計量国語学第十六巻第五号〔Mathematical Linguistics, vol. 16 no. 5〕1988年

電子国語辞典の構成と実現

今津　英世　（東京工業大学院生，現松下電器産業）
田中　穂積　（東京工業大学）

ディスクリプタ：　電子化辞書　辞書検索　最長一致　トライ構造
辞書の構造化

1. 初めに

1.1. 研究の背景と目的　国語辞典，英和辞典などの辞典を引く作業はデータの検索に他ならない．また，辞書は自然言語のデータベースともいえる．データの蓄積，検索は計算機の得意とするところであるから辞書は計算機上に実現されていることが望ましい．計算機上に実現されていれば紙の辞書では不可能だった様な検索が可能にもなるだろう．また，辞書の電子化は辞書編纂作業の電子化の側面もある．

新明解国語辞典，コンサイス英和辞典，講談社和英辞典，ロングマン現代英英辞典，広辞苑などのテキスト＝データが計算機可読な状態で存在していること，記憶装置が大容量，安価，高速になってきていること，日本語を扱う計算機環境が整備されてきたことなど辞書を計算機上に実現する環境は整いつつあるといえる．本研究の目的は人間が引いて使う辞書の計算機上での効率的，効果的実現である．研究は直接的には株式会社三省堂の新明解国語辞典を計算機上で実現することを目的として行なったが，本研究の諸成果は容易に他の国語辞典や別の種類の辞典（英和辞典，和英辞典など）にも応用可能である．新明解国語辞典のテキスト＝データは電子技術総合研究所推論機構研究室と三省堂との共同作業により計算機可読なものが作成されている（文献２）．辞書データを用いた研究としては文献3, 4, 5 がある．文献３は新明解国語辞典，コンサイス英和辞典，ロングマン現代英英辞典の辞書データを扱った論文の集合である．この中にはコンサイス英和辞典の計算機上

IMAZU Hideyo, TANAKA Hozumi (Tokyo Institute of Technology) — Design and Implementation of Electronic Dictionary of Japanese.

の実現に関する論文がある．文献４はロングマン現代英英辞典を関係データベースに蓄積する研究である．また，文献５は新明解国語辞典の語釈文に関する研究である．文献3, 4, 5 の中には国語辞典の電子化に関するものはない．

1.2. 研究の概要　現在，電子化新明解国語辞典「電明解」がサン＝ワークステーション（OSはUNIX 4.2 bsd 相当）上に実現されており，研究室内で使用されている．辞書データは変更が容易なファイル構造で格納されており，また，変更の為のインターフェースも用意されている．既に研究の過程で数十の誤りを発見，修正している．辞書データの追加，削除は行なっていない．プログラムは全てＣ言語で記述している．研究の過程でのコーディング量は5000 行程度である．

現在可能な辞書引きは「いずれ」，「セレナーデ」といった見出しからの辞書引きのみである．入力された仮名綴りに対して最長一致検索を行なう．辞書引き結果の語釈文は，読み易いようにインデンテーションによる構造化がなされて表示される．このような構造化は全てプログラムを作成して行った．これについては文献９で詳細に説明してあるので参照されたい．

辞書引きの例を以下に示す．下線部が使用者の入力したものである．

（新明解）いずれ〈改行〉

【いずれ】　00イヅレ

1［《何れ］（代）　「どれ・何・どこ・どちら」の意の雅語的表現．

「―か一つを選ぶ・―〔＝どちらも〕劣らぬ・真偽―にもせよ〔＝本当であっても，無くても〕・―アヤメかカキツバタ〔＝二つの物がよく似ていて，区別しにくいたとえ〕」

2（副）　〔どういう経過をたどるにしても，の意〕

1)おそい・速いの違いは有るが，同じ結果になることを表わす．どの道．

「幾ら隠したって―〔＝どうせ，いつかは〕分かることだ」

2)余り遠くない将来において．

「詳しい事は―〔＝近いうち〕お目にかかって申し上げます・―〔＝間もなく〕雨もあがろう」

〔1は，「＜孰れ」とも書く〕

［―にせよ］

1)二つの場合のどちら・を選ぶ（になる）にしても．

2)どうなるとしても.

［一も様〈さま〉03］「おとくい・（お近づき）の皆様」の意の老人語.

最長一致検索をしている為，「せれなーど」と入力しても「セレナーデ」が表示される.

（新明解）せれなーど〈改行〉

【セレナーデ】 03 〔ド Serenade＝愛人の窓の下で聞かせるための，甘美な歌曲〕（小）夜曲.〔セレナードO3は，フランス語形〕

検索対象には「＊」を1つ含めることができる．例えば「＊すい」と入力すると以下の様に表示される.

（新明解）＊すい〈改行〉

【あき やす・い】 04［飽き《易い］（形）どんな事も続けてすることが出来ない性質だ.

1／197

これは「すい」で終わる語は197あってその1つ目を表示していることを表わしている．ここで〈改行〉を入力すると

【い かすい】 02ｷｰ［胃下垂］ 胃が異常にたれさがって，重苦しさを感じる病気.

2／197

と表示される．「あい＊」と入力した時は「あい」で始まる語が，「あ＊ん」と入力した時は「あ」で始まって「ん」で終わる語が表示される.

辞書引きはトライ構造のインデックスにより行なわれ，辞書引きに要するＣＰＵタイムは「＊」を含まない場合は数十ミリ秒以内，「＊」を含む場合は最大で1秒程度である．データの構造化については，ロングマン現代英英辞典のデータはもともと構造化されており，文献3の中の幾つかの研究と文献4はそれをどう加工するかという研究である．これに対して本研究では構造化されていないテキスト＝データをどう構造化するかを問題にしている．テキスト＝データの構造化は文献3にある研究でもコンサイス英和辞典に対してなされているが，データに誤りが多いなどの理由で全体に渡って完成してはいない．実際に製品として売られている規模の大きな電子辞書としては文献6がある．これは本研究の対象としているものとかなり近いものではあるが，文献6の電子辞書は読み取り専用で更新，追加はなく，またデータの構造化もなされていない.

本論文の構成は次の通りである．まず，2章で辞書データそのものについて，その大き

さやデータ化のなされ方などを述べる．次に3, 4, 5 章で電子化新明解国語辞典に関する考察を述べる．3章ではファイル構造，4章では辞書引き法，5章では現在実現されている辞書コマンドの使用法の概要を述べ，6章で結論を述べる．

尚，以下の文中ではオリジナルの（紙の）新明解国語辞典を「紙の新明解」，新明解国語辞典のテキスト＝データを「新明解のデータ」と呼び，とちらと解釈しても問題がない場合は単に「新明解」と呼ぶ．また，本論文中では複数の単語から成る外来語の単語の区切りに「＝」（二重ハイフン）を用いる．

2. 辞書データの諸元

この章では，実現する電子国語辞典のもととなる新明解国語辞典のテキスト＝データ自体について，その構成，データ量，データ化のなされ方などについて述べる．

紙の新明解には所々に枠で囲まれた部分があり，そこで「亜」や「哀」といった漢語の造語成分の説明がされている．新明解のデータはこの造語成分のデータと通常の本文のデータの2つのファイルから成っている．本文データはエントリー数約5万8000，1文字すべて2バイトでエントリーの終端を表わすのに1文字使う形式（シーケンシャル＝ファイル）で格納すると約7.1Mバイト（約370万字）の大きさである．造語成分データは約250kバイトである．本研究で扱っているのは現在のところ本文だけで造語成分は扱っていない．次に新明解国語辞典の辞書エントリーに入っている情報の種類とそのデータ化のなされ方を述べる．以下，説明の為に新明解のデータからの引用例を示す．

> @$_1$いずれ@00$_{*2}$イヅレ$_3$　1$_{*4}$［$_5$《$_6$何れ］（代）$_7$「どれ・何・どこ・どちら」の意の雅語的表現．「$_8$—か一つを選ぶ・—〔＝どちらも〕劣らぬ・真偽—にもせよ〔＝本当であっても，無くても〕・—アヤメかカキツバタ〔＝二つの物がよく似ていて，区別しにくいたとえ〕」＼$_{*9}$2（副）〔どういう経過をたどるにしても，の意〕1)$_{*a}$おそい・速いの違いは有るが，同じ結果になることを表わす．どの道．「幾ら隠したって—〔＝どうせ，いつかは〕分かることだ」＼2）余り遠くない将来において．「詳しい事は—〔近いうち〕お目にかかって申し上げます・—〔＝間もなく〕雨もあがろう」〔$_5$1は，「<$_b$執れ」とも書く〕［$_c$—にせよ］1)$_{*d}$二つの場合のどちら・を選ぶ（になる）にしても．2)どうなるとしても．［$_c$—も様〈さま〉03］「おとくい・（お近づき）の皆様」の意の老人語．

以下，上に引用したものの各部分についての説明を述べる．これは文献2を参照しつつ筆

者の観察の結果を加えたものである.

(1) 見出しは「@」又は「＊」又は「#」で囲まれている.「＊」で囲まれているのは重要語,「#」で囲まれているのは最重要語である.

(2) アクセント番号. この場合は全体でアクセントが共通しているが, 大分類ごとに違うことがあるし, また, 小分類ごとに違うこともある.

(3) 旧仮名遣い.「あおくさい」の旧仮名遣いが「アヲ—」となっている様に現代仮名遣いと同じ部分は省略される.

(4) 語釈大分類.

(5) 最初の部分に[ ]で囲んで正書法が示してある. 最後に〔〕で囲った部分があり, そこにその他の表記についての記述があることがある. 正書法が大分類ごと, 小分類ごとに違うことがあるのはアクセントと同様.

(6) 常用漢字音訓表にない読みを表わす.

(7) 品詞. 活用のある語はその活用型も記してある.

(8) 用例.

(9) 改行. 紙の辞書で行の途中で改行されているところはデータ中にもその旨の情報が入っている. 改行の位置に, はっきりとした規則性はない.

(a) 語釈小分類.

(b) 常用漢字以外の漢字を表わす.

(c) 派生語. 派生語の見出しには正書法, アクセントが記されている.

(d) 派生語にも語釈には大分類, 小分類がある.

(*) 2, 4, 9, aはＪＩＳコードにない特殊なコードを使っている. 上に掲げた引用の中では適宜ＪＩＳにあるコードを使っている.

3. ファイル構造

3.1. ファイル構造の条件　電子辞書のデータを格納するファイル構造には以下のことが要求される.

(1) レコード長可変:　辞書エントリーは最小のものと最大のものとではデータ量に100倍以上の差がある.

(2) 直接アクセス可能:　インデックスを用いた検索を行う為にはレコードの直接アクセスが可能でなければならない.

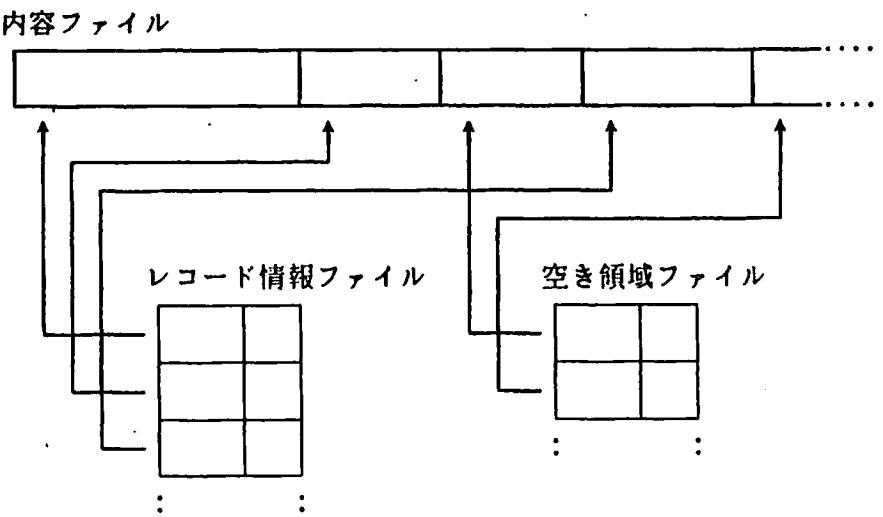
(3) 順次アクセス可能： 辞書を最初から順に走査することもあろうし，また，ある辞書エントリーの前後を見たいこともある．

(4) レコード内容の修正が容易： 新明解のデータには紙の新明解との食い違い（つまりデータ化時の誤り）が見うけられる．また，小分類番号が1つしかないエントリーに小分類番号が記されているといった論理的誤りが紙の新明解にあり，それがデータに踏襲されている例もあった．

尚，現段階ではレコード（辞書エントリー）の修正は行なうが，追加，削除は行なわない．

べた詰めのシーケンシャル＝ファイルはレコード長が可変で，順次アクセス可能であるが，データの大きさが7Mバイト以上となるとデータの修正が非常に困難である．「何バイト目からのレコード」という形でレコードをidentify すれば直接アクセスも可能である．しかし，これでは修正によるデータ量の増減に対処できない．以上を考慮し現在用いているファイル形式について以下に述べる．

3.2. ファイル構造の概要　まず，ファイル構造の概念図を以下に示す．



このファイル構造は「内容ファイル」，「レコード情報ファイル」，「空き領域ファイル」の3つのファイルを使う（参考までに言うとUNIXではファイルの形式はバイト列のみで，システム＝コールのレベルで，任意の位置から任意のバイト数の読み込み／書き込みができる）．

まず，各エントリーの内容は順に内容ファイルに入れる．このとき各エントリーが128

の倍数バイトの領域を占める様に適宜パディングする．各エントリーが占める領域をあるバイト数の整数倍に切り上げるのは修正の結果できる空き領域の管理の手間を考えてのことである．

レコード情報ファイルに各エントリーの内容の開始位置と大きさ（バイト数）を入れる．レコード情報ファイルのレコードをＣ言語の構造体で表わすと以下の様になる（実際にこの型の変数に読み込み，この型の変数から書き込んでいる）．

```
struct zrecinfo {
        long pos;               内容ファイルの開始位置
        unsigned short size;    内容のバイト数
};
```

各辞書レコードは0から始まるレコード番号でindentifyする．エントリーの記録順序は紙の新明解と同じである．

レコード情報ファイルでは各レコードの開始位置とバイト数しか関知していない．レコードの内容は現在は構造化されているが，構造化される前（辞書テキストがベタ詰めになっていた）も全く同じファイル構造を用いていた．現在の構造化は必ずしも十分なものではなく，今後，更に詳細かつ効率の良い構造にしなければならないが，その時もこのファイル構造はそのまま使える．

3.3．　アクセス方法　　レコード番号ｉのレコードを読み込む手順は以下の通りである．レコード情報ファイルの各レコードの大きさはＲバイトであるとする．レコード番号ｉの辞書レコードのレコード情報はレコード情報ファイルの先頭からＲ×ｉバイト目から入っているから，まず，そこからＲバイト読み込み，その情報をもとに内容ファイルから読み込むのである．

辞書を順に読み込むにはレコード情報ファイルを先頭から順に読み込んで，その情報をもとに内容ファイルを読み込めばよい．順次アクセス，レコード番号による直接アクセス，ともに容易にできる．

3.4．　データ量変化を伴うレコードの更新　　レコード内容修正の結果バイト数に増減があった場合の対処は以下の通りである．まず，バイト数が減った時．レコード内容が占める領域は128の倍数に切り上げるから，レコード内容が占めている領域のうち常に何バイトかは空いているのであるが，修正の結果，空きが128バイトを超えた場合は空きの部分のうち128の倍数バイト分を空き領域ファイルに登録する．

次に増えた時，多くの場合はもとの場所に収まるが，収まらなくなることもある．その場合は，まず空き領域ファイルを見て修正結果を収められる空き領域を探す．適当なものがみつかればそこを使う．なければ内容ファイルの最後に新たに領域を確保して使う．修正前にレコードが収められていた領域は空き領域として登録する．

3.5. レコード数増減への対応　このファイル構造では辞書レコードの削除はその辞書レコードに対するレコード情報をレコード情報ファイルから削除することであり，また，辞書レコードの追加は，レコード情報ファイルへのデータの挿入を伴う．従って辞書レコードの追加，削除が行なわれるとレコード番号が変わってしまう．インデックスなどにはレコード番号をレコードのidentifierとして記録するが，レコード番号が変わるとそのインデックスは使えなくなる．レコードを削除した時はレコード情報ファイルの該当部分に削除された旨を書き込むことにすれば，削除に関してはレコード番号の変更を伴わなくすることができるが，追加に関してはこの様なことはできない．

レコード情報ファイルのレコードを以下に示す構造体の様に順序を示すリスクを持つものに変更すれば辞書レコードの追加，削除が可能となる．

```
struct zrecinfoa {
        long prev, next ;       直前，直後の辞書エントリーへのリンク
        long pos ;
        unsigned short size ;
} ;
```

先に述べたファイル構造では辞書の順次アクセスはレコード情報ファイルを「順に」読むことにより行なっている．そこで順に読むのではなく，リンクを辿ることにする訳である．

現段階では辞書エントリーの追加，削除は行なわないからこのファイル構造は用いていない．

4. 見出しによる辞書引き

国語辞書の使い方で最も基本的で，かつ頻繁に行なわれるのは「いやさか」，「ともがら」といった見出しから語釈を引くことである．こういった辞書引きを順次検索により行なうことは辞書のデータ量を考えると現実的ではない．見出しの文字列とレコード番号を対応付けるインデックスを作っておいてそれを用いて検索することになる．インデックスの編成法としてはハッシュを使ったものとトライを使ったものを試みた．ハッシュによる

辞書引きでは，検索用のキー（見出し）を与えて検索場所を計算する関数（ハッシュ関数）を用意して辞書引きする．これは高速ではあるが機能的に不十分な点があり，現在はトライによる辞書引きのみを行なっている．尚，辞書引きでは平仮名，片仮名は区別しない．

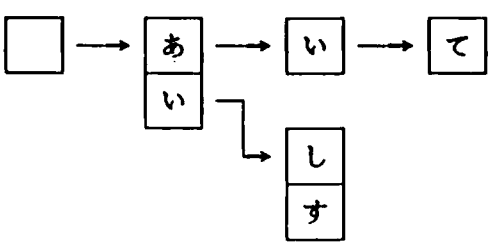
4.1．　ハッシュによる辞書引きの問題点　　ハッシュによる検索の利点はその高速性にある．一方ハッシュによる検索には問題もある．見出し文字列をキーにハッシュを作ったとすると，ある語を引くには見出しと全く同じものを与えないと引けないことである．

新明解では例えば「双眼鏡」は「双眼」の派生語として登録されており，「そうがん」という見出しはあっても「そうがんきょう」という見出しはない．逆に新明解には「躁鬱病」はあっても「躁鬱」はない為，「そうがんきょう」や「そううつ」を引こうとしてもハッシュ表にないため引けない．「セレナード」は「セレナーデ（ドイツ語）」のフランス語形であるが，新明解には「セレナーデ」という見出しはあっても「セレナード」はない為，「セレナード」を引こうとしても何も出力されない．見出しから辞書を引く時に，与えられた見出し文字列がハッシュ表になかった場合，その文字列から最後の１文字を除いた文字列をハッシュ表に探し，それでもなければ更に１文字除いたものについて探し，ということを繰り返すことで，「そうがんきょう」については「そうがん」を引くことができるが，「そううつ」と「セレナード」に関してはそれも通用しない．紙の辞書を使っている時はこういった問題は起きないのであるから，このままでは十分ではない．この問題は次に述べるトライを使った辞書引きにより解決される．

4.2．　トライを用いた辞書引き

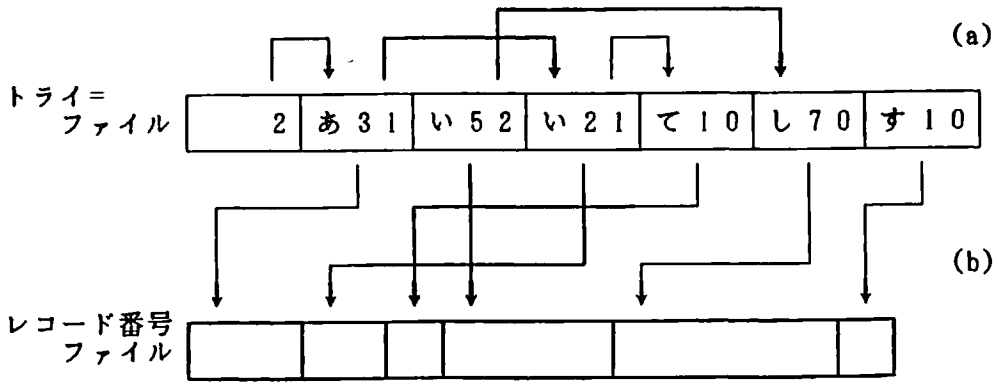
4.2.1．実現　　辞書の見出しの向きのインデックス編成法としてはトライ（trie）構造によるものがある（文献８）．トライを使ったインデックスではそれぞれの見出し文字列についてその見出しを持つ辞書エントリーを全て記録した．これは，ある文字列で見出しが始まる／終わるエントリーを全て探すことを行なう為である．

例として以下の様な非常に小さな辞書について考える．この辞書に対するトライは以下の様になる．一番左の文字が入っていない四角はトライの根を示している．



見出し	同じ見出しを持つ語の数
あ	3
あい	2
あいて	1
い	5
いし	7
いす	1

トライを用いたインデックスの情報はトライ＝ファイルとレコード番号ファイルの２つのファイルに格納する．上に例として挙げた小さな辞書のトライによるインデックスが格納される様子の概念図を以下に示す．



トライ＝ファイルは固定長のレコードの集まりであり，そのレコードは以下の構造体で表わされる．トライの各ノードはトライ＝レコードの集まりで構成される．

```
struct ztrierec {
        char val ;
```

そのトライ＝レコードが担当している文字．

```
        long recnos ;
```

そのトライ＝レコードが表わす文字列を見出しとする辞書レコードのレコード番号の集合がレコード番号ファイルのどこから格納されているか．

上に示したトライ情報格納の概念図の(b)の矢印に相当する．

```
        unsigned short nrecnos ;
```

そのトライ＝レコードが表わす文字列を見出しとする辞書レコードは幾つあるか．

```
        long children ;
```

上に示した概念図の(a)の矢印に相当する．

```
        unsigned char nchildren ;
```

(a)の矢印で指された位置から始まるノードが幾つのレコードから構成されているか．

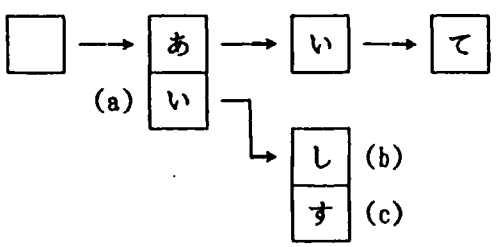
```
} ;
```

たとえばトライ＝ファイル中の［い　５　２］というノードは，「い」という見出しを持つ語の数が５であり，その下に２つの子があることを示している．各ノードの中ではレコードは担当している文字（上の構造体ではval）の昇順に並べられており，トライを探索する時はノードを構成するレコードを一旦全て読み込んで，２分検索する．トライ＝ファイルの最初のレコードはトライの根である．

これまで示してきたのは見出しの文字列を前から見て作ったトライで，いわば「順方向トライ」とでもいうべきものであった．見出し文字列を後ろから見て作る「逆方向トライ」というものも考えられる．我々はトライによるインデックスとして，順方向トライによるものと逆方向トライによるものの両方を作成した．なお現在のトライの実現では辞書レコードの追加，削除に対応できないが，辞書ファイルで用いている手法を用いれば対応することも可能である．

4.2.2．辞書引き手順　　トライによる辞書引きは最長一致により行なう．その手順を以下に示す．

(1) 検索対象文字列に従って順方向トライを辿れるところまで辿る．



トライが上の様になっていたとして検索対象として「いた」が与えられると(a)の所まで到達できる．

(2) 到達した位置から先（その位置を含む）にあるトライ＝レコードに登録されているレコード番号のうち最も小さいものを検索結果とする．上のトライでいうと「(a)から先にあるトライ＝レコード」とは(a)，(b)，(c)である．

「せれなーど」が検索対象として与えられたとすると以下の様になる．ハッシュのところで述べた通り新明解には「セレナーデ」という見出しはあるが，「セレナード」という見出しはない．「せ」，「れ」，「な」，「ー」までは順に辿って来れるが次に「ど」へ行こうとして失敗する．「ー」から先にあるトライ＝レコードに登録されているレコード番号の最小のものは「セレナーデ」に対するものである．従って「せれなーど」により「セレナード」が検索できる．「そうがんきょう」，「そううつ」についても同様に，「双眼」，

「躁鬱病」が検索結果となる．ハッシュ表による辞書引きでの問題がトライでは極自然に解決されている．

最長一致検索の他，ある文字列で見出しが始まる／終わる語の検索がトライを用いて行なえる．ある文字列で見出しが始まる語の検索の手順は以下の通りである．順方向トライのある位置まで辿る．次に，そこを含めてそこから先の全てのトライ＝レコードに登録されている全てのレコード番号の集合を作るのである．そのレコード番号の集合は，その位置が表わす文字列で見出しが始まる辞書エントリーの集合となっている．逆方向トライで同様の集合を作ると，その位置が表わす文字列で見出しが終わる辞書エントリーの集合となる．また，ある文字列Aで始まる語の集合とある文字列Bで終わる語の集合Bの積をとればAで始まりBで終わる語の集合が得られる．

4.2.3. 性能評価　現在の実現環境ではＣＰＵタイム計測の精度は1／60秒である．検索に要するＣＰＵタイムをシステム＝コールにより測定した結果は以下の通りである．

最長一致検索	0～50ミリ秒
「…」で始まる語の検索	100～900ミリ秒
「…」で始まり，「…」で終わる語の検索	300～1300ミリ秒

トライによる検索は先に述べたハッシュ表による検索より時間がかかるのであるが，この様に実用上十分なスピードが得られている．

トライ＝ファイルの大きさは順方向トライで約900Kバイト，逆方向トライで約790Kバイトで，レコード番号ファイルは順方向，逆方向同じ大きさで，約230Kバイトである．インデックスの為に使われる記憶は合計で約2Mバイトである．

4.3. 今後の課題　最長一致による検索は「セレナーデ」を「せれなーど」から引く時の様に入力した仮名綴りの終わりの方に不一致がある場合は有効に働くが，「フィルム」（「ィ」は小さい「イ」）を「ふいるむ」（「い」が小さくない）で引こうとした場合の様に仮名綴りの始めの方に不一致がある場合は良い結果が得られない（「ふいるむ」を引くと「不意」が表示される）．

小さい文字，大きい文字，「づ」と「ず」，オ列長音の「お」と「う」といった入力誤り／表記の揺れに対応する為のヒューリスティックを辞書引きに導入することは今後の課題である．

5. 辞書コマンドの概要

計算機上に実現した新明解国語辞典は「ｊｊｄ」というコマンド名で，引数なしで

ｊｊｄ〈改行〉

という操作で起動する．起動すると「（新明解）」というプロンプトを表示する．ｊｊｄコマンドは自分自身でローマ字仮名変換を行なう．仮名の入力をフロント＝エンド＝プロセッサーに頼るのは操作上好ましくないからである．引きたい語の仮名綴りを入力して〈改行〉を入力すると4.2.2で述べた手順で検索を行ない結果を表示する．

表示は論理構造を反映したインデンテーションを施してなされる．論理構造を反映した体裁のものはベタ詰めのものより格段に見易いのであるが，紙の辞書は紙面の都合でベタ詰めで印刷されている．内容の「見易さ」も電子化の大きなメリットである．表示例については第１章を参照願いたい．表示が１画面以内に収まる時はそのまま表示され，収まらない時は１画面ごとに表示が止まる．何も入力しないで〈改行〉を入力すると直前に表示されたエントリーの次の辞書エントリーが表示され，「－〈改行〉」を入力すると一つ前の辞書エントリーが表示される．以下に示す様に特定の文字列をその中に含む辞書エントリーを検索することも可能である．

（新明解）／船舶

【うき　に】　００［浮（き）荷」

1)〔海に投げ込まれたり，波にさらわれたりして〕海上に漂う，船舶の積み荷．

2)引取人が決まらないうちに，荷主が船積みして売り出す品物．

辞書を引いていてデータの誤りに気が付いた時は一番最近表示されたエントリーを

（新明解）〉ファイル名〈改行〉

とすることによりファイルに書き出すことができる．書き出されたファイルをエディターにより修正した後に

（新明解）〈ファイル名〈改行〉

とすることにより辞書データの修正結果として取り込むことができる．

6. 結び

以上，本論文では電子国語辞典「電明解」に関して，主にファイル構造，見出しによる辞書引きについて論じた．これらは実際に国語辞典を計算機上に実現する過程での考察であるが，英和辞典，和英辞典などにも通ずることがらも多いと思われる．本研究の実現で

は効率に重点を置いており，全データの格納に要する記憶は20Mバイト程度であり，また，ＣＰＵの能力もそれほど要求しない為，本研究の電子辞書をパーソナル＝コンピューターに移植することも十分可能である．

電子国語辞典の実現は，まだその端緒についたばかりの段階であるが，現在の実現でも以下の点で紙の辞書より優れている．

(1) 可能な辞書引きの方法

紙の辞書を人間が引く作業は概ね最長一致検索でカバーされる．「＊」を含む検索は紙の辞書では（事実上）不可能である．

(2) 辞書引きの手間

人間の操作は仮名綴りを入力するだけであるから紙の辞書を引くことに比べて人間の手間は圧倒的に少ない．しかも検索は数十ミリ秒で終わる．

(3) 表示の見易さ

紙の辞書は物理的に実現可能でなければならず，また，頁数が多くなると引く手間が増えるから文字をべた詰めにせざるを得ないが，電子国語辞典ではその様なことはないから見易い表示ができる．

一方劣っている点もなくはない．紙の新明解には図はないが，「清朝体」という書体の例や，音符，温泉のマーク，ＪＩＳマークなどは現れる．新明解のデータではそれぞれに適宜コードが割り当てられているのであるが，こういったものは「図形情報」なのであって，それにコードを割り当てるだけでは十分ではない．紙の辞書には，たとえ図はなくても一部図形情報を含んでいるのである．現在の電子国語辞典の実現では文字情報しか扱うことができず，この点では紙の新明解に比べ劣っているのである．

今後の課題としてはこれまでに述べた辞書引きのヒューリスティック（４章）や構造化に関することの他にも多々考えられる．以下それを列挙する．

(1) 現在のところ辞書を仮名見出しから引くことが実現されているのみでこれだけでは「電子国語辞典ならではの利用形態」とは言えない．派生語，例文，表記からの検索や，参照先，対義語の自動参照なども実現すべきである．表記からの検索には漢字の入力が必要であるが，その際は漢字の読みからだけでなく，画数や部首などから漢字を入力できることが必要である．

(2) 以下の様なエントリーもあるが，

【あく　き】　01［悪鬼］＝〉あっき

「あっき」の方に「あくき」とも表記する旨のことを書いておいてこのエントリー自体は削除し，「あくき」で「あっき」が引ける様にすべきであろう．

(3) 現在はエントリーの追加，削除ができない．電子化国語辞典には辞書編纂の電子化の側面があることを考えると追加，削除はできてしかるべきである．

(4) 辞書データはネットワーク中の１つの計算機に置いておいてそれを他の計算機から使う様なＬＡＮの環境に合わせた実現．

(5) 多くの計算機環境では漢字の入力には漢字入力フロント＝エンド＝プロセッサーが用いられているが，電子新明解国語辞典を包含して，随時それを呼び出せる様なフロント＝エンド＝プロセッサーは非常に便利であろう．実際本システムは２０Ｍバイトの容量を持つハードディスクがあれば，通常のパソコンにのせることができる．日本語ワードプロセッサと併用して使うことが可能だろう．現在その作業を進めているところである．

謝辞

本研究を進めるに当たって有益な助言を下さった田中研究室の皆様に感謝します．また，本研究に用いた新明解国語辞典のデータは非常に高品質で誤りが少なく，本質的でない苦労が非常に少なかったことは特筆に値する．新明解国語辞典を磁気テープ化された元電子技術総合研究所パターン情報部長の淵　一博博士（現ＩＣＯＴ所長），高品質なデータを提供して下さった電子技術総合研究所，株式会社三省堂の皆様に感謝します．

文献

1. 金田一京助他（1982）　新明解国語辞典第３版．株式会社三省堂．
2. 横山晶一，萩野孝野（1984）　国語辞典磁気テープのドキュメント．［電総研彙報］48［8］，692－67．
3. 長尾　真（1982）　昭和55，56年科学研究費補助金　試験研究(1)研究成果報告書．
4. 中村順一，長尾　真（1987）　辞書情報検索ツール　―LDOCE／RDB―．［特定研究言語情報処理の高度化第２回研究発表資料］．
5. 鶴丸弘昭（1987）　日常辞書の機械化とその応用．［第１回「大学と科学」公開シンポジウム日本語の特性と機械翻訳予稿集］，116－129．
6. OASYS 100 GXCD 広辞苑CD－ROMシステム操作説明書（1987）　富士通株式会

社.

7. UNIX Programmer's Manual Chapter 3

8. D. E. Knuth (1973) [The Art of Computer Programming Volume 3], 481－489, Addison－Wesley.

9. 今津英世, 田中穂積 (1988) 電子国語辞典「電明解」の構成と実現. [日本ソフトウェア科学会論理と自然言語研究会ワークショップ].

(1987. 5. 2受付)