欧文文献のコンピュータ読み取りプログラムの開発

岩永正裕・山岸陽一

Development of Program Converting European Letters into Character Codes

Masahiro IWANAGA and Yōichi YAMAGISHI

Abstract

If we can take in literatures in European language in computer, more easily we can translate them and can construct our own dictionaries. There are some commercial programs and systems for that purpose, but they are not suitable for us because of their prices or their performances.

Then we developed the program to convert European letters into character codes by using C lanuage and the personal computer (PC-9801). The converting rate was about 5 charcters per second and the percentage of convertion was over 95%.

1. まえがき

欧文文献をコンピュータにキャラクタコードとして手軽に取り込むことができれば，日本語ワードプロセッサを活用して，学生に欧文の下に日本語訳を付加させ，その添削を行ったり，研究室独自の辞書を編集することも可能になると考えられる。

しかしこれを可能にするためには非常に高価なシステムが必要であったり，また安価なソフトがあっても必ずしも満足できる性能ではないのが現状である。

そこでパーソナルコンピュータ（日本電気製 PC-9801）上でC言語（TURBO C Ver. 1.5）を用いて上記ソフトを開発したので報告する。

2. 教育における本ソフトの必要性

ゼミナールや卒業研究において10人前後の学生に対して欧文文献の読解を課してきたが，必ずしも能率良く教育できたとは考えていない。

読み合わせの方法（学生が翻訳し，教員がその場で添削する）は，1人の学生の準備が不十分な場合途端に進度が遅くなること，教員が添削しても担当者以外の学生には十分に注意を払って聞き取ってもらえないこと，読解した日本語訳が形として残しにくいこと，学生がシャイな場合に皆の前で発表するときに十分にその実力を発揮してくれないことなどが欠点であった。

そこで2，3年前からパーソナルコンピュータ上で日本語ワードプロセッサを使用して，まず学生に欧文を打ち込ませ，各文の下に日本語訳を打たせる方法を取った。これによって上記の読み合わせによる欠点をほぼカバーでき，さらにほとんどの学生がコンピュータに興味をもっていることから当初我々が予想したよりもずっと欧文読解の能率が上がり，添削もワードプロセッサの機能を使用することにより比較的楽に行えることがわかった。しかし学生がキーボードに慣れない場合に，欧文の打ち込みにかなり苦労すること，タイプミスが多発し，添削にあたりまず原文と学生が打ち込んだ欧文を照らし合わせて誤りを正すという手間が増えることがわかった。

そこで欧文文献をキャラクタコードとしてコンピュータに取り込むソフトがあれば，上記の手間が省け，欧文文献を簡単にコンピュータ処理することができると考えた。さらに将来研究室独自の欧文辞書を編集することも可能となると考えた。研究室で使用しているパーソナルコンピュータ上で作動するこのような市販のソフトを捜したがほとんどなく，あっても十分な性能を持ち合わせていなかった。またこのような目的を達成するには別途高価なシステムを購入する必要

平成元年9月28日受理

があることがわかった。

以上が筆者らが非力を省みず，本ソフトを開発しようと決心したいきさつである。

3. ソフト開発環境

パーソナルコンピュータとして日本電気製PC-9801を使用し，欧文文献を画像としてコンピュータに取り込むためにイメージスキャナ（日本電気製PC-in503H）を，文献の読み取り範囲の設定にマウスを使用した。

ソフト開発言語としてはTURBO C Ver. 1.5を使用した。

4. 文字認識の方法

パターン認識の方法に関して筆者らはまったくの素人であり，次のように考えた。人間はパターン認識に優れると言われるがそれは次のような方法を用いているからではないか。

(1) 経験を多く記憶していて類似するものを捜す。

(2) 評価関数を1個ではなくいくつも持っている。

(3) 1つ1つの評価関数に対する許容範囲が比較的広い。

そこで本プログラムに関しても同様な方法が適用できないかを考えた。

(1) 文献をキャラクタコードに変換する過程で文字認識の基礎となる辞書を自然にユーザーが作成できるようにした。

この辞書に納める内容は次のものとした。

(a) 文字の画像データ

(b) 文字構成ドット数

(c) 文字の画像データを解析して，文字構成ドットをカバーできる最小の長方形領域Rを求め，その縦横のドット数。

(d) 領域Rの左上を原点とした文字構成ドットの縦横の重心位置 ξ, η。

$$\xi=\sum x\gamma/\text{文字構成総ドット}$$
$$\eta=\sum y\gamma/\text{文字構成総ドット}$$

ここに $\sum$ は文字構成ドットの全てに対しての和を表し，x, y は領域Rの左上を原点としたドット単位の座標，γ は整数割り算の精度を上げるための重み（定数）である。

(e) 重心位置を原点とした座標に関する二次モーメントを用いて計算される次の量。

$$\xi_2=\sqrt{(\sum (x\gamma-\xi)^2)}/\text{文字構成総ドット}$$
$$\eta_2=\sqrt{(\sum (y\gamma-\eta)^2)}/\text{文字構成総ドット}$$

(f) キャラクタコード

(2) 上記の辞書に記録する内容のうち(a), (c)～(e)を評価関数として以下のようにキャラクタコードを決定する。

（i） (c)(d)(e)の評価関数に関して許容範囲にある文字を辞書から選び出す。

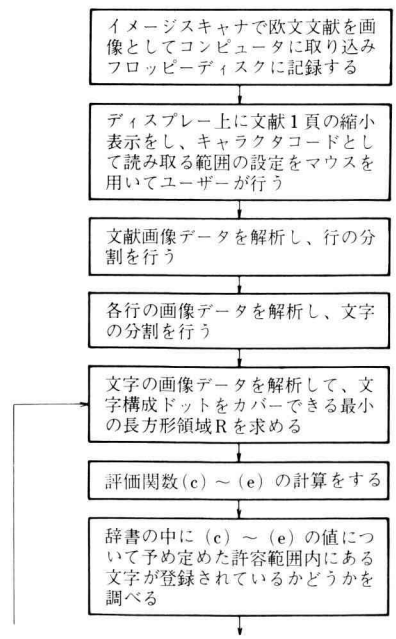
（ii） 調べている文字と選び出された文字の間で画像データを重ね合わせて，相違ドット数を計算する。

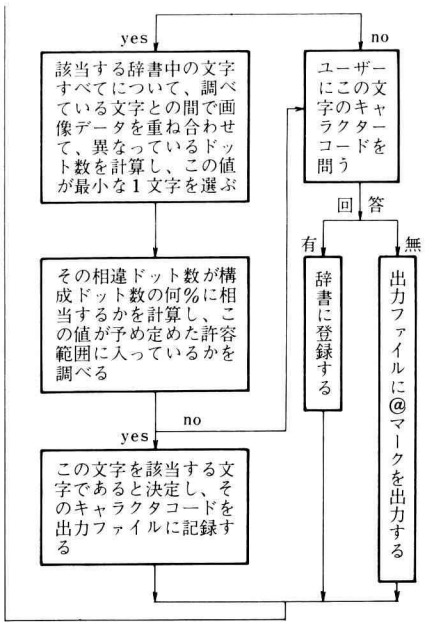
（iii） この相違ドット数が最小な文字を選び，その相違ドット数が構成ドット数の何％に相当するかを計算する。この値が許容範囲内に入っているときはじめてキャラクタコードを決定する。

(3) これらの評価関数に関する許容範囲の上限はユーザーが予め定義できるようにした。

5. プログラムの概要

本プログラムの流れは以下の通りである。





6. プログラム実行結果

1つの英語文献の読み取りを例に取り本プログラムの実行結果を示す。

初め辞書内容がゼロからスタートし，文献を読み取りながら辞書を作成する。この辞書を用いて同じ文献中のまだ読み取り処理をしていない頁を辞書の容量を変化させ読み取り，1秒に読み取る平均字数と，解読率を求める。Fig. 1はこのようにして求めた平均読み取り速度と辞書容量の関係である。辞書容量が増加するに従って読み取り速度は遅くなり，平均読み取り速度は1秒当り数文字である。Fig. 2は解読率と辞書容量の関係を示し，解読率は辞書容量が増加するに従って増加し辞書容量を増やせば95% 以上の解読率を達成できることがわかる。

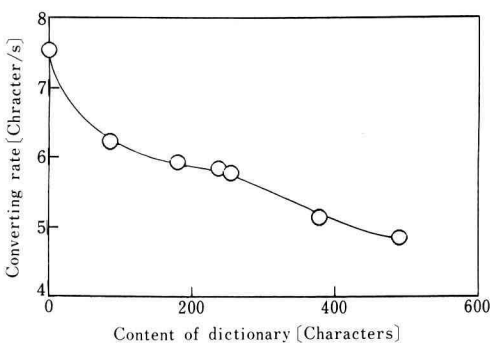


Fig. 1. Relation between the converting rate and the content of dictionary.

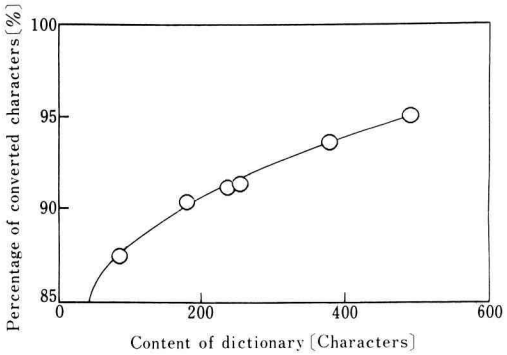


Fig. 2. Relation between the percentage of converted characters and the content of dictionary.

7. 評価関数の検討

評価関数が文字認識にあたってどのような役割を果たしているかを調べる。まず許容範囲を狭く取ってなるべく多くの文字(1,441 文字)を辞書に登録した。次に辞書に登録されている文字を辞書に登録されている他のコードの文字で解析し，誤読がどのように発生するかを調べた。Fig. 3は縦軸に誤読率を，横軸に文字パ

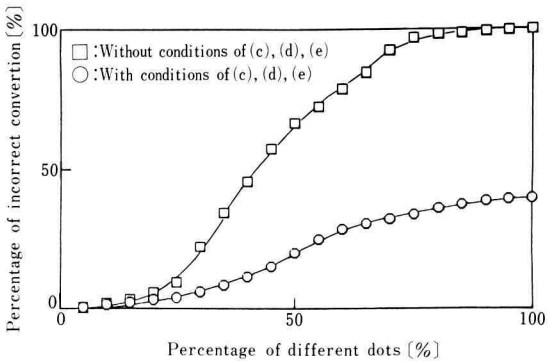


Fig. 3. Relation between the percentage of incorrect convertion and range of conditions.

ターン重ね合わせの誤差パーセントを取った。□は(c)(d)(e)の評価関数を使用しない場合，○はそれらを使用した場合の結果である。

文字パターン重ね合わせの誤差パーセントの許容範囲と(c)(d)(e)の評価関数の許容範囲を適当に定めることにより，誤読率をかなり低く抑えることができることがわかる。

8. むすび

欧文文献をコンピュータにキャラクタコードとして手軽に取り込むプログラムを開発した。しかし筆者らはパターン認識ならびにC言語についてまったくの素人であり，諸兄の御教示を頂ければ幸いである。

9. 謝辞

本報告をまとめるにあたり卒業研究生芦野克巳君の協力を得た。ここに謝意を表します。

付録

1. プログラムの構成

プログラムは以下の2つに分離して作成した。

(1) イメージスキャナを用いて欧文文献を画像としてコンピュータに取り込み，フロッピーディスクに記録するプログラム。

(2) フロッピーディスクに記録された画像を解析してキャラクタコードに変換するプログラム。

(1), (2)のプログラムの実行形式での大きさはそれぞれ25キロバイト，45キロバイトであった。

2. 辞書の構成

辞書は下記の1文字分のデータの繰り返しとして構成した。

辞書の大きさは辞書に記録される文字数ならびに1文字のビットイメージデータの大きさにより変化するが，その上限を150キロバイトとしてプログラムを作成した。ビットイメージデータの大きさは原文のキャラクタの大きさとイメージスキャナの読み取り線密度の設定値により変化する。参考のために本報告で使用した文献の場合イメージスキャナの読み取り線密度は240ライン/インチを使用して，1文字の辞書データの大きさの平均値は約52バイトであった。

バイト	内　容
1	1文字のデータのバイト数
5	キャラクタコード(最後に'0'を配するので実質は4文字分)
2	文字構成グラフィクドット数
1	重心の x 座標 ξ
1	重心の y 座標 η
1	x 方向の2次モーメント ξ_2
1	y 方向の2次モーメント η_2
文字により変化	ビットイメージデータ

3. 誤読例

本文7節で辞書を解析することにより誤読がどのように発生するかを調べたが，そのときに発生する誤読例を文字パターン重ね合わせの誤差パーセントによって分類して示す。

文字パターン重ね合わせの誤差パーセント	誤読例
0～10%	数字の1と英字の1
10～20%	Iと英字の1
20～30%	bと　h cと　j eと　o

誤読は原文の活字のかすれや汚れ，イメージスキャナで画像を取り込むときのノイズ等に影響を受け，誤読を完全になくすことは困難である。特に数字の1と英字の1等を区別することは非常に難しい。