自 然 言 語 処 理 101−12
（1994. 5. 28）

部分文字列情報の利用による日本語単語の高速検索

中村 俊久　　黒橋 禎夫　　長尾 眞
富士通（株）　　京都大学工学部 電気工学第二教室

要 旨

自然言語処理において、形態素解析はべた書き文からあらゆる可能な単語を抽出しなければならないので、日本語単語の高速辞書検索は必要不可欠なことである。そこで本検索法では、まず登録する単語を部分文字列にわけ、それぞれわけられた文字列に単語の長さ情報をもたせて登録していくことを考えた。これにより辞書検索を部分文字列で行なうことができ、素早く単語の長さ情報を得て目的とする単語までスキップすることができるので、むだな辞書引き動作をなくすことができた。したがって、部分文字列を利用する簡単なシステムとなり、データ構造にはパトリシアを用いてまたデータの検索・挿入・削除にはハッシュ法を用いた高速な検索方法を提案する。なお、本検索法を用いた日本語形態素解析システム（JUMAN-NK）で、従来のシステム (JUMAN-mcc) との解析処理時間を比べると、かなり高速化できたことを確認した。

High-Speed Retrieval of Japanese Word by Use of Partial Character String Information

Toshihisa Nakamura　　Sadao Kurohashi　　Makoto Nagao
Fujitsu Limited　　Department of Electrical Engineering, Kyoto University

Abstract

It is important to realize high-speed retrieval method, because all possible words must be extracted from solid Japanese written sentences in Japanese morphological analysis. We intend to separate words to partial character string, then record them in the dictionary with character length information. It is possible to extract words quickly by using character length information, and remove useless retrieval action. For estimating this method, we build a simple retrieval system which makes use of partial character string information. In this system, we used Patricia as the data structure, and hashing for retrieval, registration, and deletion of words. JUMAN-NK, which uses our hi-speed retrieval system, can do Japanese morphological analysis 2.7 times as fast as the current system JUMAN-mcc.

1 はじめに

辞書検索システムおよび辞書の構成は、自然言語処理の基盤を成すものである。特に日本語文では、分かち書きされていないべた書き入力文などを辞書検索する場合、単語の切れ目が明確でないから検索回数がどうしても多くなる。さらに形態素解析では、文中のあらゆる可能な単語を抽出しなければならないので、むだな辞書引き動作をなくす、単語の抽出方法と高速な検索アルゴリズムが必要となる。

そこでこれまでの検索アルゴリズムとして、以下のものが考えられる。

- B木 (B-tree)
- ハッシュ法 (hashing)
- トライ (trie), パトリシア (patricia)

B木は多量のデータを補助記憶に蓄えてその上で検索を行なうのには適しているが、検索の計算量はデータの数を n とすると $O(\log n)$ である。ハッシュ法は記憶域のむだが多いが $O(1)$ の計算量で検索, 登録, 削除ができる。トライも、記憶域のむだが多いが工夫次第 (パトリシア) で少なくすることができ、計算量は文字列の長さを k としたとき、$O(k)$ で検索、登録などができる。しかし、この検索アルゴリズムも各種の機能について最良といえるようなものはなかなかない。また、それぞれの検索アルゴリズムがどのような機能が優れているかを選ぶ必要がある。

したがって本研究は、データ構造にトライを改良したパトリシアを用いた。そして日本語単語を部分文字列に分け、単語の長さ情報をデータとしてもたせる工夫をした。これにより辞書検索が、入力文の先頭の部分文字列で行なうことが可能となった。また長さ情報によって目的とする単語までスキップすることができ、むだな辞書引きをなくすことができた。辞書の操作は計算量で優れているハッシュ法を用いて、パトリシアとうまく組み合わせることにより、高速な検索を実現している。本稿では、日本語形態素解析システム (JUMAN) で高速検索を確認している。

2 日本語べた書き文に対する辞書検索

まず日本語べた書き文を辞書検索する場合の問題点を示す。またこの問題点に対するこれまでの解決法を示す。

2.1 問題点

日本語文を計算機で処理する場合、まずはじめに文中の単語を認識し、その品詞、活用形などを調べることが必要となる。この処理 (日本語形態素解析) は、一般に次のように行なわれる。

- べた書き入力文がどのような単語に分かれるかわからないので、文中のあらゆる可能な単語を抽出して、それらの中から正しい組合せを取り出す。

日本語文の場合、単語の区切りが明示的に示されていないため、この「文中のあらゆる可能な単語を抽出する」処理をどのように行なうかが大きな問題となる。すなわち文中のどの部分 (文字列) が単語であるかがわからないので、全く工夫をしない場合には、文中のすべての部分文字列についてそれが辞書に登録された単語であるかどうかを辞書引きによって調べなければならない。

例えば、

入力文

- コンピューターに計算をさせてデータを処理する。

という文の場合「コンピューター」「に」「計算」などだけでなく、「に計」、「コンピューターに計算をさせてデータを処理する。」などのあらゆる部分文字列を考えなければならない。

2.2 最長文字列数の利用による解決法

この問題に対する1つの解決法は、先頭の1文字が同じ文字ではじまる見出し語の中の最長文字列数を別に登録しておくことにより、辞書中に存在しない長い語の検索を防ぐという方法である。

例えば先ほどの入力文から「コンピューター」を抽出する場合を考えてみる。

辞書内容

コンピューターネットワーク	⟸ 13
コンピューターシステム	「コ」で始まる見出し語
コンピューター	の中で最大の単語の長さ

先頭の文字が「コ」で始まる見出し語が上の3語である場合、辞書にはその最長文字列数 (13) が登録してある。まず入力文の先頭の1文字「コ」だけを調べて「コ」で始まる見出し語の最長文字列数を求め、入力文からその長さの文字列を切り出す。そして、切り出された文を後方から1文字ずつ左にシフトさせながら辞書と

の照合を行なう。

コンピューターに計算をさせてデータを処理する。

⇓13文字分切り出す

コンピューターに計算をさせ

⇓1文字ずつシフトさせ辞書との照合をする

コンピューターに計算をさ

⇓1文字ずつシフトさせ辞書との照合をする

コンピューターに計算を

⋮1文字ずつシフトさせ辞書との照合をする

コンピューター（一致）

⇓1文字ずつシフトさせ辞書との照合をする

コンピュータ

⋮1文字ずつシフトさせ辞書との照合をする

コ

このように、入力文から先頭の1文字で辞書中の見出し語の最長文字列数を知ることができ、辞書中に存在しない長い語の検索を防いでいる。しかし、この方法には以下のような問題がある。

- 入力文から見出し語が一致するまで1文字ずつシフトさせてむだな辞書検索をしなければならない
- 入力文中に全く見出し語が存在しなくても、むだな辞書検索をしなければならない
- 入力文と辞書の内容が一致した後も、一致した見出し語がなくなるまで辞書検索しなければならない

2.3 トライ、パトリシアによる解決法

トライは木を利用する検索アルゴリズムで、キーの値自身（辞書の場合は見出し語文字列の内部コード）による添字付けによって分岐する枝を選ぶという考え方である。データは葉だけに格納されているので、分岐処理(図1)を施しながら目的のデータに出会うまで繰り返す。トライの操作の計算量は、文字列の長さをkとするとき、$O(k)$の計算量で検索, 登録などの操作ができ、データの個数nによらない。

またトライと同様にキーの値自身によって分岐を決める方法でパトリシアがある。これは値を1ビットずつ取り出してそれによって左右へ分岐を決める。全部のビットを調べるのではむだが多いので、振り分けが実際に起こるビットに対応した節点だけをつくり、そこには値の何番目のビットを調べるかの情報を記録して、0,1の値で左右に振り分ける方法である。

分かち書きされていないべた書き日本語入力文から見出し語を検索する場合、見出し語をこのような木構造辞書に変換すれば入力文の先頭から文字自身をキーの値と

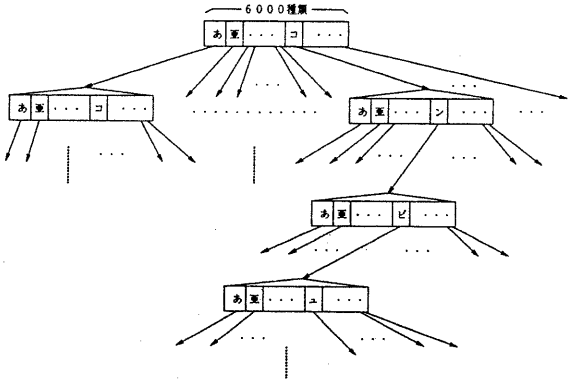

図1: トライ構造

して分岐する枝を選んでいくことで見出し語と一致することができる。また、分岐する枝がなければその時点で検索を止めることができる（その文字列を含むそれ以上長い見出し語がないことがわかる）ので、むだな辞書検索をしなくて済む。

このように木構造の辞書システムを用いる方法は、機能的には日本語べた書き文の検索に適しているが、実際には次のような問題がある。トライの場合は、アルファベット26文字だけに限られている英単語と違い、日本語単語は字種が多いため（横ブロックに約6000種類）辞書容量が爆発してしまう。またパトリシアの場合は、日本語単語を2進数(0,1)で表して辞書化すると1文字あたり16ビットも必要となり、辞書検索回数が増加する問題があり、またシステム自身も複雑となってしまう。

3 部分文字列とハッシュ法を利用した辞書システム

本章では、見出し語の部分文字列情報という新しい考え方を用い、これとハッシュ法を組み合わせることにより疑似的にパトリシアの機能を実現した日本語辞書システムを説明する。

3.1 部分文字列の利用

部分文字列とは、ある文字列を先頭部分から右側に1文字ずつシフトさせて切り出した文字のことである。この部分文字列とそのもとの単語自身の長さ（文字数）を辞書に登録して利用する方法を考えた。

例えば、「コンピューター」を部分文字列にわけて長さ情報をもたせるとすると、「コンピューター」の単語自身の長さは7文字となる。これを単語の部分文字列に

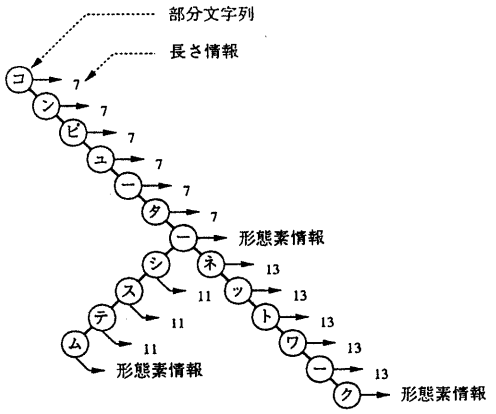

図 2: パトリシア構造の例

与えると次のようになる。

コ	...	7
コン	...	7
コンピ	...	7
コンピュ	...	7
コンピュー	...	7
コンピュータ	...	7
コンピューター	...	(単語自身)

この部分文字列をキーとして辞書に登録する。そして入力文に対して辞書検索を行なう場合は先頭の文字から順に調べる。もしその値が部分文字列の長さ情報だとすると、先頭からその長さ情報だけ文字列をスキップする。また辞書検索できなければその時点で検索を止める。したがって部分文字列を利用する場合、以下の利点がある。

- 入力文の先頭から順に辞書検索するので短いものから調べることができる。
- 部分文字列になっている場合は検索を続けて、なくなればそこで検索を止めることができる。

つまり、このことでトライの基本的な機能を実現することができる。

さらに、単語の部分文字列が長さ情報をもつ利点として次のことがある。

- 単語の先頭部分の部分文字列だけで、その単語自身の長さがわかる。

長い日本語単語などを日本語入力文から抽出する場合、入力文の先頭部分の部分文字列だけで単語自身の長さがわかれば、抽出する単語自身まで素早くスキップすることによって不必要な検索を減らすことができる。つまり、この機能は日本語単語によるパトリシアの実現である。

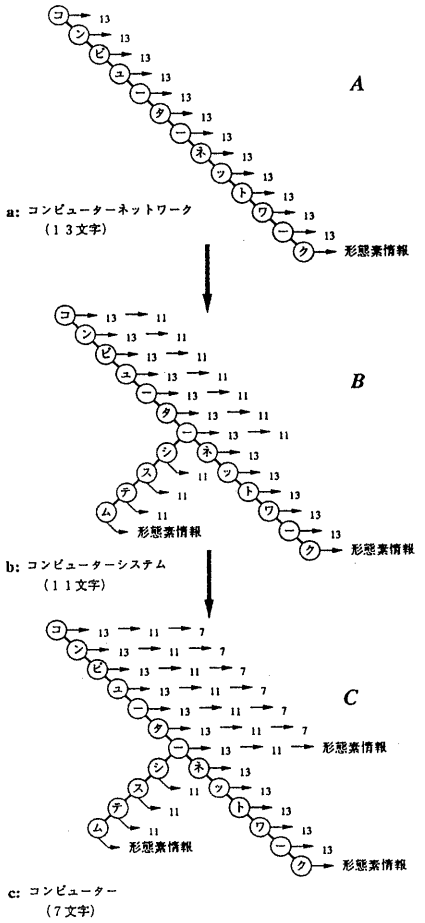

図 3: 単語登録

一方、単語を部分文字列で登録すると次のような問題が考えられる。

- 部分文字列すべてを登録することによる辞書の増大。

しかしこの問題は、パトリシアのキー同士のデータ圧縮特性をうまく用いて解決している。この問題については、4.2節で述べる。

ここで 2.2節の辞書内容の単語をパトリシア構造に部分文字列情報を用いて表した状態を (図 2) で示す。作成した日本語単語の辞書では、部分文字列はトライでありながら、長さ情報からのスキップ動作によってパトリシア構造を実現している。

3.2 ハッシュ法の利用

単語の部分文字列と長さ情報および単語自身の辞書検索システムには、1回の辞書検索が最も高速であるハッシュ法を用いた。ハッシュ法はある決まった範囲の配列をおき、キーの値を引数としてある関数の値を計算する。また関数の値が配列の添字の範囲におさまるようにして、データを格納する。ハッシュ法の検索は、キーの値を引数として関数の値（ハッシュ関数）を計算し、ある特定の配列（ハッシュ表）と結びつけることによってデータを参照する。

ハッシュ法の操作の計算量は、ハッシュ関数の計算も配列の参照も手間には n に無関係であるので、表の検索を $O(1)$ で実現できる。

ハッシュ法の特徴は、配列の大きさ m を大きめにとる必要性から、記憶域のむだが多い。しかし、上手に実現すれば平均して $O(1)$ の計算量で検索、登録、削除のすべてを行なえる最高速の検索アルゴリズムである。

3.3 辞書のアルゴリズム

日本語辞書は、検索キーとなる単語（見出し語）と辞書内容から構成される。辞書のデータ構造には、パトリシアを用いている。ここで日本語形態素解析システム(JUMAN)の辞書を作成した手順を用いて、パトリシアに部分文字列情報をうまく組み合わせたデータ構造を実際に説明する。

3.3.1 単語登録・削除

部分文字列情報を用いた検索法で、単語登録および削除する場合の手順を、(図2)パトリシア構造の例を完成させる手順とそこから単語自身（見出し語）を削除する手順で示す。

[登録手順]

パトリシア構造に部分文字列をキーとして長さ情報を登録する。そして単語自身をキーとして形態素情報を登録する。

[条件]

- **パトリシア構造上で部分文字列の同一キーがある場合は、長さ情報の短いデータを優先的に登録する。**
- **パトリシア構造上で部分文字列のキーと単語自身のキーが同じ場合は、単語自身のデータ（形態素の品詞情報）を優先的に登録する。**

長さ情報を(図3)のように登録する場合は、単語自身の長さを求めて単語の先頭から部分文字列のキーに登録する。

A の状態は、まだ枝が存在しないので単語の先頭から順番に長さ情報を登録しながら「コンピューターネットワーク」の枝をつくり、葉には形態素情報を入れる。

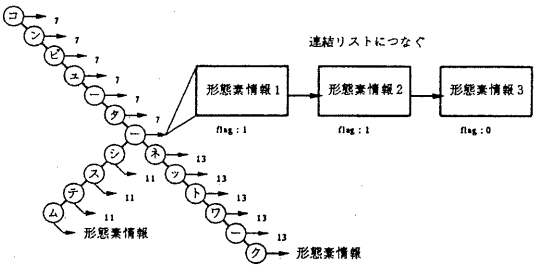

図 4: 登録単語の表記が同じ場合

B の状態は、パトリシア構造で部分文字列の同一キーが生じるため、a,b それぞれの長さ情報のデータを比較しながら短い長さの方を登録して枝をつくり、葉には形態素情報を入れる。

key	*a*の長さ	比較する	*b*の長さ	登録する長さ
「コ」	13	⇔	11	⇒ 11
「コン」	13	⇔	11	⇒ 11
⋮	⋮		⋮	⋮

C の状態も、パトリシア構造で部分文字列の同一キーが生じるため、長さ情報のデータを比較しながら短い長さの方を優先的に登録する。またパトリシア構造上で部分文字列のキーと単語自身のキーの同一キー（「コンピューター」）が生じるので、単語自身のデータ（形態素情報）を優先的に登録する。

- **登録する単語自身（キー）の表記が同じ場合は、データを連結リストにつなぐ。**

ハッシュ法でデータの登録, 検索, 削除を行ってるため、登録する単語自身（キー）が同じでデータ内容が異なる場合は、(図4)のようにキーは同じで異なるデータを連結リストにつなぐ。

またそのデータの形態素情報には、次のデータの有無を示すフラグ（有：1、無：0）が存在する。

[削除手順]

削除するキー

- コンピューター

削除キーを検索キーにして辞書からデータを検索する。もしデータ内容が削除キーのデータ内容と同じ場合は、辞書上の削除キーの形態素情報を長さ情報に変更して再登録する。このとき、長さ情報は、（削除キーの長さ＋1文字）とする

パトリシア構造から見出し語を削除する最も簡単な方法は、(図5)のような場合である。はじめに削除キーを

検索キーにして辞書からデータを検索する。そして、そのデータと削除キーの形態素情報が同じであれば、パトリシア構造上のキー「コンピューター」の形態素情報を長さ情報に変更して再登録する。ここに再登録する長さ情報は、（削除キーの長さ＋１文字）とする。これは、実際のデータを削除せずに部分文字列の長さ情報に変更するだけで手軽にパトリシア構造上で削除ができる。

また形態素データが連結リストでつながれている場合は、削除すべきレコードを通常の連結リストを切り離す手法で削除する。

3.3.2 検索アルゴリズム

パトリシアと部分文字列を用いた検索法で、分かち書きされていない日本語入力文から単語（見出し語）を効率良く抽出する方法を、単語登録の例の辞書を用いて示す。

[検索手順]

入力文の先頭１文字を検索キーとしてパトリシア構造の辞書からデータを検索する。

⇓

辞書から取り出したデータが長さ情報の場合は、検索を開始した入力文の先頭から長さ情報だけスキップして切り出し、それを新しい検索キーにしてデータを検索する。

⇓

辞書から取り出したデータが形態素情報の場合は、形態素の候補として格納する。また他の同じ表記の単語がないかフラグの有無を調べて、有る場合は連結リストをたどりフラグがなくなるまで調べる。無い場合は入力文の検索キーの位置を１文字分右にシフトして、新しい検索キーとしてデータを検索する。

⇓

もし何も検索できなければ、その文字列の検索は終りにして入力文の先頭の位置を１文字シフトしてはじめからもう一度同じ手順で入力文の文字列すべてが終るまで繰り返し行なう。

この検索手順を用いて次の例文からの単語の抽出を (図6) で説明する。

例文

- コンピューターシステムを開発する。

aの状態：例文（入力文）の先頭の１文字「コ」を検索キーとして、パトリシア構造の辞書から長さ情報７を得る。そして入力文の先頭から７文字スキップして「コン

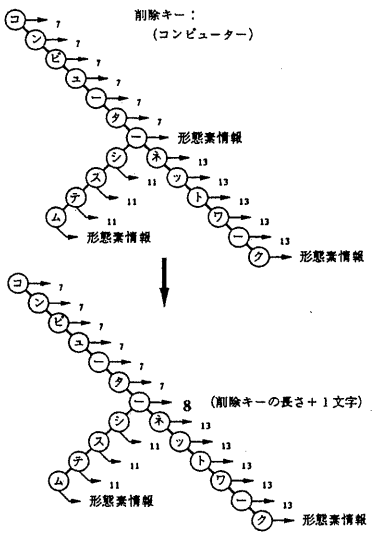

図 5: 単語削除

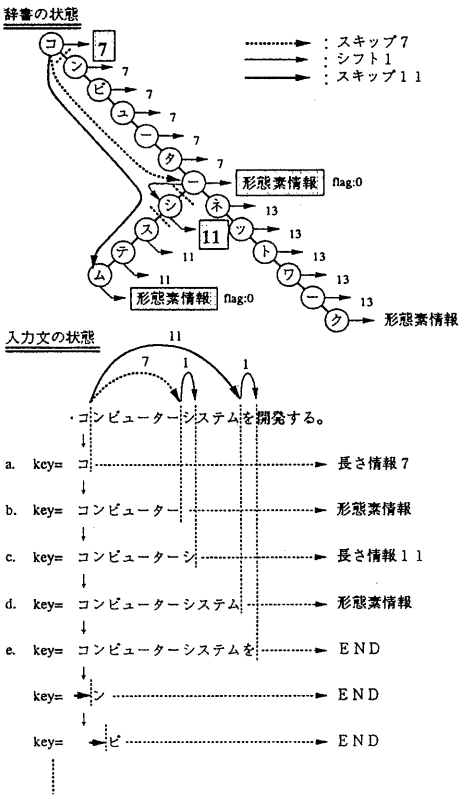

図 6: 単語検索

ビューター」を検索キーとする。

b の状態：「コンピューター」を検索キーとして、辞書検索を行ない形態素情報を得る。形態素情報があった場合は、他に同じ表記の単語がないかフラグを調べ、その値が 0 なので入力文の検索キーの位置を1文字右にシフトして次の検索キーとする。

c の状態：「コンピューターシ」を検索キーとして、辞書検索を行ない長さ情報 11 を得る。そして入力文の先頭から11文字スキップして次の検索キーとする。

d の状態：「コンピューターシステム」を検索キーとして、辞書検索を行ない形態素情報を得る。形態素情報があった場合は、他に同じ表記の単語がないかフラグを調べ、その値が 0 なので入力文の検索キーの位置を1文字右にシフトして次の検索キーとする。

e の状態：「コンピューターシステムを」を検索キーとして、辞書検索を行なって何もデータが検索できなかったのでこの文字列の検索は終り、入力文の先頭の位置を1文字シフトして「ン」を検索キーとして、辞書検索を行なう。また何もデータが検索できなかったのでこの文字列の検索は終り、入力文の先頭の位置を1文字シフトして「ピ」を検索キーとして、辞書検索を行なう。以上のような手順で、入力文の文字列すべてが終るまで繰り返し行なう。

4 実験結果と評価

実験は、朝日新聞の天声人語 1 年分（1989）約 100 万文字と社説 1 年分（1989）約 99 万文字を日本語形態素解析システム（JUMAN-mcc）と本検索法を用いた日本語形態素解析システム（JUMAN-NK）で形態素解析を行なった。

4.1 従来システムとの比較

従来システム (JUMAN-mcc) の検索方法は、2.2節で説明した最長文字列数による方法を少し改良して高速化したものである。この検索方法は、入力文から先頭の 1 文字目だけで辞書に登録してある見出し語の最長文字列数を知るだけでなく、2 文字目,3 文字目まで調べて最長文字列数を知ることができ、1 文字のときよりも辞書との照合回数が減り検索効率を上げている。しかし、2.2節で述べた問題は変らない。従来システムと比較したものを表1に示す。

4.2 形態素解析による性能評価

日本語入力文の形態素解析により性能評価を行なった。実験を行なった計算機は、SUN ワークステーション SS10 である。従来システムでは辞書を補助記憶に B 木構造としてもち、使用頻度の高いものを主記憶上のハッシュテーブルに移動して、2 回目以降の出現はハッシュ法による主記憶上での検索を可能としている。我々の作成したシステム (JUMAN-NK) は、辞書を補助記憶にパトリシア構造のハッシュテーブルとしてもち、ハッシュ法による補助記憶上での検索を可能としている。

表 1: 実験システムの比較

	従来システム (JUMAN-mcc)	本システム (JUMAN-NK)
特徴	・主記憶上に使用頻度の高いデータをハッシュテーブルとしてもち、検索効率を上げているが入力文からの見出し語の抽出にむだがある。	・パトリシアと部分文字列情報によりむだのない辞書引きを可能として検索効率を上げた
入力文から見出し語の抽出法	・入力文の先頭の 1～3 文字を調べ、登録単語の最長文字列数を得て入力文からその数だけ文字列を切り出し、後方から 1 文字ずつ照合して目的のデータを得る。 ※むだな辞書引きがある。	・入力文の先頭の 1 文字からパトリシア構造の辞書と照合して長さ情報からのスキップ操作によりトライの枝をたどり、目的のデータを得る。 ※枝上のスキップによりむだな辞書引きがなく高速である。
辞書検索	・辞書を補助記憶にある B 木構造から使用頻度の高いものを主記憶のハッシュテーブルに移動して検索効率を上げている。	・パトリシアと部分文字列情報を用いた辞書構造をもち、ハッシュ法により登録, 検索, 削除の操作ができる。

日本語入力文のテキストと結果を表 2に示す。朝日新聞の天声人語（1 年分）約 100 万文字を形態素解析するのに、従来システムだと実行時間に 241 分かかったのが本システムだと 93 分で処理でき、約 2.6 倍も高速化できた。また朝日新聞の社説（1 年文）約 99 万文字を形態素解析するのに従来システムだと実行時間に 206 分かかったのが本システムだと 76 分で処理でき、約 2.7 倍の高速化ができたことを確認した。また、実験に使用した主記憶のメモリー値も、従来システムは、解析の終りに 35MB を必要とするのに対して、本システムは、検索以外の形態素解析に必要な値、5MB だけで済んだ。これ

表 2: 実験結果

	従来システム (JUMAN-mcc)	本システム (JUMAN-NK)
朝日新聞 天声人語 1年文 100万文字	実行時間 241(分)	実行時間 93(分)
	使用メモリー 35(MB)	使用メモリー 5(MB)
	使用ディスク 24(MB)	使用ディスク 67(MB)
朝日新聞 社説 1年文 99万文字	実行時間 206(分)	実行時間 76(分)
	使用メモリー 35(MB)	使用メモリー 5(MB)
	使用ディスク 24(MB)	使用ディスク 67(MB)

は、本システムが主記憶上に、ハッシュテーブルとして辞書をもたないためである。しかし、辞書がある補助記憶の値は、従来システムが 24MB なのに対して、本システムは、ハッシュによる辞書システムのため 67MB も必要となった。

本検索法で辞書作成を行なう場合、登録する単語をすべて部分文字列に分けているので、辞書容量が大きくなるのではないかと心配があった。しかし、日本語単語は先頭の何文字かは同じ文字列があるので、単語の先頭部分だけを共有する場合がある。これをパトリシアのデータ構造にすると単語の先頭からの共有部分がデータの圧縮効果をうみ、辞書容量を少なくしている。これを (図 7) で示す。すると 3 つの単語の中で「コンピューター」が共有する文字列になり、パトリシア構造によりデータの圧縮となる。もしパトリシア構造にしなければ、登録する部分文字列数は合計で 31 文字になる。ところが、これをパトリシア構造で登録すると部分文字列数の合計は 17 文字で済む。また、本システムの辞書の見出し語数は 27 万語ある。これを部分文字列にすると合計で 90 万エントリーとなる。しかしこれを本システムで辞書登録すると合計で 36 万エントリーとなり、40% の大きさにまで圧縮され辞書容量がそれほど大きくならなかった。

日本語単語を検索するシステムは、高速検索を実現しようとするとどうしても辞書や検索システムが複雑になりがちである。しかし本システムでは、高速検索のために用いているハッシュを UNIX 等で広く用いられている NDBM システムを用いて手軽なシステムで実現している。これにより、辞書や検索システムが非常にシンプルにでき上がった。

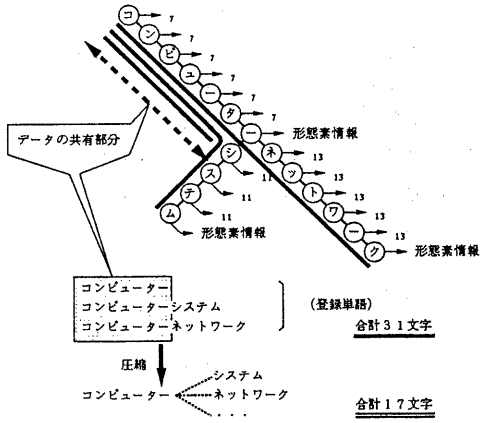

図 7: データの圧縮効果

5 おわりに

分かち書きされていないべた書き日本語入力文から見出し語の抽出を部分文字列情報とハッシュ法を利用して、かなり高速化できることを確認した。本検索法によって、日本語単語のパトリシア機能をもつシンプルでかつ高速な日本語単語の検索システムを実現した。

謝辞

本研究に対してさまざまな助言をして下さった京都大学工学部長尾研究室の皆様に感謝いたします。

参考文献

(1) 石畑, 長尾, 前川, 川合, 所, 米澤 : アルゴリズムとデータ構造, 岩波書店, (1989).

(2) 長尾 : 日本語情報処理, 電子通信学会, (1985).

(3) 松本, 黒橋, 宇津呂, 妙木, 長尾 : 日本語形態素解析システム JUMAN, 京都大学工学部長尾研究室, (1992,1993).