人文科学とコンピュータ　29－7
（1996．1．27）

日本語情報処理のためのプログラミング環境の設計と実現

天野純一　早川栄一　並木美太郎　高橋延匡
{namiki,hayakawa}@cc.tuat.ac.jp
東京農工大学　工学部

本稿では、日本語テキスト処理を行なうアプリケーションプログラム（AP）への支援を目的としたプログラミング環境について述べる。日本語は文中で語を分ち書きしないという特徴があり、このために、文字列検索で語を誤って検出するなどの現象が起きる。これは日本語を扱うAPに共通の問題である。そこで本研究では、形態素解析を利用して文中の語を認識し、さらにそれらを単位として基本的なテキスト処理を行なう機能を、Cのライブラリとして設計した。APはこれを用いて、表記や品詞を検索キーとして文章中の語の位置を得たり、語を読みで五十音順ソートしたりする機能を利用できる。現在までにその一部を実現した。

Design and Implementation of a Programming Environment for Japanese Text Processing

AMANO Jun-ichi,　HAYAKAWA Eiichi,
NAMIKI Mitarou　and　TAKAHASHI Nobumasa
Department of Computer Science,
Tokyo University of Agriculture and Technology

This paper describes a programming environment for application program(AP) with Japanese text processing. Japanese text does not show boundaries of words, and thereby character string search sometimes misdetects a word. Since APs with Japanese processing have this problem in common, a C language library is designed. This library recognizes words in Japanese text by morphological analysis, and offers basic text processing functions regards word as its unit. APs can use spelling and part-of-speech of a word as a search-key to find the position of the word in sentences, and can sort words with their pronunciation. Some parts of the library are implemented by now.

1．はじめに

テキストは検索性や記述性に優れたデータ形式であり、これらの作成や閲覧・検索などを目的として、多くのアプリケーションソフトウェア（AP）が作成されている。これらの AP のユーザは、ソートや検索などのテキスト処理を「語」に対して行なうことが多い。しかし日本語は語を分ち書きしないので、テキスト中の語を認識して処理する AP を作成することは難しい。

本稿では、語を単位としたテキスト処理を行なってこの問題に対処するCライブラリについて述べる。

2．日本語テキスト処理の問題点と現状

2.1 語単位テキスト処理の必要性

ユーザが AP のテキスト処理機能を利用する場合、語をその対象にすることが多い。たとえばワードプロセッサにおける置換や検索は、当て字を直したり参照したい項目の位置を得たりするために多く用いられる。AP によっては、文章の中から一定の条件を満たす語を抜き出すことが主な操作になる。たとえば筆者の研究室では現在、CD-ROM 辞書中の文の語から新たな見出し項目を作るシステムや、マニュアル文の中の意味を調べたい語にコメントを貼り付けるシステムを研究している。人文科学の分野では、コンコーダンス作成システム[1]のような AP を用いた、文学作品中の語の分布の解析なども考えられる。

このように、ユーザは文字列というよりも語を意識してテキスト処理を利用している。そこで、文字の他に語を処理単位としたテキスト処理が AP に求められる。

2.2 日本語の処理に伴う問題点

日本語の文では語の分ち書きを行なわないので、AP は文字列としてのテキストを与えられても語の区切りがわからない。そのためにテキスト処理を行なうと、ある語を検索すると別の語の一部に誤ってマッチしてしまう（例：「程」が「程度」に）ような現象が起きる。分ち書きをする英語では pen と pencil は区別できるが、日本語テキストは文字単位で扱う限りこのような区別はできない。

また日本語では、語はその表記の他に発音すなわち読みでも取り扱われることがある。したがって、表記の文字列を対象とした通常のテキスト処理では、読みを必要とするソートなどの処理で期待する結果を得られない。

2.3 語の認識手段

テキスト中の語を計算機が認識する手段として、一般に形態素解析が用いられる。日本語に対する形態素解析は数多くの実現例（[2][3]など）が報告されているが、これらの多くは解析処理だけでシステムが閉じており、AP への応用は考慮していない。たとえば、テキストの解析単位は一つの文なので、文章すなわち複数の文に対するテキスト処理には不便である。また、自然言語処理に伴う複雑なデータ構造や手続きが AP からの利用を妨げている。

2.4 他の研究の現状

形態素解析をライブラリ化した例にはリコーの小松による研究[4]がある。この例ではライブラリが AP に提供する機能は解析だけなので、テキスト処理は AP の側で作成しなければならない。

3．作成目的

前章で述べたように、日本語テキストを扱う AP には、語を単位にして検索・置換などの各種のテキスト処理を行なう機能を持つことが望ましい。このような基本的機能をあらかじめ用意しておけば多くの AP

に活用でき、AP の実現が少ない手間数で行なえる。また、語の認識に必要である形態素解析処理は AP にとって扱いにくい。

そこで、形態素解析を利用して語を単位にしたテキスト処理を行なう機能を、AP 記述言語のライブラリとして作成する。

4．設計方針

4.1 全体の方針

本ライブラリの設計方針を次に挙げる。

(1) 基本的なテキスト処理機能で構成する

ライブラリとして関数を作成する場合、語の類別などの高水準な機能を用意する方針と、検索などの基本的な処理を行なう関数をいくつか用意する方針が考えられる。テキスト処理を行なう AP は数多く考えられることから、本ライブラリでは汎用性を重視して後者を採る。

(2) 簡潔な API を提供する

自然言語処理は解析木の表現などに複雑なデータ構造や手続きが必要である。しかし本ライブラリの利用者は一般のAPプログラマであり、特別な知識は求めない。そこで、AP が直接扱うライブラリ関数やデータ構造はできるだけ簡潔で理解しやすい形にする。

(3) 主記憶の消費量を抑える

本ライブラリのテキスト処理機能が用いる形態素解析結果は、後から第２、第３候補を取り出すために、その曖昧性を解析木という形で主記憶のヒープ領域に保持する。一つのノードは数十バイトのデータからなるので、文章全体では数百キロバイトになることがある。AP が用いる主記憶容量をライブラリが圧迫しないために、解析木の内容のコピーなどは行なわない。

(4) 拡張性を重視する

本ライブラリは任意の AP を対象にすることから、AP 側から後にさまざまな要求が出ることが予想される。その中にはライブラリの拡張が必要なものもある。そこで、機能拡張をしやすい全体構成をとる。

4.2 実現する機能と全体構成

本ライブラリで実現する主な機能を挙げる。これらの機能と AP および形態素解析の関係を図１に示す。それぞれの機能の内容は次章で述べる。

(a) 形態素解析の実行機能
(b) テキスト中の語の検索機能
(c) テキスト中の語の置換機能
(d) 読みによる語のソート機能

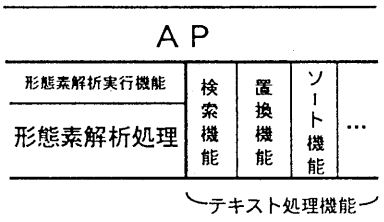

図１　ライブラリの全体構成

AP は初めに形態素解析実行機能 (a) に処理したいテキストを渡して解析結果を受け取る。解析結果の実体は文ごとに独立した木構造だが、次章で述べるように「文章」として扱えるように抽象化されている。テキスト処理機能 (b)(c)(d) を利用するときはこの結果を処理の対象として渡す。

このように、形態素解析機能とテキスト処理機能を分けたことによって、形態素解析の実行は一つのテキストに対して一度だけですむ。また、テキスト処理機能の拡張を、形態素解析に手を加えずに行なえる。

各機能の API を図２に示す。

機能	ライブラリ関数の形式
解析	文章型 *文章オープン(CHAR *テキスト); └解析された文章
検索	LONG *語検索(文章型 *文章, 語型 検索キー); └検出位置の配列 └処理対象の文章
置換	INT 語置換(文章型 *文章, LONG 位置, CHAR *表記); └置換の成功/失敗 置き換える表記┘
ソート	INT 語ソート(語型 *語, INT 個数); ソートする語の配列┘

図２　ライブラリ関数の API

５．AP に提供する機能の設計

5.1　形態素解析の実行機能

語単位でテキスト処理を行なうためには形態素解析された結果が必要であるが、AP からテキスト処理を呼ぶたびに解析しては効率が悪い。そこで、解析をテキスト処理から分離する。

本機能の動作の手順を述べる。AP はこれから処理をしたい主記憶上のテキストを本機能に渡す。本機能はテキスト中の各文を解析して主記憶中に解析木を展開する。さらにこれらを探索して各文の最尤候補を得る。

解析木は文ごとに独立しているので、そのままでは文章全体に対する処理を行なうことができない。そこで、文の各最尤候補を文章に現れる順に連結して、図３のように語に分割された「文章」としてAPに返す。

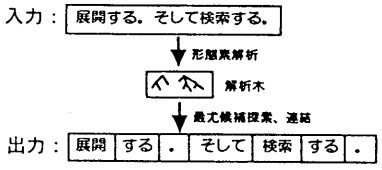

図３　文章の解析結果のイメージ

ここで、分割された各語の実体は解析木のノードであり、それぞれをポインタ変数で指している。しかし AP やテキスト処理機能からは、語は線形に連続しているように見ることができる。したがって、テキスト処理を文字単位と同じような API で行なうことができる。

5.2　語の検索機能

コンコーダンス作成システムで文章中からコンコーダンス項目になる語を選び出したり、校正支援システムで誤った表記の語を検出するなど、検索処理は文章に対するテキスト処理において中心的な機能である。これが日本語テキストを対象にする場合、文章中の語が分ち書きされないので、語を文字列検索で探すと「プロ」が「プログラム」にマッチするなど、ユーザが期待しない検出が起きることがある。そこで、語を検索キーにして文章中の語の位置を求める検索機能を作成する。

この機能は、AP から「文章」と検索キーを引数として受け取る。そして、検索キーにマッチする文章中の語の位置を AP に返す。

本機能を用いた AP の記述例を図４に示す。この例は、校正支援システム中で文章中の誤った表記の語の位置を返すという処理である。この中の「語単位検索()」が本機能を呼ぶライブラリ関数である。本機能では検索キーとして、語の表記の他に、形態素解析で得られる情報である品詞や活用形を許す。これによって、たとえば文学作品の解析における「文章中の形容動詞を抜き出したい」というような要求に応える。表記や品詞はここでは「語型」という名で定義された一つの変数に格納されるので、これらの検索条件を同時に適用することができる。

```
LONG    *誤った表記の検出(文章型 *文章)
{
        LONG    *位置;
        語型    検索キー;

        検索キー.表記 = "割り込み";
        検索キー.品詞 = 普通名詞;

        位置 = 語検索(文章, 検索キー);

        return 位置;
}
```

図5　語の検索機能の利用例

5.3　語の置換機能

ワードプロセッサにおける当て字や送り仮名の修正など、文章中の語の表記を直したりする場合のテキスト処理機能としては、置換機能が用いられる。通常の文字単位の置換で、置き換える語の長さと元の語の長さが異なるときを考える。たとえば「割り込み」を「割込み」に直す場合である。この場合は他の箇所の内容が置換によって壊されないように、置換箇所以降のテキストをバッファにいったんコピーして退避する必要がある。この操作は、対象になるテキストが大きくなるほど動作所要時間と使用記憶容量を増やす。そこで、文章中の語の位置を処理対象として指定させる置換機能を作成する。

この機能は、AP からは「文章」と置換対象の語の位置、および置換後の表記を引数として受け取る。そして図6のように、その語の表記を置換され、その結果は「文章」に反映される。

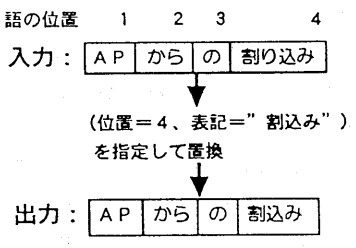

図6　置換機能のイメージ

表記の基準によっては、名詞は「割込み」動詞は「割り込む」と書くなど、品詞によって送り仮名が異なることがある。本機能と前述した検索機能を組み合わせて用いることによって、これらに対応した表記の修正を行なうことができる。その例を図7に示す。この例では、図5の関数を利用して、名詞「割り込み」の表記を「割込み」に修正する。また、同様にして用言の活用にも対応できる。

```
VOID    誤った表記の修正(文章型 *文章)
{
        LONG    *位置;

        位置 = 誤った表記の検出(文章);
        while (*位置++ != NULL)
                語置換(文章, *位置, "割込み");
}
```

図7　置換と検索を組み合わせる例

5.4　語のソート機能

コンコーダンスの項目は、ユーザの参照性を高めるために画面表示の際にそれらをソートする。校正支援システムでは、検出機能で拾い上げた語を五十音順に並べることによって「表す／表わす」などの同音異表記語を発見することがしやすくなる。また、われわれが現在研究中の学習用言語処理系では、BNF の非終端記号を読み順で並べて表示することによって、プログラミング言語の構造の理解を学習者に促す。

これらのように、複数の語を表記ではなく読みでソートする機能にはさまざまな用途が考えられる。しかし、漢字の読みはキーワードを語として扱うことで初めて確定する。また、仮名キーワードの順序は通常の文字コードによるソート規則と国語辞典で用いられている規則とでは、平仮名と片

仮名の評価などが異なる。

これらに対処する処理は AP にとって容易ではないので、ライブラリとして実現する。この機能は AP からソートするキーワード群とキーワードの個数を受け取る。キーワードが語の読みの情報を含んでいれば、それをソートのキーにする。ない場合は、辞書検索をして漢字仮名変換を行なう。そのとき、処理速度を重視して、「市場（いちば／しじょう）」などの曖昧性がある読みは第一候補を仮定する。仮名にされたキーワードを国語辞典[5]の見出し語順序規則に従って前後を判断し、クイックソートする。AP には、漢字表記のまま順序がソートされて返ってくる。本機能の利用例を図８に示す。また、その実行結果を図９に示す。

```
VOID    語の一覧表示(語型 *表示する語, INT 個数)
{
        INT     i;

        語ソート(表示する語, 個数);

        for (i=0; i<個数; i++){
                printf("No.%d:%s¥n", i+1, 表示する語->表記);
                表示する語++;
        }
}
```

図８　ソート機能の利用例

```
ふり
鰤
フリー
振仮名
振り仮名
ブリキ
フリゲート
振り子
ブリザード
プリズム
振り袖
振り出し
ブリッジ
振付け
ブリン
```

図９　ソート機能の実行結果の例

６．実現

本ライブラリは、筆者が所属する研究室で開発された形態素解析[2]を利用している。この解析では 151 項目の品詞接続表と約 10 万語の辞書を用い、各ノードとアークにコストを与える。そして、コストの合計が最小の経路を最尤候補とする「最小コストパス探索」を特徴にしている。

本ライブラリは、この形態素解析の実現環境であり、これも筆者の研究室で開発された「OS/omicron 第２版」とＣ処理系“CAT”の上で現在実現を行なっているところである。現在では、仮名文字列のソート処理が約 400 行の規模で実現されている。

７．おわりに

本稿では、文字ではなく語を単位にしてテキスト処理を行なうためのＣライブラリについて述べた。今後は実現を急ぎ、本稿で挙げた各種の AP への適用を行なう。

謝辞

本研究は、文部省科学研究費課題番号 06208102「沖縄の歴史情報研究」により行なわれた。

参考文献

[1] 坂口，他：コンコーダンスを指向したテキストデータベースの開発，情報処理学会人文科学とコンピュータ研究会，CH29-3（1996）

[2] 下村，他：最小コストパス探索モデルの形態素解析に基づく日本文誤り検出の一方式，情学論，Vol.33，No.4（1992）

[3] 木谷：固有名詞の特定機能を有する形態素解析処理，情学論，Vol.33，No.5（1992）

[4] 小松：日本語解析用クラスライブラリ「Jpn クラスライブラリ」の開発，第 48 回情学全大，4Q-5（1994）

[5] 金田一，他：新明解国語辞典第四版，三省堂（1989）