情 報 学 基 礎 57－3
自 然 言 語 処 理 136－3
（2000．3．21）

文字コードに依存しない情報検索の実現

内田 淳† 梅村恭司† 中川 聖一†

uchida@avenue.tutics.tut.ac.jp
umemura@tutics.tut.ac.jp
nakagawa@tutics.tut.ac.jp
† 豊橋技術科学大学 情報工学系
〒441 豊橋市天伯町雲雀ヶ丘1-1

多くの言語，特殊な言語に関連するシステムを構築しようとすると文字コードが問題になる．このため，文字コードの規格が整備される前の言語や，古語などを計算機で処理するのは困難であり，イメージで処理することが行なわれている．我々はイメージを扱うことで，フォント図形を利用した情報検索を提案する．この際，文字列の比較に必要な図形のマッチングに対する計算量が問題であり；我々はテキスト処理を応用することでマッチングの高速化を実現した．

キーワード：文字コード，フォント，情報検索

Character Code Independent Information Retrieval

Jun Uchida† Kyoji Umemura† and Seiichi Nakagawa†

uchida@avenue.tutics.tut.ac.jp
umemura@tutics.tut.ac.jp
nakagawa@tutics.tut.ac.jp
† Department of Information and Computer Sciences,
Toyohashi University of Technology
1-1, Tempaku, Toyohashi, Aichi 441, Japan

When we build systems which treat many or special languages, character code is a source of trouble. It is difficult for computers to process a language before a character encoding is defined. Ancient Japanese language is an example. Those characters are handled as an image in many case.
We propose an information retrieval used font figure, using image format. It is difficult to match the figures efficiently in matching of two strings. We realized high performace matching by adapting a method used in text processing.

key ward : Character Code, Font, Information Retrieval

1. はじめに

多くの言語、特殊な言語に関連するシステム、たとえば、辞書システムを構築しようとすると文字コードが問題になる。それは、文字コードが定まらなければ、一般にコンピュータは、それを文字列として処理することができないからである。このため、文字コードの規格が整備される前の言語や、古語などを計算機で処理するのは困難であり、イメージで処理することが行われている。

このような状況は、正しく文字コードの体系が整備されれば解決できるのは事実であるが、実際の問題として、古語を含む全ての言語が統一したコードに載ることは、その必要性から考えて、実現することは難しいと思える。また、仮に整備されたとしても、経済原則から通常使われない部分のコード系とフォントは、システムに実現されないことは起こる。実際に、アメリカで稼働しているシステムでは、日本語のドキュメントを表示させるのには、いろいろなトラブルが発生する。

本来は、すべての文書の情報をイメージとして処理することによって、文字コードと独立の処理を行うことができるべきであり、それが理想ではあるが、現実に人間が情報を扱うとき、文字という単位が存在する。したがって、情報は文字の列で表現されるという枠組みの中でも、文字コードの体系からくる不都合を解消できれば、システムを構築する自由度は大きくなる。

本研究は、コンピュータの内部で使用されているフォント情報に注目し、その情報に関して検索する処理を考えた。文字列の一致を行なう際の，問題となるその計算量のオーダを改善した.

2. フォントを利用した検索

国際規格で定まっているコード体系とは、数字とフォントの対応表が主要な情報である。つまり、数字に人間が認識できる形を与えていることになる。コード系を使うのは、フォントを直接扱うことが計算機の容量と処理速度の面から実際的でないと考えられたからである。しかしながら、フォント情報をそのままコードとして使うという方法[5]もある。実際に、フォントをコンピュータシステムは処理しており、フォントをコードとするというのは通常より多いビットを使ったコード体系の一つと考えることができる。これを使えば、処理効率と引替えに、コード体系と独立の情報処理システムが可能かと思われるが、一つ問題がある。それは、一致の比較、すなわち、検索（サーチ）処理である。たしかに、入力したものを並び替えて出力するということはできるが、文字列処理の重要な部分である検索が難しくなる。逆に、こがある程度、可能であるということになれば、フォントをそのままコードとして扱うというコード体系を利用することが現実味を帯びてくる。

文字列の位置を見つけることは前方からの一致でも実現できるが，画像の全数に対する比較が必要である．我々は前処理を許すことで，画像中の文字列の位置を画像の全数より少ないオーダで見つけることに目的を設定した.

前処理をすることによって，検索のオーダを対数とするのはテキスト処理では重要である．しかし，これを図形で行う方法は必ずしも明らかでない．そこで，我々はテキスト処理と同様に前処理を行なうことで対数オーダで文字図形を見つけ，検索の高速化を計った.

3. フォントベースの画像を利用したシステム

文字コードを扱うことでは外字や多言語の処理が困難であるためフォントをベースとした文字画像を利用して行なわれる例を説明する.

3.1 フォント画像を利用した外字処理

文字コードが存在しない文字，特に古語等の扱いは外字としてシステム特有のコードポイントに割り当てて利用するのが一般的である．従ってあるシステムで利用している外字が他のシステムでは別のコードポイントに割り当てられていることはしばし起こり，このような外字を含む情報の相互交換は困難である.

また，現在のHTML文書では標準の文字コードしか扱うことしかできないためテキスト中で外字を扱うことができない．そのため，古語を含む人名，地名等の検索が必要なシステムにおいてはこの外字処理が問題になり，フォントから画像ファイルを生成することでネットワーク上で外字の扱いを可能にした歴史資料検索システムも存在する[6].

例えば，「もり・おうがい」という人名を正式な漢字で表現するためには外字処理が必要になる．このような外字の扱いをプログラミング上で可能にするために，以前，我々は画像を直接利用できるプログラミングシステム[7],[8]を作成した(図1).

このように外字処理をシステム独立で行なうアプローチとしてフォントをベースとした画像を扱う方法が存在する.

3.2 フォント画像を利用した多言語処理

多言語システムにおいても，画像を利用して多種の言語，文字の表示を行うアプローチも存在する[1),3)]．これは異なる言語の文字セットを全て図形として同じ枠組で処理できる利点がある.

ISO2022,Unicodeで有名なISO10646のような文字

```
import java.applet.Applet;
import java.awt.*;

public class PrintImage extends Applet {
   Image image;

   public void init() {
      image = getImage(getDocumentBase(),"鷗外");
   }

   public void paint(Graphics g) {
      g.drawImage(image, 0, 0, this);
   }
}
```

図1 フォント画像で外字を処理するシステム例

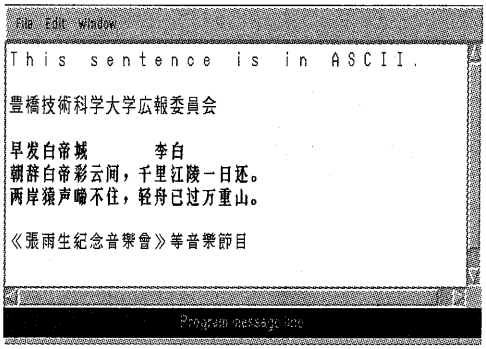

図2 フォントを利用した多言語処理システム

コードを利用して多言語処理を実現しようとする規格も存在するが，言語によっては文字の定義は異なり，その処理方式も異なる．コードと文字が1対1に対応しない場合もあり，これらを同じ枠組で処理することは難しいと考えられている．

アルファベットのように文字とコードが1対1であるものもあれば，インドのデーヴァナーガリ文字のように幾つかの図形で一文字を構成する場合もある．

また，アラビア文字のように右から左に記述され，語中の位置により文字の形が異なるものも存在する．このような処理方式の違いがテキストにおける多言語処理の混乱を招く原因になっている．これらは，表示図形情報，表記方向情報を利用すれば扱うことは可能である[2]．しかし，我々はフォント図形を処理し，検索において，これら全てを同じ枠組で扱うことを考えた．

また，我々の研究室ではフォントを画像化することで多言語処理を行なうプロトタイプシステムを作成済であり，図2に動作例を示す．

4. 図形化テキスト

このように文字コードを利用した不都合を無くすために我々はフォントを直接利用することを考えた．フォントを利用することにより，全ての言語，文字が同等な枠組で扱える．この考えが文字コードに依存しない情報検索の土台となる．

フォントを直接利用するため画像形式の文書を利用する．この画像はフォント図形を組み合わせた図形であり，今後，図形化テキスト(図3)と呼ぶことにする．

つまり，図形化テキストとは図3に示すようにフォント図形を並べて作成し，利用するフォントセットに制限を持たないため，文字セットは混合させられる．そして，本研究ではこの図形化テキストを対象とした検索について考える．利用したフォントセットの大きさは知ることはできるため，幾つかの文字セットが混在するなかで問い合わせた文字列図形の場所を見つけることを検索として行なう．

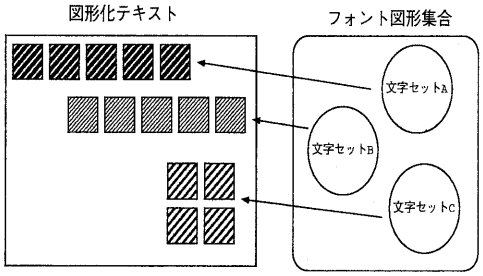

図3 図形化テキスト

5. 計算量の要求

フォントを内部コードとして扱うため，扱うデータの単位が大きくなり，一般的に文字コードを利用することに比べ計算量は大きくなる．今，1byeコードの文字が16×16の大きさで表されていると単純にデータ量は32倍になる．また，画像を扱うことから検索のためには図形同士による一致の比較が必要であり，これは先頭からのマッチングでも可能であるが，この部分がフォントで検索することの現実味を無くしている原因である．

なぜなら，先頭からのマッチングでは，問い合わせた図形の位置を見つけるためには，検索対象の図形化テキスト全ての画素に対する比較が必要になる．従って，全体の計算量は図形化テキストのサイズに比例する．我々は前処理を許して，オーダの低い検索方法が必要と考えた．

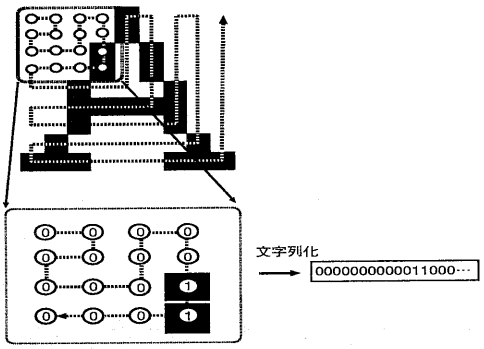

図4　フォント図形の文字列化

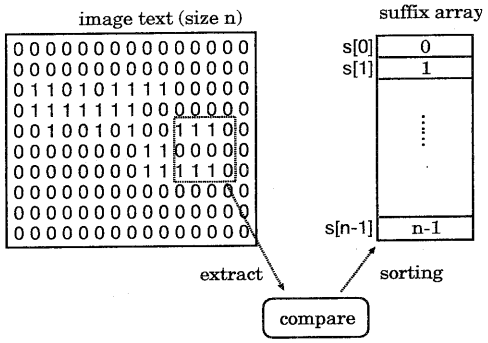

図5　サフィックスソート

6. フォントのコード化

計算機の内部で処理されるコードは 0,1 の文字列である．我々はフォントを内部コードとして扱うために，フォント図形を連続した 0,1 の文字列つまり，一意の順序を持った情報に変換することを行なった．今，図形化テキストには bitmap 形式の二値画像を利用した．そのため，図形化テキストをある順序で走査することにより，白画素であれば 0, 黒画素であれば 1 という法則で文字列に変換した (図 4)．

フォント図形を一定の順序で走査し文字列化する変換テーブルを単一フォント，複数フォント単位で行なう二つのものを用意した．前処理により，query の文字列図形，図形化テキストそれぞれに対し，このテーブルを自動生成する．ここで，テーブルの内容は着目する画素に対するオフセット値である．

7. 対数オーダ図形マッチング

文字列に変換したフォント図形に対して，テキストで利用される高速化処理を適応した．具体的には n-gram 解析で部分文字列の扱いを容易にすることで知られる suffix array を利用する．suffix array は全ての部分文字列に対してインデックスを作成することで部分文字列を参照により扱えるメリットがあり，n-gram 解析のような部分文字列を頻繁に扱う分野では強力な手法として利用されている．[4)]

今, 図形化テキストを大きな 0,1 から成る文字列と考えたとき，文字列化した図形は部分文字列として考えることができる．そこで suffix array と前述したテーブルを扱うことで全ての画素から文字列化した図形が参照できる．我々はフォントの位置を知らずに処理を行なうため，部分文字列の場合と同様に意味のない図形も扱うため，インデックスで参照できる suffix array の効果は高い．

また，画像を 0,1 の文字列に変換したため，これを計算機の内部コードと同じ二進数値として扱うことにより大小関係を考えることができる．そして，この二進数値化した図形と suffix array を利用することによって図形化テキスト中のフォント図形をソートすることができ，そのソートしたフォント図形に対して二分探索が可能となる．テキスト処理ではこれを文字コードを利用して行なうが，我々はフォントをコードとして扱うことで実現した．

7.1 suffix ソートの利用

前述した suffix array と二進コードに変換したフォント図形の大小関係を利用することにより画像をソートすることができる．suffix array は図形化テキストの画素数を n とすると 0 から n-1 の整数を格納しておく．この整数と画素の関係は，画像をラスタ走査した時の画素の並びに対応している．従ってフォント図形をソートするとは画素に対応する suffix array の内容をソートすることになる．

そこで，比較関数では図形化テキスト中の全ての画素について，テーブルで切り出した図形の大小関係を比較する．そして，この比較結果により画素に対応するインデックスをソートする (図 5)．また，このソート部分にはクイックソートを利用した．

ここまで説明してきたフォント図形のソーティングを前処理で行なうことにより，実際にフォント図形のマッチングに二分探索が利用できる状況を作り出すことに成功した．

7.2 二分探索の利用

我々はフォント図形を一意の順序関係を持った情報に変換し，これを前処理でソートすることで二分探索の適応を可能にした．これは，前処理を許し，画像の全数より少ないオーダで文字の位置を見つける我々の目的を満

足している.

この手法は，大きさnの画像に対して一回分のマッチングで単順に$O(\log n)$の計算量しか要さないことを意味する．我々はフォント図形の位置知らずに検索を行なうため，全ての画素について切り出しを行なわなければならない．そのためマッチングの回数が多くなることからマッチングに必要な計算量を対数オーダとし，フォントをコードとした検索を現実的にした．

8. 計 算 量

本節では我々が考案した図形マッチングの計算量について，前方からのマッチングによる方法と比較する．

今，図形化テキストの大きさn，queryの大きさmとし切り出す図形の大きさlとすると，前方からのマッチングでは，単順に全ての図形パターンと比較するためには，

$$O(lmn) \qquad (1)$$

必要とするが，我々の手法では前処理で平均的に$O(n\log n)$必要であるがその後は常に

$$O(ml\log n) \qquad (2)$$

で収まり，図形化テキストの増大に対して比例的には大きくならない．

9. 情報検索への拡張

本節では，ここまで説明してきた図形マッチングを実際に情報検索に応用する方法を紹介する．主な流れは以下の通りである．

(1) まず，図形化テキストと同じ大きさの領域を確保する．
(2) queryから一定の大きさの図形をテーブルにより切り出し，文字列化する．
(3) ソート済の図形化テキストに対して切り出した文字列で二分探索を行なう．
(4) 一致するものが見つかれば，1で確保した領域の一致した場所に対し色づけを行なう．
(5) 2～4をqueryの全ての画素について行なう．
(6) 色づけを行なった領域を利用しスコアを計算する．

以上の手順を図6に示す．3で一致するものを全て探す必要があるが，ソートによって一致した場所の前後に存在することが分かっているため，その前後を探索すれば簡単に見つけることができる．しかし，前方からのマッチングでは画像の全数を探索する必要がある．

9.1 スコアの計算

一つの図形化テキストを扱う場合スコアは図形化テキストを幾つかの領域に分割し，その領域毎に求める．スコアは一致した図形パターンの長さとし，領域のスコアは，

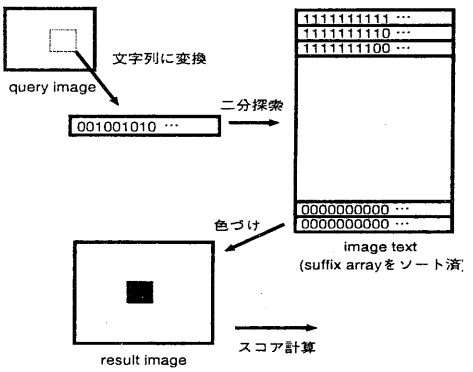

図6 フォント図形を利用した検索の概要

$$SCORE_{region} = \max(L_1, L_2..., L_n) \qquad (3)$$

で求める．ここでLは一致した図形パターンの長さで，nは領域で一致した図形パターンの数である．通常の情報検索ではスコアの加算を行なうが，ここではそれを最大のもので置き換えて近似することを考えた．つまりサブ領域で最大に一致する図形パターンの長さをその領域のスコアとした．これによりスコア計算による計算量を少なくできる．なぜなら部分的に一致するものを考慮すると単順に加算することはできず，重みづけが必要になる．これは一致した図形パターンの大きさや，構成する黒画素の数等を考慮しなければノイズが無視できないため，計算が複雑になることは明らかである．

実際の検索結果では，スコアが上位の領域を幾つか候補としてあげる．複数の図形化テキストを扱う場合は領域に分割せずに，同様な手法でテキスト毎にスコアを求めることで実現できる．

9.2 検 索 例

文字コードを利用することでは難しい検索をフォントを利用することで実現した例を示す．結果はスコアの最も高い領域から前後に2倍の領域で画像を再現したものである．また，結果中の枠はスコアが最も高い場所を示している．

(1) 外字を含んだ文字列の検索

一般にテキストにおける検索では文字列中に外字が含まれると検索することはできない．システムによっては外字の扱いは異なり，同じ文字が全く異なるコードに割り当てられるためである．しか

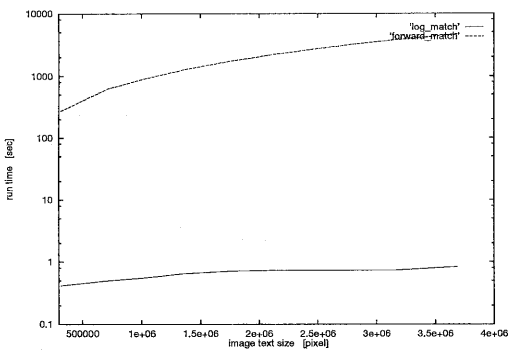

図 9 マッチングの計算時間

し，フォントを直接扱うことで外字を含んだ文字列での検索が可能である (図 7).

(2) 処理方式が異なる文字列を含んだ検索
文字または文字列の処理方式が異なる言語が混在する文書において，文字コードをだけを利用して検索を行なうことはできない．以下の例 (図 8) は日本語とアラビア語という表記方向が異なる言語が混在する文書に対して検索を行なった例である．フォントを利用することで，このような処理方式が異なる言語が混在する文書に対して検索が可能である．

10. マッチングの計算時間

前方からのマッチングで文字列を見つける方法と，我々の考案した二分探索を利用した方法の計算時間を比較し，図 9 に示す．実行環境は Dual PentiumII (433MHz,Memory 512Mbye) である．

グラフからも我々の手法は前方からマッチングを行なう方法に比べ，数百から数千倍の速さで実行することができた．

11. おわりに

我々は文字コードに依存しない情報検索を実現するためにフォント図形を利用した検索技術を提案した．フォント図形を扱うために画像形式の図形化テキストを利用し，このテキストに対して検索を行なった．この際，図形間のマッチングを行なうため計算量が問題になり，我々は前処理でフォント図形を一意の順序が存在する文字列に変換することでソートを行ない，その後，二分探索を利用することでマッチング処理の高速化を行なった．つまり，マッチング処理のオーダを前処理の後で図形化テキストの大きさ N に対して $O(\log N)$ とした．

マッチング部分の高速化を行なうことでフォントをコードとして利用した検索を実際的にし，文字コードに依存しない情報検索を実現した．そして，古語等の外字を含んだ文書，文字列の処理方式がそれぞれ異なる多種の言語を含んだ文書に対する検索を行なった．

現在，我々はマッチング部分に曖昧な機能の実装を行ない，形の類似するフォントでマッチングが行なえる柔軟な検索の実現を計っている．多種のフォントセットが利用できる柔軟な検索を実現することが今後の課題である．

謝　辞

本研究は住友電工及び NTT 未来ねっと研究所との共同研究の成果であり，研究のサポートに深く感謝します．

参考文献

1) A. Maeda, T. Fujita,L.S. Choo, T. Sakaguchi, S. Sugimoto, K. Tabata. A Multilingual Browser for WWW without Preloaded Fonts. In Proceedings of the International Symposium on Digital Libraries 1995 (ISDL'95),pp.269-270,Tsukuba,Japan,1995.
2) 上園 一和,片岡 朋子,筧 捷彦,国際化 Web Browser の設計, 情報処理学会研究会報告書,NL132-8,pp.59-64,1999.
3) 張暘, 梅村 恭司, 文字コード独立の多言語テキストエディタの実装, 情報処理学会第 58 回全国大会論文集 (3),pp.19-20,1999.
4) M. Yamamoto and K. Church, Using Suffix Array to Compute Term Frequency and Document Frequency for All Substring in a Corpus ,Proc.cf the sixth Workshop on very Large Corpora, Ed.Eugene Charniak,Montreal,pp.28-37,1998.
5) 梅村 恭司, 文字フォントをベースとした文字列処理の提案, 情報処理学会研究会報告書,OS68-3,pp.17-23,1995.
6) 桶谷猪 久夫, 歴史史料検索システムにおける外字処理, 情報処理学会研究会報告書, 人文科学とコンピュータ 39-3,pp.55-62,1998.
7) 内田 淳, 梅村 恭司, 図形定数を扱うプリプロセッサの実装, プログラミング・シンポジウム報告集,pp1-6,1999.
8) 内田 淳, 梅村 恭司, 画像を入力とするプログラミングシステム, コンピュータシステム・シンポジウム論文集,pp103-107,1998.

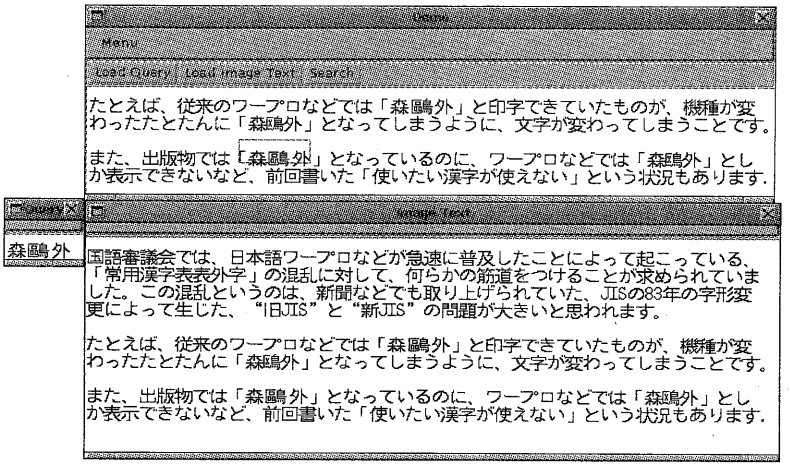

図7　外字を含んだ文字列の検索

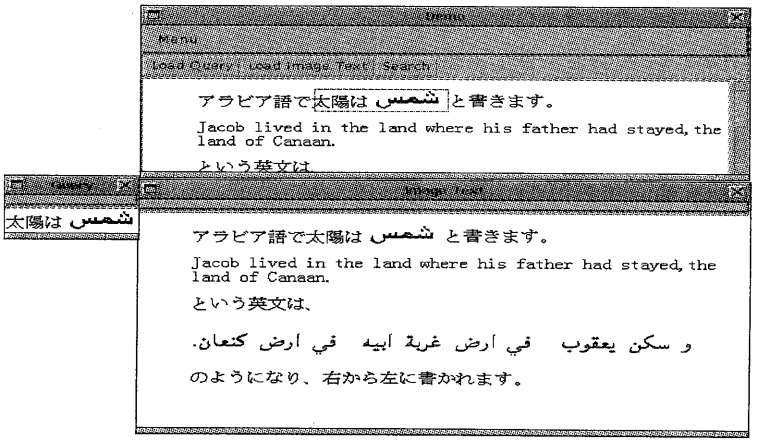

図8　処理方式が異なる文字列を含んだ検索