B. PREFIX -- Revised Version

By John B. Smith

The PREFIX program described in detail in Automated Language Analysis: 1967-1968 has been revised during the past year. As a result, the logic has been greatly simplified -- the program is less than half as long as the first version -- and the run time reduced. Before discussing the revised algorithm, I shall describe briefly the data structures and the logic of the earlier version of PREFIX: however, the reader is urged to consult last year's annual report for a more detailed description and listing of the program.

The earlier version of PREFIX was basically a table-lookup procedure based upon a list of English prefixes, a list of words accompanying each list, and a key. A word that has initial letters that match a prefix does not necessarily have a legitimate prefix. For example, the word ate does not consist of the at prefix and a stem. Consequently, it was necessary to associate with each prefix additional information in the form of a list of words that indicated when a word with matching initial characters is or is not a legitimate prefixed word. Our task would have been simpler if we could have included in this list all words that are exceptions to the rule. This procedure would work well for the prefix in -- there are only some two to three hundred words listed in the Random House Dictionary that contain these initial letters but which are

not prefixed words. However, the exceptions for the prefix a are voluminous. The technique that was employed was to use either an exception or an inclusion list -- whichever is shorter -- and to indicate the kind of list by a key. Thus, associated with each prefix is a list of either exceptions or inclusion words and a key indicating the kind of list.

To optimize processing time, it was decided to assume that text words would be passed in alphabetical order against the alphabetized list of prefixes, resulting in the need for only one pass of both lists. This works well except when one prefix is "contained in" the succeeding prefix -- that is, when the first prefix is shorter in length than the prefix that follows it and when it matches the prefix that follows it character by character. For example, a is "contained in" ab, ad, etc. In this case, words with legitimate a prefixes may come after words with ab prefixes: atypical would come after all words with ab, ad, anti, etc., prefixes. To solve this problem a list of these "troublesome" prefixes was kept and after "normal" processing failed to reveal a prefix, the prefixes on this list were tested. The result was a procedure of some three hundred statements and fairly complex logic.

A simple and more elegant solution to the problem was found by using truth-table logic. (A truth table is matrix representation of all possible combinations of values for relevant parameters and the resulting logical condition.) In the case of PREFIX there are three relevant parameters: the initial letters of a word match or don't match an English

prefix, the word is or is not found in the associated CLUD list, and the CLUD list is an inclusion or exclusion list. The following table (Table # 1) summarizes all possible combinations of these factors and indicates whether or not the word has a legitimate prefix. Ones indicate that the parameter or condition is "true" or present; a zero indicates that the condition is not met.

Clearly, a word has a prefix only in those cases in which the initial letters match. But more significantly, it is meaningless to test for this condition until a check of the CLUD list and its accompanying key is made. To facilitate this procedure, a different data organization was used. The CLUD list was sorted alphabetically and a pointer attached to indicate the associated prefix and key (see Table 2). The complete text can thus be processed with a single pass against the alphabetized CLUD list. A scan of the CLUD list reveals whether a text word is on the list or not. The main processing procedure then branches into two simple nested conditional logic paths. If the word is found, then a test of the associated key is made. If it is a one, indicating an inclusion list, a test for prefix match is made. If all three tests are successful, then we can say that the word has a legitimate English prefix. On the other hand, if the text word is not found in the CLUD list, a quick search of the prefixes beginning with the same letter of the alphabet as the text word is made. Those, and only those, prefixes with a key of zero are tested against the initial letters of the word. If a match is found, we can conclude that

Initial Letter Match	CLUD Match	KEY	Has Prefix?
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 1

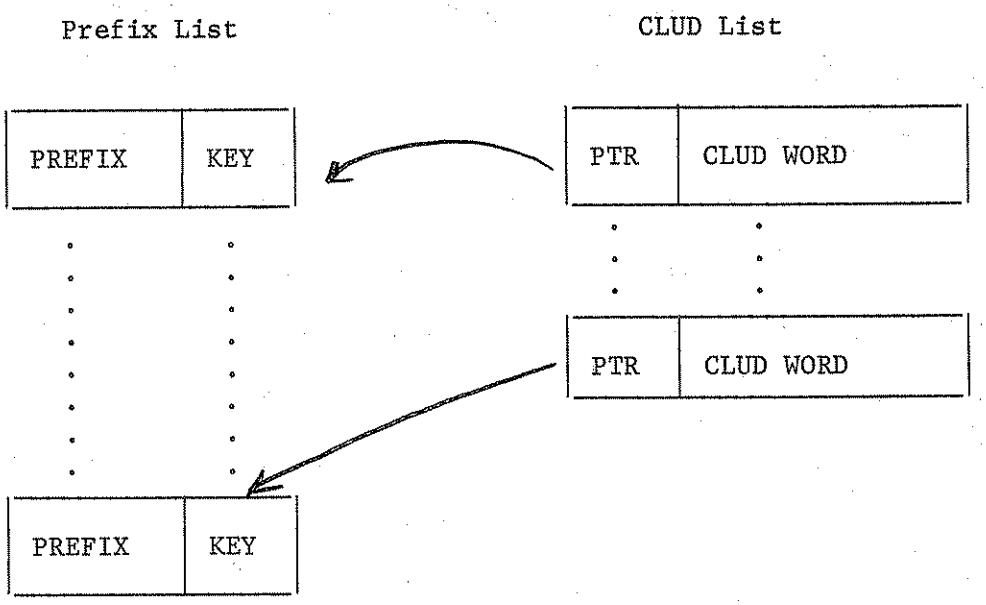

Table 2

the word has a prefix.

Thus, by changing the data representation, the processing algorithm is greatly simplified. Resulting processing time for a text of some 100,000 word-tokens is reduced from 8-10 minutes to 6-8 minutes, a 20-25% increase in efficiency. PREFIX is now catalogued in the VIA1 sequence. For instructions on how to use it, see the User's Manual included in this report.