システムソフトウェアと
オペレーティング・システム 68－3
（１９９５．３．２４）

文字フォントをベースとした文字列処理の提案

梅村恭司

ＮＴＴ基礎研究所

多種の文字を扱うシステムの作成経験をふまえ、文字コードをオプショナルとする処理の一つの枠組みを提案する。文字コードが未定義の文字を操作することの困難さを述べ、文字コードの代わりに文字図形の情報を利用する方法を検討した。そのなかでも、コードを用いて実現していた文字の等価性の判定を、図形情報で実現することの妥当性を実験した。

A proposal for string processing based on character fonts

Kyoji Umemura

Nippon Telegraph and Telephone
Basic Research Laboratories

3-1 Morinosato-Wakamiya, Atugi-shi, Kanagawa, 243-01 Japan

We propose a framwork that makes character codes optional. This idea comes from an system that handles many kinds of characters. We describes the difficulty to handle a character whose code is undefined. We try to use character fonts instead of character code. The challenge of our approach is to realize the search mechanizm. We have verified the feasiblity of text search using fonts.

1. はじめに

多種類のメディアの登場で、多種類の情報源が利用できるようになってきた。情報源が多種類になり、各種のリファレンス情報などを扱いはじめると文字コードにない文字を扱う必要性が生じる。そのときには外字を使用する。多くの情報を一つにまとめたシステムを作成した経験では、網羅的な辞書のような情報では外字は必ず現れ、かつ、辞書ごとに外字の種類が異なる。このような場合、外字を含む情報の処理には困難がある。

現状では外字が入ると、情報を伝達したり保存したりする基本的な操作も実現できないことが多い。ネットワークが関係すると状況はさらに困難になる。たとえば、外字を含むような電子メールを送ることはできない。外字を扱う場合には、外字に限って文字図形情報を操作することが一般的である。しかし、このような場合でも検索などの処理には問題がある。辞書などの言語全体に関る情報を扱うとき、外字は例外的な状況ではない。また、電話帳などの網羅的な情報システムを構築するときにも、外字の問題が深刻な問題となる。

そこで、外字の問題を解決することを契機として、文字コードというものを考え直したい。一つのアプローチとして、外字を例外の処理と考えず、すべての情報が外字の枠組みで記述されているという状況を設定する。そして、その状況で文字処理方式を考える。これは、文字コードへの依存をなくす処理を試みることになる。このうち、等価性を判定することが文字コードを使わなくても実現できるかを検討する。

2. 問題設定

文字コードとは、あるコードに対応する文字図形の表である。この表を共有することで情報交換が成立する。この表にない文字が登場すると情報交換ができないことになる。これを解決するために、文字コードを使用する方法を、文字の図形イメージを伝達する上での便法と位置付けた。つまり、情報は文字の図形情報を伝えることを原則とするという立場をとる。このような枠組みで一般的な文字処理が可能かどうかを検討した。この考えの延長上に、清書された文章情報も図形イメージでおこなうという指針も成立し研究もある[2][3]が、ここでは文書は図形とは考えない。情報は二次元のものというより、文字図形表現の一次元の列であるという立場をとる。文字コード独立な情報処理についての問題は重要であり、そこに注意を集中する目的で問題を単純にした。

われわれは、文字の図形情報で文字が定まるという立場を採用した。図形情報を記述する方法はたくさんあるが、ここでは詳細を規定せず、それがビットマップを生成できるとだけ仮定する。そして、文字毎に定まったコードは存在しないと考える。別の言い方をすると、すべての文字を外字と同じ処理することを考えた。

コードがなくても、ビット列としての文字編集はできるが、検索について問題が生じる。検索については、単純にパターンの一致を取ることだけでは、実用的な融通性を持たせることは難しい。すなわち、字体が一ドットでも違うと、字としての同一性を判定できないことになる。この問題の原因は、図形表現の同一性について、ビットマップでの厳格な一致だけでは機能不足であると考える。すなわち、複数のパターンを「同じ」とクラス分けするという機能が、ここでの検討課題である。

3. 文字認識処理の導入

複数の文字の図形表現の間の同一性を判定する処理は文字認識の処理である。文字認識の処理を導入し、情報をメモリへ取り込むときに、同時にパターンをクラス分類し、同時に格納する処理をするとしよう。そのクラス番号を文字コードのよう

同値性の基準にすることができる。クラス番号と文字コードは対応するものではあるが、文字にアプリオリに番号がついているとは異なる。極端な場合、「犬」と「大」と「太」が同じクラス番号であっても検索の上で支障はない。このような場合でも、もともとの図形情報を保存するので、その気になれば区別する方法がある。ここが文字コードだけに依存する方法との違いである。文字認識処理といっても、手書きの文字の認識のようにパターンの揺れが大きいものでなく、また、スキャナーの文字認識のように、雑音が問題となる状況でもない。したがって、本格的な文字認識よりも簡便な方法が適用できる状況である。

図形に基づいた同値関係を定義しているので、図形的に似ているものという類似判定ができる。これには積極的な意味がある。たとえば、現実の文字情報では図形が似ていることによる誤りが混入していることを経験する。辞書システムを構築するとき、ある単語が引けない原因を調査したら、情報が「海千山千」が「海4山4」となっているという誤りがあった。これ以外にも、「ー（長音）」「-（マイナス）」などの混乱は生じやすい。「字として似ている」という条件で検索をする状況が頻発するのではないが、現在の体系にはない機能である。もし、スキャナ入力を元情報とする文字情報が増えるならば、「字として似ている」ものの検索は必要な機能である。実際に、FAXなどのスキャナによる情報で文書データを蓄積する試み[5]も行われている。文字図形よる同一性を設定する処理は、人間の行っている同値性の処理を近似した処理になっていると考えることもできる。日本語の処理において、半角と全角のアルファベットによる混乱や、全角のなかでも句読点による検索の失敗はよく経験する。このような場合でも文字図形に基づく検索に意味がある。

ISOなどで定まりつつある中国語、韓国語、日本語をとりこんだ文字体系[1]でも、同じ源の漢字が複数のコードに割り当てられている。関連するものが2つではなくて3以上であるので、半角と全角より複雑な処理が検索などで必要とされている。それゆえ、文字図形の類似による検索にも意味が生じてくる。

4. フォント情報による実験

簡便な方法でも有効な同値関係が実現できるかを実験した。16bitx16bitの日本語のフォントを使用して、単純なハミング距離を使用した。即ち、パターンの距離は以下の定義した。

距離 = ビットの総数（パターン1 .排他論理和. パターン2）

ここで、X windowシステムで使われているk16と呼ばれているフォントに対して、距離が16（総ビット数の1/16）以下について調べた。6877文字のなかの543文字について距離が16以下のものが存在した。逆に、6334文字にはそのような文字は存在しなかった。その分類は以下のようなものである。

a)「・」「-」「、」など、フォントのドット数が少ないもの
b)「→」「←」などの線分を共有しているもの
c)「ふ」「ぷ」「ぶ」など、濁点がついているもの
d)「目」「且」、「爪」「瓜」など部品の追加があるもの
e)「概」と「慨」など面積の小さい部品が違うもの

また、子細に調査すると、(i)「単純なハミング距離が、濁点の有無などの異差を吸収できる」(ii)「全体の92%の文字はハミング距離16以上離れている」ということが判明した。ハミング距離内に他の文字がはいらない文字を孤立文字と呼ぶことにする。さらに、ハミング距離16で結ばれる文字をまとめてクラスとした。すなわち、このクラス内部と外部との距離は少なくとも17以上は離れているという固まりを作

成した。クラスに入っているかどうかで同値類を構成するが、この同値類は以下のようなものである。

すなわち、$\alpha \equiv \beta \leftrightarrow \exists\omega 1, \exists\omega 2, \ldots \exists\omega n; \alpha \approx \omega 1, \omega 2 \approx \omega 3, \ldots \omega n \approx \beta$

ただし、≈　は与えられたハミング距離以下の関係とする。

このクラス分けでは、クラス内部では距離が離れている可能性があるが、クラス外部との距離が保証されているというのは重要な性質である。すなわち、ハミング距離が近くのものが異なるクラスに入ることはない。このため、クラス分類に基づき検索するときに、距離が近いパターンの検索に失敗することを防止している。いいかえれば、検索の再現率を保証したクラス分けになっている。実験の結果、543文字は183個のクラスを構成し、もともとドット数が少ないパターンを除いて、多くの文字が一つのクラスとなることはなかった。したがってハミング距離を使用すると、濁点などの変形を同一視するサーチが実現でき、かつ、検索のときも十分の能力を保持していることがわかった。ハミング距離は極めて単純な方法であるが、このような単純な方法でも、「コンピュータのために設計されたフォント」という条件と「完全に分別をしなくてもよい」条件があるので、クラス分けに使用することができる。

5. ハミング距離を変えた実験

前節と同じ実験をハミング距離を、0、16、32、48、64と変化させて同じ実験を行なった。孤立字の数と全体の字数の割合とクラスの数と全体の字数の比をグラフにしたのが図1である。

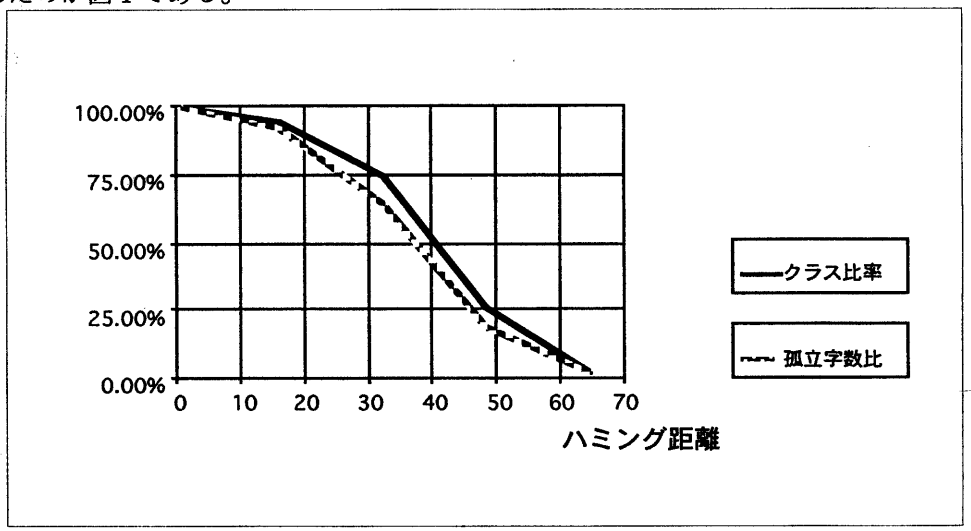

図1　ハミング距離による変化

ハミング距離を大きくしていくと、ドット数の少ない文字が一つの大きなクラスを構成するようになり、48の距離では、仮名とアルファベットの全部が一つのクラスを構成する状態になる。64の距離では有効な検索を行なえるような同値類にはなっていない。実験を行なったk16というフォントでは、16から32の範囲が漢字の細部の差を吸収し、かつ、検索のときに使える範囲である。

ハミング距離は、このような同値類を構成する場合にはもっとも単純な方法であり、文字であるという知識を全く使用していない。それでありながら、濁点の有無などを処理し、かつ、十分に弁別のクラスをもつような同値類を設定できる。ハミング距離は、字としての変形に対して敏感な距離でありながらも、有効な同値類を構成できたのは、文字図形のファミリが一つであったことの影響が多い。現実に多種類のフォントにまたがって類似処理をするには、ハミング距離よりは、字の変形に影響されにくい距離を使う必要がある。それであっても、完全に分離しなくても構わないという立場をとれば、選択の自由度は大きい。

6. 文字図形の前処理による効率化

図形のシーケンスで情報を交換するという原則の上に従ったとしても、よく使う文字図形について前処理をしておくことを考えることができる。存在しうる図形の種類は手書きの文字というわけではないので、種類が限られると考えられる。ビットマップにした図形情報に対してハッシュ関数などを使って判定すると、前処理がなされている確率が大きいと考えられる。このような状況では前処理が意味を持つ。外字が出現たとき、すなわち、前処理がなされていない文字図形が現れたときには、前処理に使用したのと同じ方法で類似性を計算すればよい。そうすれば、外字の処理が継子になることはない。外字にも通常文字と同じ機能を提供できるのが新しい特徴である。

たとえば、同一性を判定するときも、あらかじめやクラス分けを実行しておけば、前処理が済んだ図形であることを検出して、クラス番号をもとめることができる。このクラス番号による検索の方法は文字コードを使用する方法と同じである。

7. 既存の文字コードの利用

提案した手法は、定められた文字コードが存在しなくても処理できる枠組みであるが、既存の文字コードを利用して処理の効率化ができる。そのとき、文字コードは文字図形を示す番号となり、フォントのそのもののビットマップを記述する代わりにコードを使うという位置付けになる。この立場では、文字コードを使用した情報は、ビットマップ情報を辞書をつかって圧縮した形式と解釈できる。すなわち、すべての図形表現がフォント辞書に存在するときに可能な圧縮表現と解釈する。これは、現存の処理と似た意味合いになるが、コードがオプショナルであるのが新しい。このおかげで、フォント辞書にない文字が存在するときでも、統一的な処理が可能となるのである。

8. 文字の順番付け

ソートなどでは文字の順序付けが問題になる。アルファベットでは文字の順番が定まっているが、図形情報からそれと同じ順番を生成することはできない。フォントから計算できる文字の順番付けは、既存のものとは異なる順番になってしまう。これについても、文字図形について文字コードと同じように設定する必要があり、図形情報を使用しても自動的に既存のものと同じ順番を定めることはできない。

しかし、日本語において、漢字の順序付けについて、既存のコードが合意のとれた順番であるとはいえない。ある辞書においては画数などの図形情報にもとづいた編纂がなされている。文字の字種が多い状況では、図形に基づく順番をあらたに設定するのも不自然な状況ではない。たとえば、図形としての複雑さのような尺度を考えよう。外字を含むような状況で情報を整理するには、このような順番が、むしろ、役立つと考える。

9. 異形文字

英語には大文字と小文字の区別がある。漢字においても旧字体と新字体の対応関係がある。このような関係は情報の検索に必要な機能であるが、ここで提案した図

形表現に基づく情報だけでは実現できないものである。大文字と小文字との対応関係は「字」としての対応というよりも言語構造に根差した対応である。これについては、自動的に等価関係を設定するのは無理がある。したがって、個別に同値性を定義するか、歴史的知識や音声表現などに基づく類似性を言語ごとにあたえるという方法、すなわち、図形を用いても異形文字を処理することは不可能で、文字コードと同じ操作をする以外に方法はない。

しかし、文字コードを使用する場合より処理が複雑になることはない。文字コードと同じように文字図形のレベルで関係を設定しておけば、それを同一クラスとして処理することは文字コードの処理と全く同一である。

10. 国際文字コードとの関連

各国の文字コードを区分けして、一つの大きな文字コード表が作成されている[1]。そこでは、中国語の漢字、韓国語の漢字、日本語の漢字が別のコードで記述されている。日本での漢字コードで必要なように、類似した文字が別々のコードで表現されていると、その対応をとるという処理が必要となる。たとえば、半角文字と全角文字という区別で苦労するのと同様な状況が生じそうである。しかも、全角半角のように対応が完全ならば問題ないが、中国語の漢字、韓国語の漢字、日本語の漢字のような場合には、一部に対応がつかないものが含まれるので処理は困難になる。この場合に、表現された図形イメージにもとづく一致という処理が有効であると考える。

ハミング距離のような単純な距離でうまくいくかどうかは検討を要するが、手書きやスキャナの入力の文字認識よりも簡便な文字認識の方法で機能すると推測される。

11. 他の研究との関連

文字情報をビットマップレベルとして扱う提案に関しては原田の研究[4]がある。本研究は対象を文字に制限しているが、等価性について具体的な操作について検討した。また、パターンに関して、具体的な距離空間を導入しているもの原田とは異なる。

12. まとめ、

文字コードを仮定しないで、図形情報を基に文字情報を処理する枠組みを検討した。そこで問題となる文字の同一性について、図形表現による類似という概念を用いることで、文字コードが果たしていた役割の肩代わりができることを述べた。そのうえで、これが、継子になりがちであった外字の処理を統一的にする方法であることを述べた。

謝辞

図形で情報を扱うという発想はＮＴＴ基礎研究所の原田康徳氏、東京大学の萩谷昌巳氏との議論からうまれた。文字認識に関するフレームワークについては、同じくＮＴＴ基礎研究所の石井健一郎氏、萩田紀博氏に手ほどきをうけた。深く感謝したい。

参考文献

[1] Universal Mutiple-Octet Coded Character Set(UCS) Part1: Architecture and Basic Multilingual Plane Unified Ideographic CJK Character (Version 1.0), ISO/IEC DIS 10646-1, 1991

[2] Scott E. Kim: Viewpoint: Toward a Computer for Visual Thinkers, Stanford University, 1988

[3] Yuzuru Tanaka, Kinya Takahashi, Mojitaba Mozaffari: Transmedia Machine, Journal of Information Processing, Vol.12, No.2, 1989.

[4] 原田康徳、尾内理紀夫：ビットマップ通信：ソフトウエア部品間の柔軟な通信をめざして, In SWoPP'94, 情報処理学会、沖縄、1994

[5] Kenneth W. Church, William A. Gale, Jonathan I. Helfman and David D. Lewis: FAX: An Alternative to SGML, In COLING '94 Vol.1 pp. 525-529, Kyoto, 1994