by Fidditch to study word co-occurrence patterns in syntactic contexts. On the other hand, applications involving information extraction or retrieval from text will usually require more accurate parsers.

An alternative is to create a parser that would attempt to produce a complete parse, and would resort to partial or approximate analysis only under exceptional conditions such as an extra-grammatical input or a severe time pressure. Encountering a construction that it couldn't handle, the parser would first try to produce an approximate analysis of the difficult fragment, and then resume normal processing for the rest of the input. The outcome is a kind of "fitted" parse, reflecting a compromise between the actual input and grammar-encoded preferences. One way to accomplish this is to adopt the following procedure: (1) close (reduce) the obstructing constituent (one which is being currently parsed), then possibly reduce a few of its parent constituents, removing corresponding productions from further consideration, until a production is reactivated for which a continuation is possible; (2) Jump over the intervening material so as to restart processing of the remainder of the sentence using the newly reactivated production.

As an example, consider the following sentence where the highlighted fragment is likely to cause problems, and may be better off ignored in the first pass:

> "The method is illustrated by the automatic construction of both recursive and iterative programs operating on natural numbers, lists, and trees, in order to construct a program satisfying certain specifications *a theorem induced by those specifications is proved,* and the desired program is extracted from the proof."

Assuming that the parser now reads the article 'a' following the string 'certain specifications', it may proceed to reduce the current NP, then SI $\rightarrow$ to"+V+NP, SI $\rightarrow$ SA, SA $\rightarrow$ NP+V+NP+SA, until finally S → S+"and"+S is reached on the stack. Subsequently, the parser skips input to find 'and', then resumes normal processing. The key point here is that TTP decides (or is forced) to reduce incomplete constituents rather than to backtrack or otherwise select an alternative analysis. However, this is done only after the parser is thrown into the panic mode, which in case of TTP is induced by the time-out signal. In other words, while there is still time TTP will proceed in regular top-down fashion until the time-out signal is received. Afterwards, for some productions early reduction will be forced and fragments of input will be skipped if necessary. If this action does not produce a parse within a preset time (which is usually much longer than the original 'regular' parsing time), the second time-out signal is generated which forces the parser to finish even at the cost of introducing dummy constituents into the parse tree. The skipped-over fragments of input are quickly processed by a simple phrasal analyzer, and then attached to the main parse tree at the points where they were deleted.

An example parse structure returned by TTP is shown below. Note (vrbtm X) brackets which surround all un-parsed tokens in the input.

Sentence:

> Mrs. Hills said that the U.S. is still concerned about "disturbing developments in Turkey and continuing slow progress in Malaysia."

TTP Approximate Parse:

```
(sent
  (np
    (name
      mrs
      .
      hills))
  (vp
    (verb said)
    (thats
      (compl that)
      (sent
        (np
          (t_pos the)
          (name u.s.))
        (vp
          (verb
            is
            (adv still))
          (venpass
            (verb concerned)))))
    ((vrbtm about)
      (vrbtm ")
      (((np
        (adj disturbing)
        (n developments)))
        (pp
        (prep in)
        (np
          (np
            (name turkey))
          and
          (np
            (a_pos_v continuing)
            (adj slow)
            (n progress))))
        (pp
        (prep in)
        (np
          (name
            malaysia
            .))))
      (vrbtm "))))
```

As may be expected, this kind of action involves a great deal of indeterminacy which, in case of natural language strings, is compounded by the high degree of lexical ambiguity. If the purpose of this skip-and-fit technique is to get the parser smoothly through even the most complex strings, the amount of additional backtracking caused by the lexical level ambiguity is certain to defeat it. Without lexical disambiguation of input, the parser's performance will deteriorate, even if the skipping is limited only to certain types of adverbial adjuncts. The most common cases of lexical ambiguity are those of a plural noun (nns) vs. a singular verb (vbz), a singular noun (nn) vs. a plural or infinitive verb (vbp,vb), and a past tense verb (vbd) vs. a past participle (vbn), as illustrated in the following example.

> "The notation used (vbn or vbd?) explicitly associates (nns or vbz?) a data structure (vb or nn?) shared

(vbn or vbd?) by concurrent processes (nns or vbz?) with operations defined (vbn or vbd?) on it."

We use a stochastic tagger to process the input text prior to parsing. The tagger, developed at BBN (see Meteer et al., 1991) is based upon a bi-gram model and it selects most likely tag for a word given co-occurrence probabilities computed from a relatively small training set. The input to TTP looks more like the following:

"The/dt notation/nn used/vbn explicitly/rb associates/vbz a/dt data/nns structure/nn shared/vbn by/in concurrent/jj processes/nns with/in operations/nns defined/vbn on/in it/pp ./."

In a 'normal' operation, TTP produces a regularized representation of each parsed sentence that reflects the sentence's logical structure. This representation may differ considerably from a standard parse tree, in that the constituents get moved around (e.g., de-passivization), and the phrases are organized recursively around their head elements. However, for the purpose of the evaluation with Parseval an 'input-bracketing' version has been created. In this version the skipped-over material is simply left unbracketed.

As the parsing proceeds, each sentence receives a new slot of time during which its parse is to be returned. The amount of time allotted to any particular sentence can be regulated to obtain an acceptable compromise between parser's speed and accuracy. In our experiments we found that 0.5 sec/sentence time slot was appropriate for the Wall Street Journal articles (the average length of the sentence in our WSJ collection is 17 words). We must note here that giving the parser more time per sentence doesn't always mean that a better (more accurate) parse will be obtained. For complex or extra-grammatical structures we are likely to be better off if we do not allow the parser wander around for too long: the most likely interpretation of an unexpected input is probably the one generated early (the grammar rule ordering enforces some preferences). In fact, our experiments indicate that as the 'normal' parsing time is extended, the accuracy of the produced parse increases at an ever slowering pace, peaking for a certain value, then declining slightly to eventually stabilize at a constant level. This final level-off indicates, we believe, an inherent limit in the coverage of the underlying grammar.

3 The TTP Time-out Mechanism

The time-out mechanism is implemented using a straightforward parameter passing and is limited to only a subset of nonterminals used by the grammar. Suppose that X is such a nonterminal, and that it appears on the right-hand side of a production S → X Y Z. The set of "starters" is computed for Y, which consists of the word tags that can occur as the left-most constituent of Y. This set is passed as a parameter while the parser attempts to recognize X in the input. If X is recognized successfully within a preset time, then the parser proceeds to parse a Y, and nothing else happens. On the other hand, if the parser cannot determine whether there is an X in the input or not, that is, it neither succeeds nor fails in parsing X before being timed out, the unfinished X constituent is closed (reduced) with a partial parse, and the parser is restarted at the closest element from the starters set for Y that can be found in the remainder of the input. If Y rewrites to an empty string, the starters for Z to the right of Y are added to the starters for Y and both sets are passed as a parameter to X. As an example consider the following clauses in the TTP parser:

```
sentence(P) :-
  assertion([],P).
assertion(SR,P) :-
  clause(SR,P1),
  s_coord(SR,P1,P).
clause(SR,P) :-
  sa([pdt,dt,cd,pp,ppS,jj,jjr,jjs,nn,nns,np,nps],P2),
  subject([vbd,vbz,vbp],P1),
  verbphrase(SR,P1,P2,P).
thats(SR,P) :-
  that,
  assertion(SR,P).
```

In the above code, P, P1, and P2 represent partial parse structures, while SR is a set of starter word tags where the parsing will resume should the present nonterminal be timed-out. First arguments to 'assertion', 'sa', and 'subject' are also sets of starter tags. In the 'clause' production above, a (finite) clause rewrites into a left sentence adjunct ('sa'), a 'subject', and a 'verbphrase'. If 'sa' is aborted before its evaluation is complete, the parser will jump over some elements of the unparsed portion of the input looking for a word that could begin a subject phrase: a pre-determiner (pdt), a determiner (dt), a count word (cd), a pronoun (pp,ppS), an adjective (jj, jjr, jjs), a noun (nn, nns), or a proper name (np, nps). Likewise, when 'subject' is timed out, the parser will restart with 'verbphrase' at either vbz, vbd or vbp (finite forms of a verb). Note that if 'verbphrase' is timed out both 'verbphrase' and 'clause' will be closed, and the parser will restart at an element of set SR passed down to 'clause' from assertion. Note also that in the top-level production for a sentence the starter set for 'assertion' is initialized to be empty: if the failure occurs at this level, no continuation is possible.

The forced reduction and skip-over are carried out through special productions that are activated only after a preset amount of time has elapsed since the start of parsing. For example, 'subject' is defined as follows:

```
subject(SR,PG) :-
   timed_out,!,
   skip(SR),
   store(PG).
subject(SR,P) :-
   noun_phrase(SR,P).
```

When a non-terminal is timed out and the parser jumps over a non-zero length fragment of input, it is assumed that the skipped part was some sub-constituent of the reduced non-terminal (e.g., subject). Accordingly, a place holder (PG) is left in the parse structure under the node dominated by this non-terminal. This placeholder will be later filled by some material recovered from the skipped-over fragment which is put aside by store(PG). In the bracketing-only version of TTP unparsed fragments are placed verbatim within the outmost bracket of the timed-out constituent, e.g.,

```
(S
   (NP we)
   (VP
      (V receive)
      (NP more pro letters
               (VRBTM than) (VRBTM con))
   ...
   ))
```

Note that (VRBTM *) brackets are invisible to the evaluation program, as will be explained in the next section.

There are a few caveats in the skip-and-fit parsing strategy just outlined which warrant further explanation. In particular, the following problems must be resolved to assure parser's effectiveness:

(a) how to select starter tags for non-terminals, and

(b) how to select non-terminals at which input skipping can occur.

Obviously some tags are more likely to occur at the left-most position of a constituent than others. Theoretically, a subject phrase can start with a word tagged with any element from the following list (still not a complete list): pdt (pre-determiner), dt (determiner), cd (numerical), jj, jjr, jjs (adjective), pp, ppS (pronoun), nn, nns (noun), np, nps (name), vbg, vbn (participle), rb (adverb), in (preposition). In practice, however, we may select only a subset of these, as shown in the 'clause' production above. Although we now risk missing the left boundary of the subject phrase in some sentences, while skipping an adjunct to their left, most cases are still covered and the chances of making a serious misinterpretation of input are significantly lower.

On the other hand certain adjunct phrases may be of little interest, possibly because of their typically low information contents, and we may choose to ignore them altogether. Thus in the 'ntovo' object string production below (where 'to' is the tag of word 'to'):

```
ntovo(SR,[P—PSA]) :-
   subject([to],P1),
   sa([to],PSA),
   tovo(SR,P1,P).
```

'sa' (sentential adjunct) material is skipped entirely if 'subject' is timed out. We must note here that this scheme will behave as expected only if there is no other 'to' tag between the skip point and the beginning of 'tovo' phrase, e.g.,

```
"Financial planners often urge
 investors to diversify ... "
|___ NTOVO ___|
  N(P) + TO + V(P)
```

When this is not the case, skipping may have to be redone. As an example, consider the following fragment:

```
"... urge
those flying to New York to take ... "
|____ NTOVO _____________|
|____ NP ___________| to |_ VP _|
```

If 'subject' of 'ntovo' is timed-out, the parser will first jump to "to (New York)", and only after failing to find a verb ("New") will redo 'skip' in order to take a longer leap to the next 'to'. This example shows that a great deal of indeterminacy still remains even in the tagged text, and that the final selection of skip points and starter tags may require some training.

A related problem is the selection of nonterminals that can be allowed to time out. This is normally restricted to various less-than-essential adjunct-type constituents, including adverbials, some prepositional phrases, relative clauses, etc. Major sentential constituents such as subject or verbphrase should not be timed (though their sub-constituents can), or we risk to generate very uninteresting parses. Note that a timed-out phrase is not lost, but its links with the main parse structure (e.g., traces in relative clauses) may be severed, though not necessarily beyond repair. Another important restriction is to avoid introduction of spurious dummy phrases, for example, in order to force an object on a transitive verb. The time-out points must be placed in such a way that while the above principles are observed, the parser is guaranteed a swift exit when in the skip-and-fit mode. In other words, we do not want the parser to get trapped in inadvertently created dead ends, hopelessly trying to fit the parse.

As an additional safety valve, a second time-out signal can be issued to catch any processes still operating beyond a reasonable time after the first time-out. In this case, a relaxed skipping protocol is adopted with skips to only major constituents, or outright to the end of the input. Dummy constituents may be introduced if necessary to close a parse fast. This, however, happens rarely if the parser is designed carefully. While parsing 4 million sentences (85 million words) of Wall Street Journal articles, the second time-out was invoked less than 10 times.

4 Parser Evaluation With Parseval

Parseval is a quantitative method of parser evaluation which compares constituent bracketings in the parse's output with a set of 'standard' bracketings. A parse is understood as a system of labeled brackets imposed upon the input sentence, with no changes in either word order or form permitted. Using three separate scores of 'crossings', recall and precision assigned to each parse (and explained in more detail below) the measure determines parser's accuracy indicating how close it is to the standard. For the purpose of this evaluation Penn Treebank bracketings have been adopted as standard.

In the rest of this section we demonstrate how Parseval typically processes a sentence. The example used here is sentence 337 from Brown Corpus, one of the set of 50 sentences used in initial evaluation. In this example, the sentence has been processed with TTP time-out set at 700 msecs.

Sentence 337

"Mr. Nixon, for his part, would
oppose intervention in Cuba without
specific provocation."

TTP PARSE (lispified)

```
(S
   (NP (NAME mr. nixon))
   (VP
      ,
      (PP
         (PREP for)
         (NP (T_POS (POSS his)) (N part)))
      ,
      (VERB would)
      (VO
        (VERB oppose)
        (NP (N intervention)))
     (PP
        (PREP in)
        (NP (N cuba)))
     (PP
        (PREP without)
        (NP (ADJ specific) (N provocation))))
  .)
```

The first two steps that the evaluator takes is to delete certain kinds of lesser constituents. This is done because of a great variety of treatment of these items across different parsers, and their relatively minor role in deciding correctness of a parse. The first phase deletes the following types of token strings from the parse:

1. Auxiliaries – "might have been understood" → "understood"
2. "Not" – "should not have come" → "come"
3. Pre-infinitival "to" – "not to answer" → "answer"
4. Null categories – (NP ()) → (NP)
5. Possessive endings – "Lori's mother" → "Lori mother"
6. Word-external punctuation (quotes, commas, periods, dashes, etc.)

The revised parse of sentence 337 is shown below.

```
(S
  (NP (NAME mr nixon))
  (VP
     (PP
        (PREP for)
        (NP (T_POS (POSS his)) (N part)))
     (VERB )
     (VO
        (VERB oppose)
        (NP (N intervention)))
     (PP
        (PREP in)
        (NP (N cuba)))
     (PP
        (PREP without)
        (NP (ADJ specific) (N provocation))))
  )
```

Subsequently, certain parenthesis pairs are recursively deleted, namely those that enclose either a single constituent or word, or an empty string. After this phase, sentence 337 looks like the following:

```
(S
  (NP mr nixon)
  (VP
     (PP for
        (NP his part))
     (VO oppose intervention)
     (PP in cuba)
     (PP without
        (NP specific provocation))))
```

Now the parse can be compared against the standard, which is shown below:

```
(S
  (NP mr nixon)
  (PP for
     (NP his part))
  (VP oppose
     (NP intervention
        (PP in cuba)
        (PP without
           (NP specific provocation)))))
```

We may note that there are several differences between the two structures, the most apparent is that various PP phrases are made part of VP in one of them while not in the other. In addition, TTP's VO constituent creates a "crossing" fault with respect to the standard, as it encompasses neither a subset nor a superset of any standard bracketing.

...(VO oppose intervention) ...

...(VP oppose (NP intervention ...

Other measures of closeness between these two structures are recall and precision, defined as follows:

$$recall = \frac{\#\ standard\ constituents\ in\ candidate}{\#\ constituents\ in\ standard}$$

$$prec = \frac{\#\ candidate\ constituents\ in\ standard}{\#\ constituents\ in\ candidate}$$

In the current example, both the candidate and the standard parses have 9 constituents. Seven of these constituents are common to one another, and there is one crossing fault. Therefore this TTP parse is evaluated as follows:

crossings	=	1
recall	=	77.78%
precision	=	77.78%

5 Evaluation of TTP

Two sets of evaluations were performed with TTP. In the first set, 50 sentences from Brown Corpus were used. In the series of runs with timeout value ranging from 100 to 2500 msecs, TTP scores varied from average crossing rate of 1.38 (for 100 msec time-out) to 0.82 (at 1700 msec); recall from the low of 59.37 (at 1800 msec!) to 62.02 (at 500 msec); and precision from 70.52 (at 100 msec) to 77.06 (at 1400 msec). The mean scores were: crossing rate 0.92, recall 60.57% and precision 75.69%. These scores reflect both the quality of the parser as well as the differences between grammar systems used in TTP and in preparing the standard parses. For example, average recall scores for hand parsed Brown sentences varied from 79% (for LSP grammar on which TTP is based; Sager, 1981) to 97%, with the average of 94%. We also plotted average parsing time per sentence for various time-out values. This is summarized in the table below (timing values are in milliseconds). These results were obtained on Sun SparcStationELC.

T/O	C	R	P	TIME
100	1.38	61.54	70.52	160
200	1.24	60.10	72.05	200
500	1.00	62.02	75.22	300
600	0.88	62.02	76.33	316
1000	0.90	61.78	76.49	380
1500	0.86	59.86	75.91	420
2000	0.86	59.37	76.00	460
2500	0.82	60.34	76.76	490
Mean	0.92	60.57	75.69	

The second set of tests involved 100 sentences form Wall Street Journal. Since WSJ sentences were usually longer and more complex than the 50 Brown sentences, we used time-outs of 250 msecs and more. The table below summarizes average crossings, recall and precision scores for TTP. Note that the performance peaks at timeout 500 and 750 msecs. These results are for a Sun SparcStation2.

T/O	**C**		**R**	**P**	**TIME**
	TOT	**PER**			
250	71	2.91	55.08	61.50	305
500	70	2.60	55.22	63.16	438
600	72	2.68	54.20	62.51	477
750	69	2.57	55.22	63.85	540
1500	68	2.60	54.57	64.01	797
30000	59	2.17	51.79	66.26	2930

The above statistics can be compared with those obtained parsing the same set of 100 sentences with a 'regular' parser: NYU's Proteus Parser (Grishman, 1990). Both parsers are based on the same grammar system (although Proteus grammar provides a much better coverage of English).

Proteus Statistics

C		R	P	TIME
TOT	PER			
56	2.34	63.74	62.87	24000

TTP Best Statistics

T/O	**C**		**R**	**P**	**T**
	TOT	**PER**			
500	70	2.60	55.22	63.16	438
% change					
	+25%	+11	-13	+0.4	
750	69	2.57	55.22	63.85	540
% change					
	+23%	+9.8	-13	+1.5	
30000	59	2.17	51.79	66.26	2930
% change					
	+5%	-7	-19	+5	

One should note that the per-sentence crossing ratio and recall score indicate how good the parser is at discovering the correct bracketings (precision is less useful as it already includes crossing errors). Clearly, both the crossing ratio and precision improves as we let the parser take more time to complete its job. On the other hand, the recall, after an initial increase, declines somewhat for larger values of time-out. This, we believe, points to the limitations of the underlying grammar used in these tests: initial correct hypotheses (enforced by preferences within the parser) are replaced by less likely ones when the parser is forced to backtrack.

6 Conclusions

At present TTP is a part of a natural language information retrieval system. Along with a stochastic part-of-speech tagger, morpho-lexical stemmer and phrase extractor, it constitutes the linguistic pre-processor built on top of the statistical information retrieval system PRISE, developed at NIST. During the database creation process TTP is used to parse documents so that appropriate indexing phrases can be identified with a high degree of accuracy. Both phrases and single-word terms are selected, along with their immediate syntactic context which is used to generate semantic word associations and create a domain-specific level-1 thesaurus. For TREC-1 conference concluded last November, the total of 500 MBytes of Wall Street Journal articles have been parsed. This is approximately 4 million sentences, and it took about 2 workstation-weeks to process. While the quality of parsing was less than perfect, it was nonetheless quite good. In various experiments with the final database we noted an increase of retrieval precision over the purely statistical base system that ranged from 6% (notable) to more than 13% (significant). Therefore, at least in applications such as document retrieval or information extraction, TTP-level parsing appears entirely sufficient, while its high speed and robustness makes an otherwise impractical task of linguistic text processing, quite manageable, and moreover, at par with other statistical parts of the system.

Further development of TTP will continue, especially expanding its base grammar to bring the coverage closer to Sager's LSP or Grishman's Proteus. We are also investigating ways of automated generation of skipping parsers like TTP out of any full grammar parser, a process we call 'ttpization'. TTP has been made available for research purposes to several sites outside NYU, where it is used in variety of applications ranging from information extraction from text to optical readers.

Acknowledgements

This paper is based upon work supported by the Advanced Research Project Agency under Contract N00014-90-J-1851 from the Office of Naval Research, Contract N00600-88-D-3717 from PRC Inc., the National Science Foundation under Grant IRI-89-02304, and by the Canadian Institute for Robotics and Intelligent Systems (IRIS).

References

Church, Kenneth Ward and Patrick Hanks. 1990. "Word association norms, mutual information, and lexicography." Computational Linguistics, 16(1), MIT Press, pp. 22–29.

De Marcken, Carl G. 1990. "Parsing the LOB corpus." Proceedings of the 28th Meeting of the ACL, Pittsburgh, PA. pp. 243–251.

Grishman, Ralph. 1990. "Proteus Parser Reference Manual." Proteus Project Memorandum #4-C, Courant Institute of Mathematical Sciences, New4-C, Courant Institute of Mathematical Sciences, New4-C, Courant Institute of Mathematical Sciences, New York University.

Harrison, P., S. Abney, E. Black, D. Flickinger, C. Gdaniec, R. Grishman, D. Hindle, R. Ingria, M. Marcus, B. Santorini, T. Strzalkowski. 1991. "Evaluating Syntax Performance of Parser/Grammars of English." Natual Language Processing Systems Evaluation Workshop, Berkeley, CA. pp. 71–78.

Hindle, Donald. 1983. "User manual of Fidditch, a deterministic parser." Naval Research Laboratory Technical Memorandum 7590-142.

Hindle, Donald and Mats Rooth. 1991. "Structural Ambiguity and Lexical Relations." Proceedings of the 29th Meeting of the ACL, Berkeley, CA. pp. 229–236.

Meteer, Marie, Richard Schwartz, and Ralph Weischedel. 1991. "Studies in Part of Speech Labeling." Proceedings of the 4th DARPA Speech and Natural Language Workshop, Morgan-Kaufman, San Mateo, CA. pp. 331–336.

Ruge, Gerda, Christoph Schwarz, Amy J. Warner. 1991. "Effectiveness and Efficiency in Natural Language Processing for Large Amounts of Text." Journal of the ASIS, 42(6), pp. 450-456.

Sager, Naomi. 1981. *Natural Language Information Processing.* Addison-Wesley.

Strzalkowski, Tomek and Barbara Vauthey. 1992. "Information Retrieval Using Robust Natural Language Processing." Proceedings of the 30th ACL Meeting, Newark, DE, June-July. pp. 104–111.

Strzalkowski, Tomek. 1992. "TTP: A Fast and Robust Parser for Natural Language." Proceedings of COLING-92, Nantes, France, July 23–28.

Strzalkowski, Tomek. 1992. "Natural Language Processing in Large-Scale Text Retrieval Tasks." First Text Retrieval Conference (TREC-1), Rockville, Md, November 4–6.

Appendix: Example Parses Produced by TTP

All sentences in the examples below are from Wall Street Journal sample. Parses obtained at 750 msecs time-out on Sun SparcStation2.

Sentence:

> Mr. McGovern, 63, had been under intense pressure from the board to boost Campbell's mediocre performance to the level of other food companies.

TTP PARSE:

```
((sent
  (np
    (name
      mr
      .
      mcgovern)
    (rn
      (((punct ,))
        (appos
          (np
            (count 63)))
        ,)))
  (vp
    (verb had)
    (veno
      (verb been)
      ((pp
        (prep under)
        (np
          (adj intense)
          (n pressure))))
       ((pp
         (prep from)
         (np
           (t_pos the)
           (n board))))
       (tovo
         (prep to)
         (tvp
           (verb boost)
           (np
             (t_pos
               (poss
                 ((name campbell))
                 's))
             (adj mediocre)
             (n performance))
           ((pp
             (prep to)
             (np
               (t_pos the)
               (n level)
               (rn
                 (pp
                   (prep of)
                   (np
                     (adj other)
                     (n_pos
                       (np
                         (n food)))
                     (n companies)))))))))))
  .)
```

CROSSINGS:	1
RECALL :	86.67%
PRECISION:	81.25%

Sentence:

Any money in excess of $40 million collected from the fees in fiscal 1990 would go to the Treasury at large.

TTP PARSE:

```
((sent
  (np
    (t_pos any)
    (n money))
  (vp
    ((pp
      (prep in)
      (np
        (n excess)
        (rn
          (pp
            (prep of)
            (np
              (n
                $
                40
                million)
              (rn
                (rn_wh
                  (venpass
                    (verb collected)
                    ((pp
                      (prep from)
                      (np
                        (t_pos the)
                        (n fees))))
                    ((pp
                      (prep in)
                      (np
                        (count
                          fiscal
                          1990)))))))))))))
    (verb would)
    (vo
      (verb go)
      (pp
        (prep to)
        (np
          (t_pos the)
          (n treasury))))
    ((pp
      (prep at)
      (np
        (n large))))))
 .)
```

CROSSINGS: 7
RECALL : 50.00%
PRECISION: 47.06%

Sentence:

RJR Nabisco Inc. and American Brands Inc. say they have no plans to follow Philip Morris's lead.

TTP PARSE:

```
((sent
  (np
    (np
      (name
        rjr
        nabisco
        inc
        .))
      and
      (np
        (name
          american
          brands
          inc
          .)))
  (vp
    (verb say)
    (thats
      (compl ())
      (sent
        (np
          (n they))
        (vp
          (verb have)
          (np
            (t_pos no)
            (n plans)
            (rn
              (rn_wh
                (tovo
                  (prep to)
                  (tvp
                    (verb follow)
                    (np
                      (t_pos
                        (poss
                          ((name
                             philip
                             morris))
                          's))
                      (n lead)))))))))))
  .)
```

CROSSINGS: 0
RECALL : 100%
PRECISION: 100%

Sentence:

The Health Insurance Association of America, an insurers' trade group, acknowledges that stiff competition among its members to insure businesses likely to be good risks during the first year of coverage has aggravated the problem in the small-business market.

TTP PARSE:

```
((sent
 (np
  (t_pos the)
  (n_pos
   (np
    (n health))
   (n_pos
    (np
     (n insurance))))
  (n association)
  (rn
   (pp
    (prep of)
    (np
     (name america)))
   (rn
    (((punct ,))
     (appos
      (np
       (t_pos an)
       (n_pos
        (poss
         (n insurers)
         ')
        (n_pos
         (np
          (n trade))))
       (n group)))
     ,))))
 (vp
  (verb acknowledges)
  (np
   (t_pos that)
   (adj stiff)
   (n competition))
  ((pp
   (prep among)
   (np
    (t_pos
     (poss its))
    (n members)
    (rn
     (rn_wh
      (tovo
       (prep to)
       (tvp
        (verb insure)
        (np
         (n businesses)
         (rn
          (rn_wh
           ((verb likely)))
          (rn
           (rn_wh
            (tovo
             (prep to)
             (tvp
                (verb be)
                (objbe
                  ((np
                     (adj good)
                     (n risks))))))))))))))))
 ((pp
  (prep during)
  (np
   (t_pos the)
   (adj first)
   (n year)
   (rn
    (pp
     (prep of)
     (np
      (n coverage)
      (((vrbtm has))
       ((wh_rel
         (venpass
          (verb aggravated))))
       (((np
          (t_pos the)
          (n problem)))
        ((pp
          (prep in)
          (np
           (t_pos the)
           (n_pos
            (np
             (n small-business)))
            (n market)))))))))))))
 .)
```

CROSSINGS:	13
RECALL :	38.46%
PRECISION:	37.04%

Sentence:

All three major creditors – the IRS, Minpeco and Manufacturers Hanover – voted against and effectively doomed a reorganization plan proposed by Mr. Hunt.

TTP PARSE:

```
(((np
     (t_pos all)
     (count three)
     (adj major)
     (n creditors)))
  (vrbtm –)
  ((np
       (t_pos the)
       (name irs)))
  (vrbtm ,)
  ((np
       (name minpeco)))
  (vrbtm and)
  ((np
       (name
          manufacturers
          hanover)))
  (vrbtm –)
  (vrbtm voted)
  (vrbtm against)
  (vrbtm and)
  (vrbtm effectively)
  (wh_rel
       (venpass
          (verb doomed)))
  ((np
       (t_pos a)
       (n_pos
          (np
             (n reorganization)))
       (n plan)
       (rn
          (rn_wh
             (venpass
                (verb proposed)
                (pp
                   (prep by)
                   (np
                      (name
                         mr
                         .
                         hunt))))))))
 .)
```

CROSSINGS:	0
RECALL :	57.14%
PRECISION:	100.00%