AN INTELLIGENT EDITOR FOR NATURAL LANGUAGE PROCESSING OF UNRESTRICTED TEXT

by

DEMETRIOS GEORGE GLINOS
B.S. Trinity College, 1973
J.D. Georgetown University Law Center, 1976

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the School of Computer Science
in the College of Arts and Sciences
at the University of Central Florida
Orlando, Florida

Summer Term
1999

ABSTRACT

The understanding of natural language by computational methods has been a continuing and elusive problem in artificial intelligence. In recent years there has been a resurgence in natural language processing research. Much of this work has been on empirical or corpus-based methods which use a data-driven approach to train systems on large amounts of real language data. Using corpus-based methods, the performance of part-of-speech (POS) taggers, which assign to the individual words of a sentence their appropriate part of speech category (e.g., noun, verb, preposition), now rivals human performance levels, achieving accuracies exceeding 95%. Such taggers have proved useful as preprocessors for such tasks as parsing, speech synthesis, and information retrieval.

Parsing remains, however, a difficult problem, even with the benefit of POS tagging. Moreover, as sentence length increases, there is a corresponding combinatorial explosion of alternative possible parses. Consider the following sentence from a New York Times online article:

> After Salinas was arrested for murder in 1995 and lawyers for the bank had begun monitoring his accounts, his personal banker in New York quietly advised Salinas' wife to move the money elsewhere, apparently without the consent of the legal department.

```
    ( (tag[i-1][0] != 'p') || (tag[i-1][1] != '-') ) &&
    ( (tag[i-1][0] != 'x') || (tag[i-1][1] != '-') ) &&
    ( i < (Znum - 1) ) &&
    ( saw_that < 0 ) &&
    ( saw_subj >= 0 ) &&
    ( saw_verb > 0 ) &&
    ( saw_subj < saw_verb ) &&
    ( ( first_to < 0 ) || ( saw_verb < first_to ) )
  )
{
   if( (tag[i-1][0] == 'C') && (tag[i-1][1] == 'C') )
   {
      i--;
   }
   else
   {
      shift_right(i, Znum, 1);

      saw_subj++;
      saw_verb++;
      saw_end++;
      for ( mm = 0; mm < num_verbs; mm++) verbs[mm]++;
   }

   token[i][0] = '.';
   token[i][1] = '\0';

   tag[i][0] = 'p';
   tag[i][1] = '-';
   tag[i][2] = 's';
   tag[i][3] = 'c';
   tag[i][4] = '\0';

   count++;
}

// Else if saw a verb phrase, split off a VP-coordination
// unless the clause has already been marked for splitting
// off or unless saw "to" before any other verb form
else if( ( i > 0 ) &&
         ( (tag[i-1][0] != 'p') || (tag[i-1][1] != '-') ) &&
         ( (tag[i-1][0] != 'x') || (tag[i-1][1] != '-') ) &&
         ( i < (Znum - 1) ) &&
         (saw_that < 0) &&
         (saw_verb > 0) &&
         ( ( first_to < 0 ) || ( saw_verb < first_to ) )
       )
{
   if( (tag[i-1][0] == 'C') || (tag[i-1][1] == 'C') )
   {
      i--;
   }
   else
   {
      shift_right(i, Znum, 1);

      saw_verb++;
      saw_end++;
```