4E-3

ＬａｎｇＬＡＢの辞書２次記憶化

西山研司　坂本浩一
(日立ソフトウェアエンジニアリング株式会社)

１．はじめに

ｐｒｏｌｏｇにより自然言語処理システムを構築する場合、辞書の保存形式が問題となる。最も単純な方式は、辞書データをｐｒｏｌｏｇのホーン節で表現し、それらをすべてｐｒｏｌｏｇプログラム中に常駐させる方法である。こうしておけば、ｐｒｏｌｏｇの特徴であるユニフィケーションによりシステムが勝手に辞書引きをしてくれるため、システム開発者が辞書引きに関していっさい考慮する必要がなかった。しかしながら、実用システムの構築時には、大規模な辞書の作成が必要であるため、プログラムのメモリ容量の問題から、辞書をすべてメモリに常駐させることができないので、２次記憶化を図る必要が生じる。

本稿では、東京工業大学情報工学科田中穂積研究室で開発されたＬａｎｇＬＡＢシステムの辞書２次記憶化と、その実用性の評価について報告する。

２．システム構成

ＬａｎｇＬＡＢは、ｐｒｏｌｏｇで記述された自然言語処理システムの開発支援システムである。図１にＬａｎｇＬＡＢシステムの構成を示す。

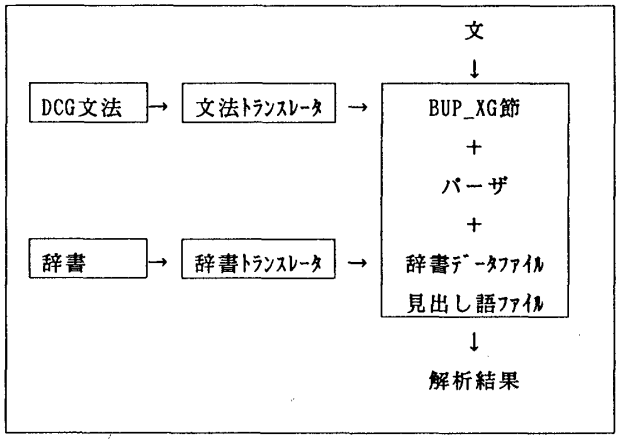

図１．ＬａｎｇＬＡＢシステムの構成

ＬａｎｇＬＡＢシステムは、ユーザが記述したＤＣＧ形式の文法規則をＢＵＰ＿ＸＧと呼ばれるｐｒｏｌｏｇプログラムに変換し、このプログラムとｐｒｏｌｏｇ記述のパーザおよび辞書に解析すべき文を与えると、構文処理と意味処理とを融合したボトムアップで縦型探索の自然言語処理を行うことができるシステムである。

３．辞書２次記憶化の方法

３．１　従来の辞書引き

従来のＬａｎｇＬＡＢにおいては、辞書引きは"dicta"という述語の起動により行っていた。

<辞書>

```
"door" : dicta(door,[[n,[[[cnt|'+']],door]]],[]).
"key"  : dicta(key,[[n,[[[cnt|'+']],key]]],[]).
```

<辞書引きの起動>

```
dicta(WORD,ARGs,TRIEs)　……①
```

ここで、例えば、WORDを'door'とユニファイした状態で、①を起動すると、dictaの第１項のWORDという変数と'door'のマッチングにより辞書引きが行われる。

３．２　辞書２次記憶化による辞書引き

(１)辞書ファイルの構成

辞書ファイルの構成を図２に示す。

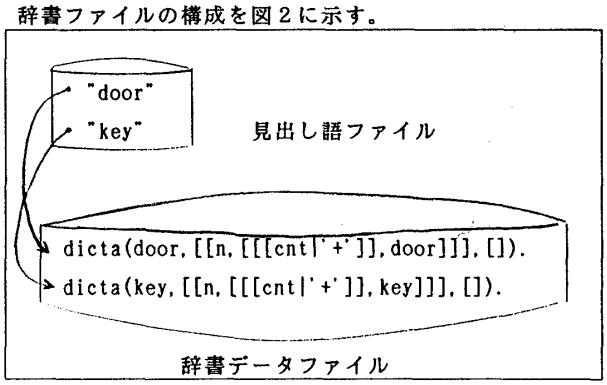

図２．辞書ファイルの構成

(２)辞書引き方法

新しい辞書引きのための述語を"dictb"とする。Ｃ言語で記述された辞書サーチルーチン"dict_search"により、辞書データファイル中の辞書ｐｒｏｌｏｇのホーン節を得る。得られたホーン節をassertした後、従来の辞書引

Implementation of a secondary memory dictionary in LangLAB
Kenji Nishiyama, Kouichi Sakamoto
Hitachi Software Engineering Co., Ltd

き述語"dicta"を起動する。

```
dictb(WORD, ARGs, TRIEs):-
        call(dict_search, [WORD], F), !, assert(F),
        dicta(WORD, ARGs, TRIEs).
```

(3)バッファによる高速化

辞書引き高速化のために、以下の二種類のバッファを設ける。ｐｒｏｌｏｇの場合、これらのバッファへの格納は、assert述語により行える。

(a)辞書バッファ

一度辞書引きした結果を格納

(b)未登録語バッファ

辞書引きの結果、未登録語であることがわかった単語を格納

以上のメカニズムを組み込んだdictbの内容を図３に示す。

```
dictb(WORD, ARGs, TRIEs) :-
      not_found(WORD), !, fail.                    ……①
dictb(WORD, ARGs, TRIEs) :-
      dicta(WORD, ARGs, TRIEs), !.                 ……②
dictb(WORD, ARGs, TRIEs) :-
      call(dict_search, [WORD], F), !, assert(F),
      dicta(WORD, ARGs, TRIEs).                    ……③
dictb(WORD, ARGs, TRIEs) :-
      assert(not_found(WORD)), !, fail.            ……④

①対象単語が未登録語バッファ(not_found)にあれば、
  failする。
②対象単語が辞書バッファ(dicta)にあれば、それを辞
  書引き結果とする。
③辞書ファイルのサーチ。
④辞書未登録語を未登録語バッファに格納。
```

図３．dictbの内容

４．評価

Ｃ－ｐｒｏｌｏｇ上で、ＬａｎｇＬＡＢの辞書を２次記憶化する前（辞書メモリ中）と後（辞書ファイル中）の実行結果を表１に示す。

表１の結果より、２次記憶に辞書をおいても、ＬａｎｇＬＡＢは十分高速で、実用に耐えうる速度であることがわかる。

５．おわりに

ＬａｎｇＬＡＢシステムを題材に、ｐｒｏｌｏｇで記述された自然言語解析システムの辞書２次記憶化の方法について述べた。

表１．辞書２次記憶化速度比較

例　　文	解析時間*1[msec]	
	メモリ	ファイル
I open the door with a key.	2,888	2,717
Tell me who swims in the river.	3,250	3,017
Tell me what to do.	2,000	1,850
I tell you that it happens.	5,167	4,967
I open not only the door but also the window.	2,217	1,850
I know as if he bought the car.	3,700	3,533
I saw the man in the park with a telescope.	5,817	5,733
It is not tied to a particular domain of applications.	8,133	7,767

*1:全解析木が得られる時間

[謝辞]

日頃御指導いただく東京工業大学田中穂積教授ならびに徳永助手をはじめ田中研究室の方々に深謝いたします。

[参考文献]

[１]上脇正、田中穂積、"辞書のＴＲＩＥ構造化と熟語処理" PROCEEDING OF LOGIC PROGRAMMING CONFERENCE '85 12.2.

[２]田中穂積、"自然言語処理システムＬａｎｇＬＡＢ"、情報処理学会第３３回予稿集(1986)

[３]奥村学、田中穂積、"ＬａｎｇＬＡＢにおける高水準辞書記述言語ＳＲＬ／０"、情報処理学会第３３回予稿集(1986)

[４]Hartono、田中穂積、"ＬａｎｇＬＡＢ上出のインドネシア語文法試作"、情報処理学会第３３回予稿集(1986)

[５]沼崎浩明、田中穂積、"ＬａｎｇＬＡＢにおける文法開発環境"、情報処理学会第３３回予稿集(1986)

[６]上脇正、田中穂積、沼崎浩明、"ＬａｎｇＬＡＢの構文処理アルゴリズムの高速化"、情報処理学会第３３回予稿集(1986)

[７]今野聡、田中穂積、"左外置を考慮したボトムアップ構文解析システム"、コンピュータソフトウェアvol.3 No.2(1986)

[８]QUEK CHEE HUEI、"英語の文法の機能拡張に関する研究"、東京工業大学情報工学科卒業論文(1988)