第 4 章

構造化文書を用いた日本語文書校正支援

4.1 はじめに

最近になっていくつかの日本語文書校正支援システムやAI辞書を搭載したワードプロセッサなどが試作されているが[2, 3, 6, 22, 24, 57, 64, 65, 78]、ワードプロセッサを使用することによって生じやすくなったといわれる各種の誤りのなかで最も現れやすいかな漢字変換の誤りや表記のゆれの検出・訂正などについて系統的に行なう手法はまだ得られていない。

ＩＣＯＴの石井[2, 3]は、既にできあがった文書をリスト形式に表現し、語用法や文体をチェックしている。そこでは常用漢字表や朝日新聞用語集の知識がＰｒｏｌｏｇで記述されており、漢字の読み、誤りやすい慣用句、言い換えた方がよいことば使いなどについての情報が出力される。

九州大学の牛島ら[6]は辞書も用いず、文法解析も行わずに文章を字面だけで解析し、推敲するツールを開発している。「推敲」というこのシステムでは句点などに注目して文を切り出し、文頭、文末、文長等を表示したり、字種別のＫＷＩＣ (Key Word In Context) を作成したり、字種別に文字列とレコード番号との相互参照表を作成したり、という処理ができるほか、かっこの対応を調べたり、指示代名詞に下線をひいて日本語ラインプリンタに出力させたりすることができる。

（株）日立製作所の絹川[22]は、文書をかな漢字変換レベル、語・句のレベル、文のレベル、段落のレベルの４つにレベル分けし、各レベルにおいて、文章の均一化を図ろうとしている。これは複数の人が分担して１つの文書を作成するような場合、即ち会社等で数人がそれぞれ分担して製品のマニアルを書くというような時に、書き方やことば使いの不統一を防ごうとするものである。曖昧さの少ない表現をするため、また、ことば使いをそろえるために、ワードプロセッサに必要となる機能の検討を行っている。

空閑[24]は、文書の作成から校正、さらに意思決定という過程を、オフィス業務の一部としての文書作成作業と取り扱い、考察している。そして、

1. 単純な入力ミスを判定して形態素をチェックする、

2. 間違いの多い構文パターンとマッチさせて、必須付属語の欠落などを発見する、

3. 誤例辞書を用いて同音異義語の誤りをみつける、

4. 文体の不統一をチェックする、

5. 係り受けの曖昧さをチェックする、

等を行おうとしている。

日本電気（株）の福島ら [64, 65] は、校正だけでなく、論理構造編集、文例提供、文書書き換えなどの機能を提供することにより、文書の作成を支援するシステムの構築をめざしている。

東京大学の建石ら [57] は、辞書を使わず、構文解析も行わずに、校正の対象とする文書と、不適当な表現を正規表現として集めたファイルとをマッチさせることにより、ユーザに注意を促す方法について検討している。また、1つの文の長さの最大値、平均値を用いて、文章の読みやすさの測定も行っている。

また、シャープの安田ら [78] は、新聞記事校正等、日本文訂正作業の省力化をめざして、

1. 表記に関する誤り、

2. 一般常識に関する誤り、

をシステム辞書とユーザ辞書を整備することにより、検出しようとしている。

以上、日本語文書の校正のために行われている研究の一部を紹介したが、これらは一般的な校正技術や語用法の取り扱いについては述べていても、ワードプロセッサの使用によって生じやすい誤変換やミスタイプ等の扱いについては、あまり明らかにしておらず、ワードプロセッサによる誤変換の例を蓄積した辞書の作成といったレベルの誤りを指摘するに留まっている。

4.2 ワードプロセッサで作成された文書の誤り調査

ワードプロセッサで作成された日本語文書中に現れやすい誤りを調査し、分類・整理を行うために、3個のサンプル文書で誤りを調査してみた。文書の大きさは、1つ目（文書A）が、約9380文字、2つ目（文書B）が、約10480文字、そして3つ目（文書C）が、約22000文字で、3文書とも情報工学分野の論文である。文書Aと文書Bは、同一人によって作成されたもので、文書CはAとBを作成した人とは別の人によって作成された。誤りを分類した結果を表 4.1に示す。

表 4.1: ワードプロセッサによって作成された文書に現れた誤り

	文書A	文書B	文書C
文法的な誤り	1	0	0
かな漢字変換の誤り	0	0	14
ミスタイプ	0	2	14
おかしなことば使い	2	0	1
1文が長すぎるもの	2	2	0
句点の打ち方の誤り	1	0	0
長い文節	8	1	0
誤りの数の合計	14	5	29

文書A、文書Bには、ワードプロセッサ独特の誤りともいえるミスタイプや、かな漢字変換の誤りが非常に少なく、ユーザは注意深く文書を入力したと考えられる。一方、その筆者の文章を書くときのスタイルか、句点から句点、読点から読点までが長い文が多いことが分かった。このようにワードプロセッサを「文書清書機」というよりはむしろ、「文書作成機」として使用する場合、ユーザは今まで自分がどのような文書を書いてきたかを確認しながら入力作業を行うため、画面に注目することが多い。従って画面を注視して読み直しをしている間にかな漢字変換の誤りやミスタイプに気がついて修正が行われていると考えられる。しかし、それほど注意しているにもかかわらず、ユーザが発見できない誤りがあった。ある所では数字に「1入力」と算用数字を用い、別の箇所では「二入力」と、数字の表現に漢数字を用いていた。いわゆる、表記のゆれといわれる誤りである。

反面、文書Cでは、文書入力者はブラインド・タッチができるため、原稿のみを見ながらローマ字かな漢字変換入力を行っている。その結果ワードプロセッサの作業中、画面にあまり注目せず、かな漢字変換の誤変換を見過ごしたりすることによる誤りが多い。誤変換の例としては、「文例集」となるべき箇所が「文例終」となっていたような誤りがあった。このユーザはあらかじめ机上で原稿を書いてしまい、あとで文書を清書するためにワードプロセッサを使っている。

これら3つの文書のすべてをとおして、文法的に誤った文は1文しか現れなかった。しかもその誤りは文書修正時に元の文章の一部を残したまま上から書き加えたことによる、いわば誤修正と考えられ、もともとユーザが文法的に誤った日本語を書いたとは考えにくかった。

4.3 日本語文書校正支援の必要性

前節で述べた調査の結果をまとめてみると、ワードプロセッサによって作成された文書には、

1. 文法的な誤りの数は少なく、

2. むしろミスタイプやかな漢字変換の誤りのような、局所的に現れる誤りのほうが多い、

3. かな漢字変換の誤りによる表記のゆれが生じやすい、

4. 長すぎたり同じ言い回しを繰り返したりといったことによる読みにくさが生じやすい、

という性質があると仮定した。

従って、英文法のように性、数の一致があり、比較的構造のはっきりした文法と異なり、日本語の文法をチェックすることによって指摘できる誤りは限られているということができる。例えば、「情報学類」と入力したかったのに、「情報が狂い」と変換されてしまったような誤りや、「ここで履物を脱いで下さい」と「ここでは着物を脱いで下さい」との差異は、文法的なチェックでは発見できない。そこで、ワードプロセッサによって作成された日本語の文書を校正する知識は従来の形態素解析で行われているような文法的なチェックとは異なる独自のものを、実際の文書から、経験的、発見的に獲得する手法が有効である。

筆者らはこの手法を検証するためＣＲＩＴＡＣ(ＣＲＩＴ iquing using ＡＣ cumulated knowledge) と呼ぶ実験システムを試作することにした。

4.4 日本語文書校正支援システムの試作

4.4.1 試作システムのシステム構成

試作システムの構成は図 4.1のようになっており、大きく分けて次の３つの主要部分：１．文書前処理部分、２．ユーザインタフェイス部分、３．校正用知識ベース部分、から成る。

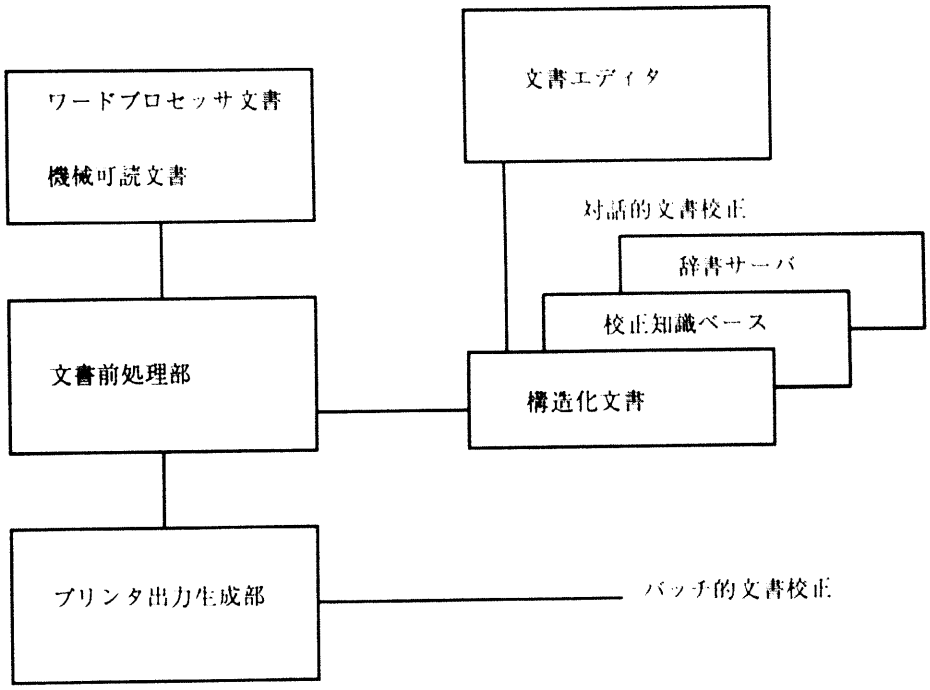

図 4.1: 試作システムのシステム構成

文書前処理部分

文書前処理部分では、べた書きの文書を、

1. 文節区切り、
2. 文節内を自立部と付属部に分離、
3. 漢字複合語を漢字短単位語基に分割して読み付け、
4. 付属語の接続検定、

の順に処理し、表 4.2に概念を示した構造化文書というＰｒｏｌｏｇの節集合に変える。これにより文書校正のための知識はＰｒｏｌｏｇの述語として宣言的に表現可能となる。ワードプロセッサで作成された文書には上記の 4 種の情報を含むものがあるが、文節区切りや漢字短単位語基はワードプロセッサや文書作成者によりばらつきが生じやすく、誤りも多く含むため、ここでは使用しないこととした。

表 4.2: 構造化文書の概念構造

文書 1 原文：今回、我々は文書を高度に構造化する手法を提案する。ここでは文は文節切りされ、単語には読みがふられて構造化文書に格納される。

文書 1 構造化文書：

B_1 : SEG_{11} = 今回	$HEAD_{11}$ = (今回, こんかい, 名詞)
B_1 : SEG_{12} = 我々は	$HEAD_{12}$ = (我々, われわれ, 名詞)
B_1 : SEG_{13} = 文書を	$HEAD_{13}$ = (文書, ぶんしょ, 名詞)
B_1 : SEG_{14} = 高度に	$HEAD_{14}$ = (高度, こうど, 名詞)
B_1 : SEG_{15} = 構造化する	$HEAD_{15}$ = (構造化, こうぞうか, サ変名詞)
B_1 : SEG_{16} = 手法を	$HEAD_{16}$ = (手法, しゅほう, 名詞)
B_1 : SEG_{17} = 提案する	$HEAD_{17}$ = (提案, ていあん, 名詞; サ変名詞)
	$TAIL_{11}$ = ()
	$TAIL_{12}$ = (は, 副助詞)
	$TAIL_{13}$ = (を, 格助詞)
	$TAIL_{14}$ = (に, 格助詞)
	$TAIL_{15}$ = (する, 動詞:終止; 連体)
	$TAIL_{16}$ = (を, 格助詞)
	$TAIL_{17}$ = (する, 動詞:終止; 連体)
B_1 : PUNC(1,1,'、')	
B_1 : PUNC(1,7,'。')	
B_1 : SEG_{21} = ここでは	$HEAD_{21}$ = (ここ, ここ, 名詞)
B_1 : SEG_{22} = 文は	$HEAD_{22}$ = (文, ぶん, 名詞)
B_1 : SEG_{23} = 文節切りされ	$HEAD_{23}$ = (文節・切り, ぶんせつ・きり, 名詞; サ変名詞)
B_1 : SEG_{24} = 単語には	$HEAD_{24}$ = (単語, たんご, 名詞)
B_1 : SEG_{25} = 読みが	$HEAD_{25}$ = (読み, よみ, 名詞)
B_1 : SEG_{26} = ふられて	$HEAD_{26}$ = (ふ, ふ, ラ行 5 段動詞)
B_1 : SEG_{27} = 構造化文書に	$HEAD_{27}$ = (構造化・文書, こうぞうか・ぶんしょ, 名詞・名詞)
B_1 : SEG_{28} = 格納される	$HEAD_{28}$ = (格納, かくのう, 名詞; サ変名詞)
	$TAIL_{21}$ = (で・は, 格助詞・副助詞)
	$TAIL_{22}$ = (は, 副助詞)
	$TAIL_{23}$ = (され, 動詞:連用)
	$TAIL_{24}$ = (に・は, 格助詞・副助詞)
	$TAIL_{25}$ = (が, 格助詞)
	$TAIL_{26}$ = (られ・て, 動詞:連用・接続助詞)
	$TAIL_{27}$ = (に, 格助詞)
	$TAIL_{28}$ = (され・る, 動詞:終止; 連体)
B_1 : PUNC(2,6,'、')	
B_1 : PUNC(2,8,'。')	

＊注＊文書 2 では B_1 がすべて B_2 となる。

ここで構造化文書について説明を行なう。構造化文書とはこの後で詳述する4つの処理によって区切られた、文節を単位とした文書の概念構造で、付加情報のついた文書とも表現することができる。文書を構造化しておくことで、構造化文書の集合を文書データベースとして用いることも可能であり、格納された文書の加工・再利用が容易な価値の高い文書処理を行なうことができる。与えられた文書は通常の文書表示機能に加えて単語の列挙や出現順序、前後の文脈関係まで表示可能で、情報検索・情報抽出の問題点解決の糸口ともなり得る。

文節区切り 入力された文は第3章に示した手法を用いて文節に区切られる。文節区切りのための規則は約３００個あり、ヒューリスティックな知識を基に、どのような文字列が現れた際に、どこに文節区切り記号を挿入するかが書かれている。なお、ここでいう文節は、大河内ら [12] の拡張文節を指す。

この方法による文節区切りの精度は、約97.5 %である [34, 36]。

自立部と付属部に分離 文節単位に辞書を引き、文節内の自立語を取り出す。自立語が辞書に存在するあいだ辞書引きを繰り返し、辞書にある自立語がなくなった時点で、文節内のそれ以降の部分を付属部とする。付属部が存在せず、自立部のみの文節もありうる。

漢字複合語の漢字短単位語基への分割 取り出された自立部の語が漢字複合語の場合には、さらにそれを漢字短単位語基に分割する。ここで、漢字短単位語基とは、漢字複合語を構成する最小単位で、以下に示す「不安定」の例では、『「不」という１文字接頭語、「安定」という２文字漢語がそれぞれ、「不安定」という漢字複合語を構成する漢字短単位語基である』と定義する。

日本語では例えば「不安定」のような語を辞書を用いて機械的に漢字短単位語基に分割しようとする場合、辞書の引き方により、「不安・定」、「不・安定」のように２通り以上の分割可能性がある場合がある。このような曖昧さを解消する方法として、確率的なアプローチをとることとし、確率付きの漢字短単位語基辞書を用いる [46, 70]。漢字複合語の中には「太平洋」「東西南北」のように漢字短単位語基に分割することが困難な語もあるが、提案する方法ではこれらの漢字複合語も確率的に、どのような分割の組み合わせが最も生起しやすいかということを基に、機械的に分割を行うことが可能である。この方法による漢字短単位語基への分割の精度は約９６．５％である。そして最も確からしい分割が決まった時点であらためて漢字短単位語基の読み辞書を引き、漢字に読みが付与される。

付属語の接続検定 付属語オートマトンを用いて付属語の接続検定を行う。付属語列には、例えば、「５段活用・カ行・連用形」というような、活用語の活用型や活用形、「接続助詞」というよ

うな非活用語の品詞を表すカテゴリ番号が付けられる。ここで用いているカテゴリ番号は大河内ら [12] の用いているカテゴリ番号と同じである。

表 4.3: 構造化文書の構成要素

seg(I,J,K,X)	文字列XはＩ番目のパラグラフのＪ番目の文のＫ番目の文節である（以後、Ｉ、Ｊ、Ｋは同じ）。Ｘをこの文節の表記と呼ぶ。
head(I,J,K,U,Y,G,L)	Ｕは文節中の自立語のリスト。Ｙは自立語ごとの読みのリスト、Ｇはその品詞のリスト。 ＬはＵが漢字複合語のときその漢字単位の構成要素のパターンを表す。構成要素のパターンには”Ｐ”、”Ｓ”、”１２”、”１”の４種類がある。”Ｐ”はＰＲＥＦＩＸの”Ｐ”で接頭語、”Ｓ”はＳＵＦＦＩＸの”Ｓ”で接尾語、”１２”は２文字漢字熟語の１文字目と２文字目、”１”は１文字漢字を表す。
tail(I,J,K,V,H)	Ｖは文節中の付属語のリスト。Ｈはリストの最後の付属語の品詞。
punc(I,J,K,D)	この文節に句読点があるとき、Ｄがその文字となる。
sent(I,J,S)	Ｓは文全体の文字列。
para(I,P)	Ｐはパラグラフ全体の文字列。
text(T)	Ｔはテキスト全体の文字列。

以上の処理により、入力された日本語文書は表 4.3に示す７種類のＰｒｏｌｏｇの節に変換される。このようにして、文書中の記号、数式、図以外の構成要素は上記７種の基本的な節から定義できる。ここでは、記号、数式、図については取り扱わないこととする。こうして得られた、Ｐｒｏｌｏｇの節集合によって表された文書を『構造化文書』と呼ぶ。

今回の試作システムでは、ワードプロセッサで作成された、もとの文書の書式に関する情報を取り扱わないことにした。これは、記号、数式、図と同様に書式は文書そのものとは独立に扱うことができ、また、出力媒体や文書の用途に応じて容易に指定したり変更したりできるほうが良いと考えたためである。

ユーザインタフェイス部分

ユーザインタフェイス部分は、構造化文書から外部表現を作成して対話的に文書を校正するための文書エディタと、できあがった文書を一括処理して校正情報や文書処理プログラムへの入力を生成するテキスト・コンパイラ、そして両者とは独立して、オンラインで単語の情報をユーザに

提供する辞書サーバ [94] からなる。それぞれについて順に説明する。

1. 文書エディタ「べた書き表現（以後ソース表現と呼ぶ）」というのは、通常、ワードプロセッサで作成される文書と同じように見え、これは構造化文書の『表記』の部分を逐次連接することによって得られる（図 4.2参照）。

```
CRITAC SOURCE A1 F72 TRUNC=72 SIZE=211 LINE=138 COL=1 ALT=1
 133  ディタから呼び出されるPrologのプログラムとして実現されている。
 134
 135  ソース・エディタではソース表現のうえで語単位、文節単位の挿入・削除・更
 136  新が行え、校正機能は表示画面上で多面単位に利用者に文書の誤りを指摘する
 137  という形になっており、校正メッセージを見ながら対話的に文書を変更できる
 138  。ＫＷＩＣエディタではキーワードや文脈をその文単位に更新できる。複数の
 139  文にまたがる変更や、文の挿入はＫＷＩＣエディタの各行からソース・エディ
 140   タの対応する部分へのスイッチにより、ソース・エディタ上で行うようにな
 141  っている。
 142
**CRITAC**| 1 | 2 | 3 | 4 |**| 5| 6| 7| 8|**| 9| 10| 11| 12|
 PROOF  Srce Next Quit Expl  SCHG  ? Back Forw    = Home Word KWIC
====>
```

図 4.2: ソース表現

「キーワード／文脈表現（以後ＫＷＩＣ表現と呼ぶ）」は、構造化文書の各自立語をキーワードとしてその読みの順番や漢字コード順等によって並べ変え、前後の文脈と共に表示したものである。漢字複合語の場合は構成要素のうち、２文字漢字熟語とそれに伴う前後の接頭語接尾語を含んだ部分を１つの自立語と定義し、１つの自立語がそれぞれキーワードとなる。従って自立部と付属部に分離する手順の段階でＮ個の自立語が辞書から抽出されたとする (これを構造化文書がＮ個の自立語をもつということにする) と、ＫＷＩＣ表現はＮ個の行からなる（図 4.3参照）。

CRITAC KWIC A1 F116 TRUNC=116 SIZE=1138 LINE=520 COL=1 ALT=0

510	の校正機能は主として仮名	漢字	変換の誤変換と表記のゆ
511	現在のところは	漢字	の基本語約３００００語
512	自立語）を中心として仮名	漢字	変換の誤変換や表記のゆ
513	リスト形式に変換し、常用	漢字表	と朝日新聞の用語週から
514	よって、指摘できる誤りは	限	られていると考え、文書
515	書の校正における技術的	課題	は最近の校正に関する
516	い日本語文章の校正は近年	急速	に機械化されるに至った
517	名漢字変換の誤りのような	局所的	に現れる誤りのほうが多
518	語文章の作成は近年急速に	機械化	されるに至ったが、英文
519	ーワードを調べて行く校正	規則	が作られている。
520	校正	規則	はＰｒｏｌｏｇの
521	あわせて現在２０個の校正	規則	が校正知識ベースとして
522	校正	規則	については特に［５］で
523	作成したり印刷したりする	昨日	は向上しつつあるようで
524	英文では既にパソコン上で	機能	するスペル・チェックさ
525	よるもので、英文並の校正	機能	を実用かするためには形
526	校正	機能	はこのエディタから呼び
527	・削除・更新が行え、校正	機能	は表示画面上で画面単位
528	ＫＷＩＣエディタの校正	機能	は主として仮名漢字変換

CRITAC	1	2	3	4	**	5	6	7	8	**	9	10	11	12
KWIC	prf	Yomi	Quit	Expl		SCHG	?	Back	Forw		=	Home	Word	Srce

図 4.3: KWIC 表現

文書エディタではこれら２つの外部表現をとおして、対話的に文書を校正できる。外部表現上の高レベル操作や、文書への更新操作を構造化文書上に反映する機能が実現されていないが、ソース表現上では語単位、文節単位の挿入・削除・更新を行うことを支援する。そして校正機能は表示画面上で、現在表示されている画面を単位としてユーザに誤りを指摘する。編集機能としては、画面最下行のコマンド行から編集コマンドが入力できることと、画面上、最下行に示してあるように、キーボードの１２個のキーに割り当てられているファンクション・キー(ＰＦキー)の機能が利用できる。図 4.2のソース表現では、ＰＦキーの設定は表 4.4のようになっている。

表 4.4: ソース表現における PF キーの設定

PF1:	校正機能の適用／解除	PF2:	次の下線部へのカーソル移動
PF3:	システム終了	PF4:	校正メッセージの表示
PF5:	文字列の書き換え	PF6:	直前のコマンドの表示
PF7:	前の画面に移動	PF8:	次の画面に移動
PF9:	直前のコマンドの実行	PF10:	コマンド行と画面間のカーソルの切り替え
PF11:	カーソル位置の文節情報の表示	PF12:	KWIC 表現への切り替え

このうち、ＰＦ１、ＰＦ２、ＰＦ４、ＰＦ１１、ＰＦ１２の５つの機能が、文書校正に関する特別な機能である。

一方、図 4.3のＫＷＩＣ表現ではＰＦキーの設定は、表 4.5のようになっており、ＫＷＩＣ表現における文書校正のための特別な機能は、ＰＦ１、ＰＦ２、ＰＦ４、ＰＦ１１、ＰＦ１２の５つである。

表 4.5: KWIC 表現における PF キーの設定

PF1:	校正規則の適用／解除	PF2:	キーワードの読みの表示
PF3:	システム終了	PF4:	校正メッセージの表示
PF5:	文字列の書き換え	PF6:	直前のコマンドの表示
PF7:	前の画面に移動	PF8:	次の画面に移動
PF9:	直前のコマンドの実行	PF10:	コマンド行と画面間のカーソルの移動
PF11:	カーソルのある行のキーワードの同音異義語の表示	PF12:	ソース表現に切り替え

ＫＷＩＣ表現上では、キーワードや文脈を、その文を単位として更新することを考えている。複数の文章にまたがる変更や、文の挿入は、ＫＷＩＣ表現の各行から、ソース表現の対応する文章へ切り替えて、ソース表現上で行う。ＫＷＩＣ表現の校正機能で特徴的なことは、漢字かな混じり語とカタカナ語等、字種は異なるが、同じ読みをもつ語の、表記のゆれを検出可能なことである。ＫＷＩＣ表現上でキーワードを読み順に並べると、同音異義語や表記のゆれをもつ語は必ず隣接して現れる。例えば、「割り当て」と「割当て」、「Ｐｒｏｌｏｇ」と「ＰＲＯＬＯＧ」は並んで表示され、「表記のゆれ」の発見が容易である。この特徴を生かして、隣り合ったキーワードの読みと表記を比較するといった校正規則を定義した。具体的には図 4.3の５２４行目と５２５行目を比較して、この文書中１回だけ用いられている「昨日」が「機能」のかな漢字変換の誤りの可能性が高いことが容易に発見できる。

ソース表現からＫＷＩＣ表現に切り替えるときにカーソルが自立語に位置付けられていれば、ＫＷＩＣ表現において画面はその自立語をキーワードとして画面中央に表示する。カーソルが自立語に位置付けられていない時に切り替えると、ＫＷＩＣ表現は自立語の「読み」の昇順に先頭の語をキーワードとする文を現在行として表示する。

逆にＫＷＩＣ表現からソース表現に移るときにはカーソルが位置付けられている文が画面中央の現在行となる。カーソルがコマンド行にある場合にソース表現に切り替えると、ソース表現はその文書の最初の段落を画面の第１行目として表示する。カーソルが自立語に位置付けられている場合には、その自立語に注目したままで、文書の表示方法を変えながら校正を進めることができる。さらに、図 4.4に示すように、ＫＷＩＣ表現ではキーワードの前後の文脈のサイズを変え、キーワードの重複を取り除くことができる。従って、文脈のサイズを０にすることで、ＫＷＩＣ表現は異なり語のリストとしても利用でき、キーワードすなわち自立語を中心としたトップ・ダウン的な校正法を可能とする。

```
CRITAC KWIC A1 F116 TRUNC=116 SIZE=540 LINE=332 COL=1 ALT=0

 322                                最大
 323                                作成
 324                                作業
 325                                四
 326                                社用
 327                                終止形
 328                                修正
 329                                出力
 330                                手法
 331                                手段
 332                                少数
 333                                商用
 334                                書式
 335                                処理
 336                                支援
 337                                試作
 338                                従
 339                                指定
 340                                仕手折
**CRITAC**| 1 | 2 | 3 | 4 |**| 5| 6| 7| 8|**| 9| 10| 11| 12|
 KWIC    Prf   Yomi Quit Expl    SCHG  ? Back Forw     = Home Word Srce
 ====>
```

図 4.4: KWIC 表現を利用した異なり語の表示

2. テキスト・コンパイラ

ユーザインタフェイスの提供する2番目の機能としてテキスト・コンパイラがある(バッチ的に処理を行なう、という意味でＢＡＴＣＨ版試作システムすなわちＢＣＲＩＴＡＣと呼ぶ)。ＢＣＲＩＴＡＣは文書のソース表現と校正メッセージ、構造化文書の出力に利用する。これはちょうど通常のコンパイラがプログラムからエラー・メッセージとオブジェクト・コードを出力するのに類似している。ここではオブジェクト・コードに相当する構造化文書の出力は、

‥(単語の区切り)＜単語＞＜読み＞(単語の区切り)‥

という形式を考えており、区切り記号や読みの有無はオプションで指定できる。このような出力は例えば、日本語文書の音声化、点字化や検索システムへの入力として利用できる。ＢＣＲＩＴＡＣではさらに、文書の統計的情報等の付加情報を合わせて出力できるようになっている。付加情報の種類としては、文献[５]で考察しているようなものが有用であると考えている。ＢＣＲＩＴＡＣの出力結果を図 4.5から図 4.6に示す。

```
1.Source file

 1    1．まえがき
 2
 3    ワードプロセッサ等により作成された計算機上の日本語文書の校正は、
 4    なって多数の研究が報告され始めた分野であり，［1］［2］［3］
 5    １４、今後の知的ワードプロセッサあるいはオフィス・システムにと
 6    重要な機能であるといえる。これらの研究には対話的校正処理を中心と
 7    正環境の実現と、いわゆるＷｒｉｔｅｒ’ｓ　ｗｏｒｋｂｅｎｃ
 8    ような各種の文書解析ツールの実現とが含まれる。
 9
10    前者の例としては、ＥＰＩＳＴＬＥシステム［１０］のように、実時間
11    解析モジュールと高機能のエディタ(日本語の場合はワードプロセッサ
12    トウェア )とを統合したものとなり、後者は目的に応じた解析を実行し
13    をファイルに出力するプログラム群［９］がある。
```

図 4.5: テキスト・コンパイラ:BCRITAC の出力結果 -1-

図 4.5には入力文書のソース表現が出力されている様子を示している。ソース表現は、文書エディタと同様に、べた書きの文書表現に行番号をつけて表示され、文書中の書式情報は取り除かれ、禁則処理等も行なわれていない。ＢＣＲＩＴＡＣでは、ソース表現は校正メッセージで指摘される行番号と文節を確認するために用いる。

```
*ERROR 31 on line 23 position 6  :「機械翻訳システム野ためには」
   適当でない漢字が使われています。
   誤変換ではありませんか？

*ERROR 33 on line 40 position 35 :「実用かと」
   未変換の可能性はありませんか？
   この「か」は漢字で書かれるはずではありませんか？

*ERROR 17 on line 24 position 28 :「よって」
   １つの文の中に同じ言葉が繰り返し使われています。
   他の言い回しができませんか？
```

図 4.6: テキスト・コンパイラ:BCRITAC の出力結果 -2-

図 4.6では、校正メッセージが表示されている。校正メッセージは校正規則に対応した警告文と、その対象となった文節、およびその文節の、ソース表現上の行番号と開始桁を示す。ＫＷＩＣ規則によって指摘された校正メッセージも、ソース表現上の文節に対する警告に翻訳されて出力される。

表 4.6では、入力文書のＫＷＩＣ表現が示される。ＢＣＲＩＴＡＣにおけるＫＷＩＣ表現は文書エディタが提供するＫＷＩＣ表現とは一部異なっており、行頭にキーワードの「読み」が

表 4.6: テキスト・コンパイラ:BCRITAC の出力結果 -3-

かんようく	、漢字の読み、誤りやすい	慣用句	、言い換
かんじ	をＰｒｏｌｏｇで記述し、	漢字	の読み
かんじ	もいえるミスタイプや仮名	漢字	変換の誤
かんじ	より注意しないため、仮名	漢字	変換の誤
かんじ	)むしろミスタイプや仮名	漢字	変換の誤
かんじひょう	リスト形式に変換し、常用	漢字表	と朝日新
きょくしょてき	名漢字変換の誤りのような	局所的	に現れる
きじゅつ	知識などをＰｒｏｌｏｇで	記述	し、漢字
くてん	を書くときのスタイルか、	句点	から句点
くてん	きのスタイルか、句点から	句点	、読点か
くりかえ	)長すぎたり同じ言回しを	繰返	し用いた
くぎ	本語の文書では単語が陽に	区切	られてい
くぎ	書を高度に構造化し、文節	区切	りや、読
けっか	現れやすい誤りを検討した	結果	、校正に
けっか	誤りを分類した	結果	を次に
けっか	この調査の	結果	、ワード
けいけんてき	を実際の文書にあたって、	経験的	、発見的
けんきゅう	性質を考慮した校正方法の	研究	が重要と
けんとう	に改良する方式などが試作	検討	されつつ
けんとう	た文書に現れやすい誤りを	検討	した結果
こうかてき	リスティックにたよる方が	効果的	であると
こうせい	日本語の性質を考慮した	校正	方法の研

陽に示されている。また、オプションとして、キーワードの並べ方を指定できる。並べ方は、自立語の読みと表記による順序づけと、キーワード、キーワードに前接する語、キーワードに後接する語を読みの降順に並べるものとがあり、この際、ＫＷＩＣ規則による同音異義語の誤変換や表記のゆれの可能性のある箇所についての校正メッセージを併せて出力することが可能である。

ユーザは誤変換を見つけたり、言い回しを変えた方がよいような表現について、ＢＣＲＩＴＡＣの出力を見ながら推敲したりすることができる。

3. 辞書サーバ

辞書サーバはユーザがエディタから対話的に各種の辞書情報を検索できるように、関係データベースシステムＳＱＬ／ＤＳ [87] の関係として国語辞書を管理したものである。現在のところ２文字漢字熟語の基本語約３００００語が格納されており、正書、読み、品詞の３つの属性をもっている（図 **4.7**参照）。ユーザは校正機能を動作させることにより指摘された漢字複合語の誤りを、この辞書を用いて確認したりできる。更に例えばこの辞書に『意味分類』という関係を追加すれば、言い回しを変えたいときに辞書から同義語を検索することができる。また構造化文書は容易に関係データベースとして格納できる構成のため、完成した文書を文書データベースとして辞書と同様に管理することにより、オンラインの単語用例検索や一般的な文書検索のように広い範囲の文書情報の要求や文書の蓄積による大容量化にも対応できる。辞書サーバへのアクセスは、あらかじめ用意された同音語等の検索以外にも、質問言語ＳＱＬを用いてユーザが動的に表現できる。これは通常の電子辞書と異なる、関係データベースシステムのもつ大きな利点である。また、辞書の属性を増やすことにより、さらに文書校正上有効な情報をユーザに提供することが可能となる。例えば、先に示したような意味分類の情報を用いれば、類義語検索や反対語表示などが考えられる。

```
CRITAC KWIC A1 F136 TRUNC=136 SIZE=290 LINE= 62 COL=32 ALT=0
-- CRITAC オンライン辞書サーバ ----------------------------------------
 更生[こうせい                                ]【サ変名詞・名詞】
 公正[こうせい                                ]【形動・名詞】
 硬性[こうせい                                ]【名詞】
 恒星[こうせい                                ]【名詞】
 抗生[こうせい                                ]【名詞】
 構成[こうせい                                ]【サ変名詞・名詞】
 攻勢[こうせい                                ]【名詞】
 後世[こうせい                                ]【名詞】
 校正[こうせい                                ]【サ変名詞・名詞】
 鋼性[こうせい                                ]【名詞】
 厚生[こうせい                                ]【名詞】
 高声[こうせい                                ]【名詞】
-----------------------------------------------------------------------
 62          最近になって、文書 構成          のツールにより、すでに出
 63   しかし出来上がった文書を 校正           ・推敲したりする補助手段
 64       本報告では日本語文書 校正           を支援するための試作シス
 65   二つの外部表現および文書 校正           環境を実現した。
 66   独自性を扱う上で効果的な 校正           環境を実現するものと考え
 67              ＣＲＩＴＡＣの 校正          ちしっきはソース表現上で
**CRITAC**| 1 | 2 | 3 | 4 |**| 5 | 6 | 7 | 8 |**| 9 | 10 | 11 | 12 |
 KWIC   Prf   Yomi Quit Expl   SCHG  ? Back Forw    = Home Word Srce
 ====>
```

図 4.7: 同音語の出力（ＳＱＬ／ＤＳ）

4.4.2 校正用知識ベース

校正支援試作システムの校正知識はソース表現上で使用されるものと、ＫＷＩＣ表現上で使用されるものの２種類に大別される。前者をソース表現上の校正知識、後者をＫＷＩＣ表現上の校正知識と呼ぶ。以下の節では、この校正知識について詳しく述べる。

4.5 試作システムの校正知識

ここでは試作システムにおける校正知識をいくつかの例を用いて説明する。

4.5.1 ソース表現上の校正知識

ソース表現上の校正知識は、複雑で曖昧な名詞句に警告を与えたり、ミスタイプや同じ表現の繰り返しをみつけたりする。日本語の文法はある種の助詞を繰り返し用いて１つの長い名詞句を作ることを許してしまうため、１度読んだだけでは分からないような、係り受けの曖昧な表現がある。文章中のそのような箇所に警告を与えてユーザに書き換えを促したり、辞書に載っていないような語が現れたときにミスタイプの可能性はないかユーザに尋ねたりするのである。また、第4.2節で述べたようにワードプロセッサによって作成された文書にはかな漢字変換の誤りや変換ミ

スによる表記の揺れが生じやすい。ここではまず不均一な語の使用をみつけて警告する校正知識を示し、それについて説明を加える。

<例4．1>ルール番号106：不均一な語の使用　XとYはテキスト中に現れる2種類の漢字短単位語基とする。ここで、ある漢字複合語XYがあり、同時にテキスト中に表現XCYが現れていたとする。Cは任意の表現である。

CASE1：Cが漢字接尾語の場合　→　警告する

CASE2：Cがひらがなの場合

CASE2-1：Cが等位接続詞の場合　→　何もしない

CASE2-2：上記以外の付属語の場合　→　警告する

CASE3：上記以外の場合　→　何もしない

この校正規則により、例えば文書中のある箇所では「編集画面」と使っている漢字複合語が、別の場所で「編集用画面」と使われていた場合、「用」は漢字接尾語なのでCASE1の適用により、警告する。また同じく、「編集のための画面」も、CASE2-2により警告する。しかし、Cとして「および」「または」などの語がくる場合は、その前後にくる語XYは同等の重みをもつ、対等な語として扱われていると考えられるため、「編集および画面」「編集または画面」という句は「編集画面」とは全く異なる概念を指している。このような場合は警告しない。

次にもう1つ、かな漢字変換による助詞の誤変換の可能性を指摘する校正知識の例をあげる。

<例4．2>ルール番号304：助詞の誤変換　X、C、Yはテキスト中にこの順に現れた3種類の漢字短単位語基とする。ここでCは、ある助詞と同じ読みの漢字である。

CASE1：Yが読点の場合　→　警告する

CASE2：Yがひらがなの場合

CASE2-1：Yが文法的にCに接続できない場合　→　警告する

CASE2-2：上記以外の付属語の場合　→　何もしない

CASE3：Yが漢字短単位語基またはカタカナ語、あるいはアルファベットで書かれている場合

CASE3-1：Xが漢字でない場合　→　警告する

CASE3-2：Xが漢字でYがアルファベットまたはカタカナで書かれている場合　→　警告する

CASE3-3：上記以外の場合　→　何もしない

CASE4：上記以外の場合 → 何もしない

この校正規則３０４は、本来ひらがなのままでよい助詞が誤った操作により漢字に変換された場合を想定して、それをチェックするための校正知識である。今、Ｃが漢字の「野」であったとする。続く文字が「、(読点)」であった場合、「野」は単独に１語で現れ、そこで「、(読点)」によって中止されていることから、「の」あるいは「や」が誤って変換されたのではないかと考えられる。そこでＣＡＳＥ１に従って警告する。知識工学分野、人工知能分野、…」のように、「分野」という語の一部が「、(読点)」によって繰り返されている可能性もあるが、ここではＣは１つの独立した短単位語基と仮定しているので、そのような場合についてはこの校正知識は適用されない。また「野」の次に「ため」のようなひらがな語が現れた場合は、「野」が１つの独立した漢字短単位語基であるとすると、接続不可能なので、警告する。それ以外の例えば「から」「にも」などの助詞の場合は、「野」が名詞であるとすると、「野にも山にも‥」「野から出てきて‥」等の使い方もあることから、これらについては警告はしない。ＣＡＳＥ３の場合は、Ｙが漢字短単位語基またはカタカナで書かれた語またはアルファベットで書かれた語であるから、Ｘが漢字でない時、すなわち、カタカナかアルファベットかひらがなで書かれている時にはＣの直前に単語の区切りがある可能性が高い。そして特にＹがカタカナもしくはアルファベットで書かれている場合には、「野」で始まる外来語のように考えられ、不自然である。たとえＹが漢字短単位語基であるとしても、「野」が接頭語的に働くことになる、これも不自然なので警告する。Ｘが漢字であるとしても、Ｙがアルファベットもしくはカタカナの場合は、ＣはＸとＹを結ぶ助詞的役割を果たすことが多いので、警告する。ソース表現上での校正知識の適用の様子を図 4.8に示す。

```
CRITAC SOURCE A1 F72 TRUNC=72 SIZE=211 LINE=120 COL=1 ALT=0
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 |      漢字複合語解析の結果、この自立語の出現確率は                          |
 |      かなり低いと指摘されています。                                       |
 |       --- 誤変換の可能性はありませんか？                                 |
 - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 135  文書そのものとは独立に扱え、また出力媒体や文書の用途に応じて容易に指定
 118  ・変更出来るべきであると考えている。この時点でＣＲＩＴＡＣで扱う前処理
 119  語の文書は英文並の区切られた単語の列であるとする。ただし、この文書には
 120  書く自立語の読み、自立語や付属語の品詞などといった情報が埋め込まれてい
 121  る。
 122
 123  利用者インターフェイスでは構造化文書から図２のような外部表現をつくり出し

**CRITAC**| 1 | 2 | 3 | 4 |**| 5 | 6 | 7 | 8 |**| 9 | 10 | 11 | 12 |
 PROOF  Srce Next Quit Expl  SCHG  ? Back Forw     = Home Word KWIC
====>
```

図 4.8: ソース表現上での校正ルール適用画面

4.5.2 ＫＷＩＣ表現上の校正知識

ＫＷＩＣ表現上の校正知識は、表記や読み等で順序付けられたキーワードに対する述語として定義される。ワードプロセッサによって作成された日本語文書に現れることの多い、誤変換や表記のゆれは、適当なキーワードの順序を与えることにより、ＫＷＩＣ表現の上で隣接したキーワード間の関係としてとらえることができる。例えば、4.5.1節で説明した＜例４．１＞のソース表現上の校正知識は、キーワードが自立部の読みの順に並んだＫＷＩＣ表現のもとで、次のように表せる。

```
CRITAC KWIC A1 F136 TRUNC=136 SIZE=832  LINE=675 COL=1 ALT=2
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 表記のゆれの可能性があります。                                        |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 668  節から導出できる文書構成  要素等      をＰｒｏｌｏｇで記述し
 669  井：計算機による日本語の  用語        ・固有名詞の校正、ＩＣ
 670  良してこれらを扱うことが  予想        される。
 671  Ｃの知識ベースに追加する  予定        である。
 672    現在このような拡張性を  より        高めるために、構造化文書
 673              [プログラム  呼び出し    ]  構造化文書に対して利
 674  部表現や校正知識ベースの  呼び出し    による校正メッセージの生
 675  ｒｏｌｏｇのプログラムの  呼出し      が指定できる。
 676  ｇで記述しており、これを  ライブラリ  としてＣＲＩＴＡＣの知識
 677  どを大まかに把握するのに  利用        できる。
 678  対話的校正を行なったり、  利用者      がＰｒｏｌｏｇで記述した
 679  し]  構造化文書に対して  利用者      が特別に実行したいＰｒｏ
 680  関するものは、文書入力の  リズム      を乱すことや、チェックに
 681              これにより、  例          えば人名のような特定の語
 682                 前者の  例            としては、ＥＰＩＳＴＬＥ
 683  切り)・・・という可変長  レコード    形式のファイルを指定でき
 684  文頭、文末のパターンや、  論旨        の流れなどを大まかに把握
**CRITAC**| 1 | 2 | 3 | 4 |** | 5 | 6 | 7 | 8 |** | 9 | 10 | 11 | 12 |
 KWIC   Prf   Yomi  Quit  Expl   SCHG  ? Back Forw    = Home Word Srce
 ====>
```

図 4.9: KWIC 表現上での校正知識適用画面

<例4．3>ＫＷＩＣ表現の校正知識 自立部の読みの順に並べられたＫＷＩＣ表現の第ｉ行のキーワードとなる自立語を $K_j(i)$、$K_j(i)$ に後接する付属語連鎖を $K_f(i)$ 、そのあとに現れる自立語を $K_j(i+1)$ と書く。$K_f(i)$ は『による』や『の』といった付属語列か、接続詞『および』等から成るものとする。ソース表現上の<例4．1>と同様の校正知識におけるＸ、Ｃ、Ｙはそれぞれ $K_j(i)$、$K_f(i)$、$K_j(i+1)$ に相当する。また、キーワード内部の構造は、$K_j(i)$、$K_j(i+1)$ のそれぞれについて、$K_j(i) = K_i1 \ || \ K_i2$、$K_j(i+1) = K_{i+1}1 \ || \ K_{i+1}2 \ || \ K_{i+1}S$ とし、$K_x1 \ || \ K_x2$ はｘ番目のキーワードが２文字漢語のうちの１文字目と２文字目で構成されていること、K_xS はそのキーワードには１文字の接尾語が後続していることを表している。

CASE1：ある接尾語Ｃがあり、$K_j(i+1) = K_j(i)$ —— Ｃ、$K_f(i)$ と $K_f(i+1)$ は空が成り立つ場合 → 警告する

CASE2：$K_j(i) = K_j(i+1)$ 、Kf(i) が空、Kj(i) = Kj(j+1) で、$K_f(i+1)$ が空でない場合

CASE2-1：$K_f(i+1)$ が等位接続詞の場合 → 何もしない

CASE2-2：上記以外の場合 → 警告する

CASE3：上記以外の場合 → 何もしない

ＫＷＩＣ表現上で文書校正知識を適用するもう１つの利点は、誤りが隣接したキーワードリスト上で表示されるため、人間にとって非常に誤りを発見しやすいことである。<例３．１>のソース表現の校正知識では、文書に誤りが点在する場合を検出しても、それを一つの画面上でユーザに分かり易く示すのは容易ではない。一方、ＫＷＩＣ表現上で『呼出し』と『呼び出し』といった表記のゆれを検出する規則や同音異義語を検出する規則は容易に記述することができるし、それをユーザに示すのは容易である。図4.9に、キーワードが自立語の読み順に並んだＫＷＩＣ表現で、表記のゆれを検出した例を示す。このように、ＫＷＩＣ表現ではキーワードの順序が本質的である。より複雑な順序として、キーワードのみでなく、それに前接する語や後接する語の表記や読みを組み合わせることで、より多くのかな漢字変換の誤りを指摘できると考えている。

現在、校正知識の数は３０個で、知識の種類は表4.7に示すとおりである。

表 4.7: 校正知識の種類とカテゴリ

カテゴリ	番号	校正知識の説明
スタイル	101	文の長さ
	102	自立語の長さ
	103	付属語の長さ
	104	接続詞の使用
	105	受身の使用
	106	表記のゆれ
	107	助詞の使い方
	108	漢数字とアラビア数字
	109	かなとアルファベット
	100	ことば使い
	110	慣用句の使い方
ミスタイプ	201,203	句点の使い方
	204	読点の使い方
	205	ローマ字かな入力
	206	ひらがな列
	207	かっこの対応
誤変換	301	同音異義語
	303	未変換の漢字接尾語
	304	助詞の誤変換
	305	接続検定
	306	漢字複合語の中の誤り

4.6 校正知識の開発環境

試作システムの校正知識のデバッグには、Ｐｒｏｌｏｇの視覚的ディバッガであるＰＲＯＥＤＩＴ [54, 55] を使用した。ＰＲＯＥＤＩＴは汎用のディスプレイ端末上でＰｒｏｌｏｇのプログラム開発を支援することを目的としたシステムで、ユーザはＰｒｏｌｏｇプログラムの編集、ステップ実行、デバッグをＰＲＯＥＤＩＴ上で行うことができる。このシステムはもともと１バイト・コードのデータしか扱えなかったため、筆者らは２バイト・コードが扱えるよう修正を加えて、校正ルールの開発に役立てた。

ＰＲＯＥＤＩＴの実行画面を、図 4.10 から図 4.12に示す。１回の実行ではそのゴールを導出するために使われたクローズの右辺に現れるサブゴールだけを表示する。そして、ユーザがトップダウン的にその実行過程を見たいと思うサブゴールを指定するたびに、それが実行され、再びそのサブゴールを導入するために使われたクローズの右辺に現れたサブゴールだけを表示する。

これにより、ある１つの校正知識が正しく文章中の誤りをみつけることができるかどうかをステップごとに実行過程を確認しながら、校正知識の開発を進めることができるのである。簡単な例でその実行過程を追ってみることにする。

この例は、文の終わり方をみて、誤った句読点の打ちかたをしているもの（ミスタイプ）をみつける校正知識である。まず句点をみつけ、その直前に現れた文節の最後の文字が「い」「え」の段の文字であったら、それは読点であるはずのものが誤って句点を打たれた可能性があるとして、警告を発する。

＜例３．４＞ルール番号 201：句読点の打ち誤り　Ｘはテキスト中に現れる漢字短単位語基、ＰはＸの直前の句読点とする。

CASE1: Ｐが空の場合　→　何もしない

CASE2: Ｐが読点の場合　→　何もしない

CASE3: Ｐが句点の場合直前の語Ｘを取り出し、その最後の１文字ＬＣをみて、

　CASE3-1: ＬＣが「い」段の文字または「え」段の文字の場合　→　警告する

　CASE3-2: 上記以外の場合　→　何もしない

CASE4: 上記以外の場合　→　何もしない

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>  CriDebug  <<<<<<<<<<<<<<<<<<<<<<<<<<<

goal : rule(p4) : exit
rule : rule(p4) <- punc(V1, V2, V3, V4) & V4 = '。' & tail(V1, ...) && (...)
rule      punc    =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =        tail      lchar        |
                                        <-
                            <-
                      <-
          ->      =
          ->      =
redo : punc(V1, V2, V3, V4)
exit : punc(2, 7, 9, '。')

103 step rule(p4).
104 え
105 え
```

図 4.10: PROEDIT − 1 −

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>  CriDebug  <<<<<<<<<<<<<<<<<<<<<<<<<<<

goal : rule(p4) : exit
rule : rule(p4) <- punc(V1, V2, V3, V4) & V4 = '。' & tail(V1, ...) && (...)
rule      punc    =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =        tail      lchar        |
                                        <-
                            <-
                      <-
          ->      =
          ->      =
call : 'る' = 'え' | 'る' = 'け' | 'る' = 'せ' | 'る' = 'て'
fail : 'る' = 'え' | 'る' = 'け' | 'る' = 'せ' | 'る' = 'て'

103 step rule(p4).
104 え
105 え
```

図 4.11: PROEDIT − 2 −

```
>>>>>>>>>>>>>>>>>>>>>>>>>>  CriDebug  <<<<<<<<<<<<<<<<<<<<<<<<<<<<
goal : rule(p4) : exit
rule : rule(p4) <- punc(V1, V2, V3, V4) & V4 = '。' & tail(V1, ...) && (...)
rule      punc    =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =
          ->      =        tail      lchar        |
                                        <-
                            <-
                      <-
          ->      =
          ->      =
          ->      =        tail      lchar        |   prst
 call : 'え' = 'え' | 'え' = 'け' | 'え' = 'せ' | 'え' = 'て'
 exit : 'え' = 'え' | 'え' = 'け' | 'え' = 'せ' | 'え' = 'て'

103 step rule(p4).
104 え
105 え
```

図 4.12: PROEDIT − 3 −

このとき、この校正知識を試行するテキストには1箇所「‥を行え。」という文章を意図的に挿入してある。この校正知識が正しく働けば、「え」の後ろの「。(句点)」は「、(読点)」の誤りではないかと警告することができる。図 4.11の画面では、句点がみつかって、直前の文節の最後の文字を取り出すところまで成功していても、その文字が「い」の段でも「え」の段の文字でもないことによって失敗している様子が分かる。そしてサブゴールの実行を進めていき、最後に図 4.12の画面中央に成功したサブゴールが示され、この校正知識がミスタイプの可能性のある句点をみつけられたことがわかる。校正知識が正しい場合についてのみ説明したが、校正知識に誤りがある場合、ゴールは失敗してしまう。このようなときは、成功するはずの文章(上記の例では例えば、「‥を行え。」のように誤りを含んでいる文章)を対象としてルールを実行し、どの時点で失敗しているのか、サブゴール内の実行過程を詳しくみていく。そして、全体の失敗を引き起こしている条件を書き直して再びＰＲＯＥＤＩＴ上で実行してみる。この操作を繰り返すことにより、校正知識が整備されていく。

4.7 日本語文書校正支援における拡張機能

最近では、新聞記事を入力として、意味解析、要約処理をし、記事の要約リストを出力する要約支援システム [19] のようなものも考えられ、実用化され始めている。そこで本節では、文書校正支援試作システムにおける構造化文書という文書モデルを有効に活用する1つの応用として重要語検出機能を追加することとした。

欧米の文書では、単語の頻度情報をもとに、重要語を抽出する手法が提案されている [90]。表意文字である漢字を欧米語の単語に対応させれば、漢字の頻度情報をもとに同様の処理で重要語を自動抽出する手法が考えられる。従来より漢字の出現頻度や、漢字カタカナ列の頻度情報に注目して、語の分類や重要語の抽出を機械的に行うことが考えられてきている [7, 8, 9] 。この機能を実現するためには、まず試作システムで本来利用していた構造化文書という文書表現の中に、自立語が重要語になりうるかどうかの情報を付け加える。つぎに必要な重要度に応じたレベルの重要語を表示するように指定すれば、画面上に重要語とその前後の文脈を表示することができるのである。本節では、日本語文書の校正を支援する試作システムに追加した重要語の検出機能と、この拡張機能の働きについて詳しく述べる。

4.7.1 重要語検出方法

重要語を検出するためのキーは、漢字もしくはカタカナ列とする。トレーニングデータとして、ＪＩＣＳＴの文献抄録を用い、ＪＩＣＳＴの文献分類カテゴリに対応して出現頻度をカウントする。その後、分類カテゴリ間の頻度分布を計算し、分布に偏りがあるものだけを、キーとなる漢字もしくはカタカナ列と定義した [7, 8, 9]。類似した偏りのあるもの同志をまとめるには、クラスタ分析を応用した手法を用いた。文中の名詞列を対象に次のいずれかの条件を満足するものを、重要語とする。

1. キーとなる漢字あるいはカタカナ列を2つ以上含み、かつ文字数が3文字以上の名詞列

2. キーとなる漢字あるいはカタカナ列を1つ含み、かつ文字数が4文字以上の名詞列

3. キーとなる漢字あるいはカタカナ列を1つ含み、かつカタカナ列を含む名詞列

この結果、上記の条件では多少ノイズとなるものも抽出されるが、重要語を拾い出せないことのないよう、いわゆる適合率を高くすることを優先して考慮した。

4.7.2 拡張機能の動作

重要語検出機能は、試作システムの2種類のモード（対話的モードとバッチモード）で使用できる。

対話的モード 対話モードでは、ユーザはソース表現とKWIC表現という2種類の外部表現を提供されている。重要語検出機能はソース表現上では重要語に下線が付され、かつ重要語のレベル（重要度）が表示される（図 4.13参照）。そしてKWIC表現上では例えば重要度5以上の語をリストにして表示するなどの操作が可能である。重要語のレベルを付けるには各種の方法が考えられるが、ここでは重要語のもつ生起確率 10^X の指数値、すなわちXの値を用いた。この機能を使うことにより、ユーザは自分がどの段落でどのような語（重要語）を用いて論を展開したか、その章のタイトルと内容が適切かどうかなどについて確認できる。

```
CRITAC SOURCE A1 F72 TRUNC=80 SIZE=57  LINE=23 COL=1  ALT=1

 14  で、文書校正のツールにより、すでに出来上がった文書に対して語用法や文体
       (1)-----                                          (1)---
 15  チェックのためのプログラムや、文章の質を評価することにより、より良い文
                  (1)------
 16  章に改良する方式などが試作検討されつつある。

 17

 18  我々は従来から漢字複合語の短単位分割や日本語文の文節切り等の手法を研究
                        (1)------   (1)----  (1)-----
 19  仕手折、これらの技術的基板から構造化文書をいう文書の概念構造を提案し、
                          (1)------                (1)----
 20  このうえで利用者用の二つの外部表現および文書校正環境を実現した。本報告
             (1)----        (1)----      (1)--------
 21  ではこれらの概念とその試作システム野うえで実現した機能について述べる。
                      (1)--------
 22  文書前処理部ではべた書きの文書を構造化文書というプロローグの節に変換す
     (1)--------                (1)------
 23  る。もとの文書のなかで書式に関する情報は現在は取り扱っていない。書式は
                        (1)-                                        (1)-
**CRITAC**| 1 | 2 | 3 | 4 |**| 5 | 6 | 7 | 8 |**| 9 | 10 | 11 | 12 |
 PROOF  Srce Next Quit Expl  SCHG  ? Back Forw     = Home Word KWIC
====>
```

図 4.13: 重要語検出画面

バッチモード バッチモードはもともと段落や文書全体といった大きな単位での校正処理や各種の統計情報を利用する、きめの細かい文書の校正を目的としている。重要語検出機能はその目的をさらに追求するために有効である。

例えば、表4.8のような重要語情報が示された場合、その章に出現する重要語を知ることによって、各章の内容を推測することが可能となる。

表 4.8: 重要語の出現箇所と出現回数

出現箇所		重要語	回　数
1章	段落1	校正作業	5
		日本語	3
		辞書ベース	2
1章	段落2	ワードプロセッサ	3
		ユーザインタフェイス	2
2章	段落1	文書データベース	2
		漢字複合語	2
		統計情報	2

また、表4.9のような出力情報により、重要語の初出箇所とその文章の性質と予測することができる。そして、重要語は最初に出現する際、その文書の読者に誤解をまねかないためにはまず、その語の定義を与えるべきであるといった経験則から、定義文でない可能性のある、例えば、

「校正知識は … 用いる」

の文を抽出し、校正知識というものの定義を与えているかどうか、実際の文章にあたって調べてみるよう、ユーザに注意を促すことも可能である。

表 4.9: 重要語の初出箇所と文末表現

初出箇所		重要語	付属語	文末表現
段落1	行5	校正作業	とは	である
段落2	行2	校正知識	は	用いる
段落2	行12	文書校正作業	を	定義する

4.7.3 拡張機能の評価

このように重要語の重要度を辞書にもたせて、それに応じて各重要度の重要語を随時表示することにより、文章の要約を作成したり、またキーワードを付与したりすることが容易に行える。ま

た、各文書ごとのキーワードを文書データベースに保存することにより、各自の作成した文書の分類・整理が可能になる。重要語の抽出方法については、求める重要語や重要度が迅速に、もれなく、精度高く行えるよう今後も改良をかさねる必要がある。

4.8 試作システムの機能と評価

4.8.1 誤りの分類と調査

校正支援試作システムの評価のため、ワードプロセッサで作成された2種類の比較的短いサンプル文書で、そこに現れる誤りを、第4.5節で述べた校正知識がどの程度検出することができるかを調査した。この際、あまり作成中の画面に注目せず、下書きの原稿を見ながら、無造作に変換キーや文字種キーを押した。なお、ここではローマ字かな漢字変換入力を行い、文節単位のかな漢字変換を行った。文書の大きさとしては、1つ（文書S）が2099文字、もう一方（文書T）は2779文字で、2文書とも情報工学分野の論文である。文書Sも文書Tも筆者が入力したもので、文書Sはしばらく時間をおいてから、同じワードプロセッサを使用して今度は比較的注意深く入力してみた結果、文書S'を得た。誤りを分類した結果を表4.10に示す。

表 4.10: サンプル文書に現れた誤り

誤りの種類	文書S	文書T	文書S'
誤変換 α	24	30	3
誤変換 β	2	0	1
未変換 α	8	0	0
未変換 β	1	2	0
ミスタイプ	7	5	2
文字種キーの押し忘れ	2	1	0
変換され過ぎ	10	5	0
固有名詞	2	0	0
表記のゆれ	0	4	0
スタイルの乱れ	0	1	0
誤修正	0	1	0
誤りの数の合計	56	39	6

ここで、誤変換に2種類あるのは、誤変換 α というのが、いわゆる同音異義語による誤りで、誤変換 β というのが、ミスタイプによる誤変換である。

未変換についても誤変換と同じく、未変換 α は単純に変換キーの押し忘れと思われるもので、未変換 β はミスタイプのせいで、該当する漢字がみつからず、変換されずに残ってしまったと思

われるものである。

その他のミスタイプには、熟練者には起こらないであろうが、「でけあがった←できあがった」(本当は右手中指の「ｉ」を打鍵しようと思っているのに、左手中指の「ｅ」を打鍵してしまったことによる誤り)や「コーコ←コード」(左手中指の「ｄ」を打鍵するはずなのに、右手中指の「ｋ」を打鍵したことによる誤り)のような、右手と左手の動作誤りが特徴的であった。

文字種キーの押し忘れによる誤りは少なく、全体で3回しか現れなかったが、これらのうちの2つはカタカナ語のあとでひらがなキーを押さずに続けて付属語を入力してしまっており、残る1つは、カタカナ語(「プログラム」)がひらがなで書かれていた。このような誤りは使用するワードプロセッサやかな漢字変換プログラムによって異なる。

「変換され過ぎ」というのは、長い単位で文節変換を行なったときや、または助詞がカタカナ語とカタカナ語をはさんで途中にあるときに本来ひらがなのままでよい文字が変換キーによって変換されてしまったような場合を指す。

標記のゆれは、外来語、専門用語に多く、アルファベットで書かれる場合にもカタカナでかかれる場合にも起こりやすい。例としては「ＰＲＯＬＯＧ」と、全部が大文字で書かれている場合と「Ｐｒｏｌｏｇ」のように最初に1文字だけが英大文字で2文字目以降は英小文字で書かれている場合の混在や、「インターフェース」のようにのばす音を「ー(長音記号)」で書いた外来語と、同じ語を「インターフェイス」のように、より原語の発音に忠実に表記した外来語などがある。

誤修正は、もともと正しく書かれていたであろう語の一部が消去されてしまっていたものである。このような例は1箇所のみで、「辞書サーバ」と書かれるはずの箇所が「辞サーバ」と書かれていた。

4.8.2 実験結果と市販のワープロ機能との評価

ここでは、提案する手法を用いて、前節のサンプル文書に現れた誤りをどの程度まで発見可能であるかという点と、同様の誤りを市販のワードプロセッサーの機能がどの程度検出、修正可能であるかという点について比較考察する。ここでは比較したワードプロセッシングソフトは広く普及しているジャストシステムの「一太郎 v.7」である。

一太郎にはツールとして、「文書校正」「スペルチェック」といった機能が提供されている。このうち、スペルチェックは、英単語のつづりをチェックする機能で、この機能では日本語はチェックできない。「文書校正」では、「表記のゆれ」「使用単語一覧」「かっこ」等の機能がある。「表記のゆれ」を機能させると、「受け付け」と「受付」のように同じ読みをもつ語がＫＷＩＣで表示される。ここで「一括置換」を行うとすべて同じ表記に変換される。また、「使用単語一覧」機能では、ＢＣＲＩＴＡＣと同様に、使用している単語を指定された並べ方で並べたリストが出力される。「かっこ」機能では、かっこの対応がとれていない場合に、対応のとれない文章の一覧が表示

される。

また、「単語情報」という機能を用いると、指定した単語の読みとその読みに関する辞書情報が表示される。実際にこれらの機能を用いて、前節の誤りについて検出実験を行い、提案する手法と比較した。

ＫＷＩＣ表現上で単語を読みの順に並べ、読みが同じで表記の異なる語が現れていたときに警告するような校正知識を働かせることで、前節で説明を行なった誤変換の内、本手法では現在４８．８％は検出できる。これについては、一太郎でも同様の結果が得られた。誤変換βについては「救って」（正確には「作って」）や、「苦情」（正確には「向上」）など、この文節が本当に誤りかどうかを判定するには品詞や語の使用頻度だけでなく、文の意味まで考えなければならない。これらの文節がローマ字かな漢字変換入力で、「**tsukutte**」と打鍵するべきところを「ｔ」を打ち損じたかあるいは、何らかの操作ミスにより消してしまったかで「**sukutte** 」となってしまったとか、同じように「**koujou**」の１番目の「ｏ」が抜けて「**kujou** 」になったのであろうといった細部について、想像はできても、英語などのように、入力した文字がそのまま出力される言語と異なり、ワードプロセッサでは、１番多くの場合、日本語の読みからローマ字つづり、ローマ字からかな、かなから漢字へと３段階の変換が行われており、最終的な漢字の出力から本来入力されるべき文字列を想像することは非常に困難である。試作システムでは現在のところ、構文的な情報や意味は扱わないので、誤変換βに対しては検出・訂正の方法がない。一太郎では、形態素解析が行われているが、これらについては正しく形態素解析が行われ、エラー情報は出力されなかった。

未変換αについては、変換されるべき正しいかな文字列が入力されているが、提案する手法を用いて作成する構造化文書では正しく自立語と付属語に分離できない可能性がある。今回のサンプルでは８個の内１個については正しく文節区切りできない。残り７個については文節切りは正しいので、文節内の自立部と付属部の構造をうまくとらえることができれば、辞書を引くことで辞書の表記を取り出すことができる。一太郎では未変換文字は場合によっては正しく文節区切りできないため、未変換αの検出率は本手法よりも同等あるいは低くなる。

一方、未変換βの内、文書Ｔに現れた「ひょうげんん」（正確には「表現」）については、ひらがなで「ん」が２つ続くような語はないので警告できる。しかし文書Ｓに現れた「ねべてきた」（正確には「述べてきた」）については、例えばミスタイプは１つの文節中では１箇所にしか現れず、かつ文書の入力が常にローマ字かな漢字変換であるとすると、そのローマ字表現「**nebetekita**」のうちのどこか１文字を別のアルファベットで置き換えることで、語を創造できるが、そこから派生する語は非常に多くなる可能性があり、処理も複雑なので、当面はこの種の誤りの訂正は考えない。但し、「ねべる」という語は辞書になく、文節区切りも失敗するので、検出することは可能である。文字種キーの押し忘れによる誤りは現段階の試作システムでは、付属語接続検定の折りに接続に失敗し、かつそのような自立語も辞書にないことから、検出し、警告することができる。

今回の文書に現れたのは「ワード・プロセッサニオイテ」と「チェックサエナク」の 2 つの誤りで、いずれも文章の 1 文節が全てカタカナで書かれている。一太郎においては、「ひょうげんん」も「～についてねべてきた」も「ワード・プロセッサニオイテ」もすべて一語の未知語とされ、特に「ついてねべてきた」では、文節区切りも不可能で全体で一語とみなされた。

以上の考察から、提案する手法を用いて現段階でどの程度の誤りが検出できるか、将来はどの程度まで可能であるかについて、誤りの種類別にグラフにしてみた（図 4.14 参照）。

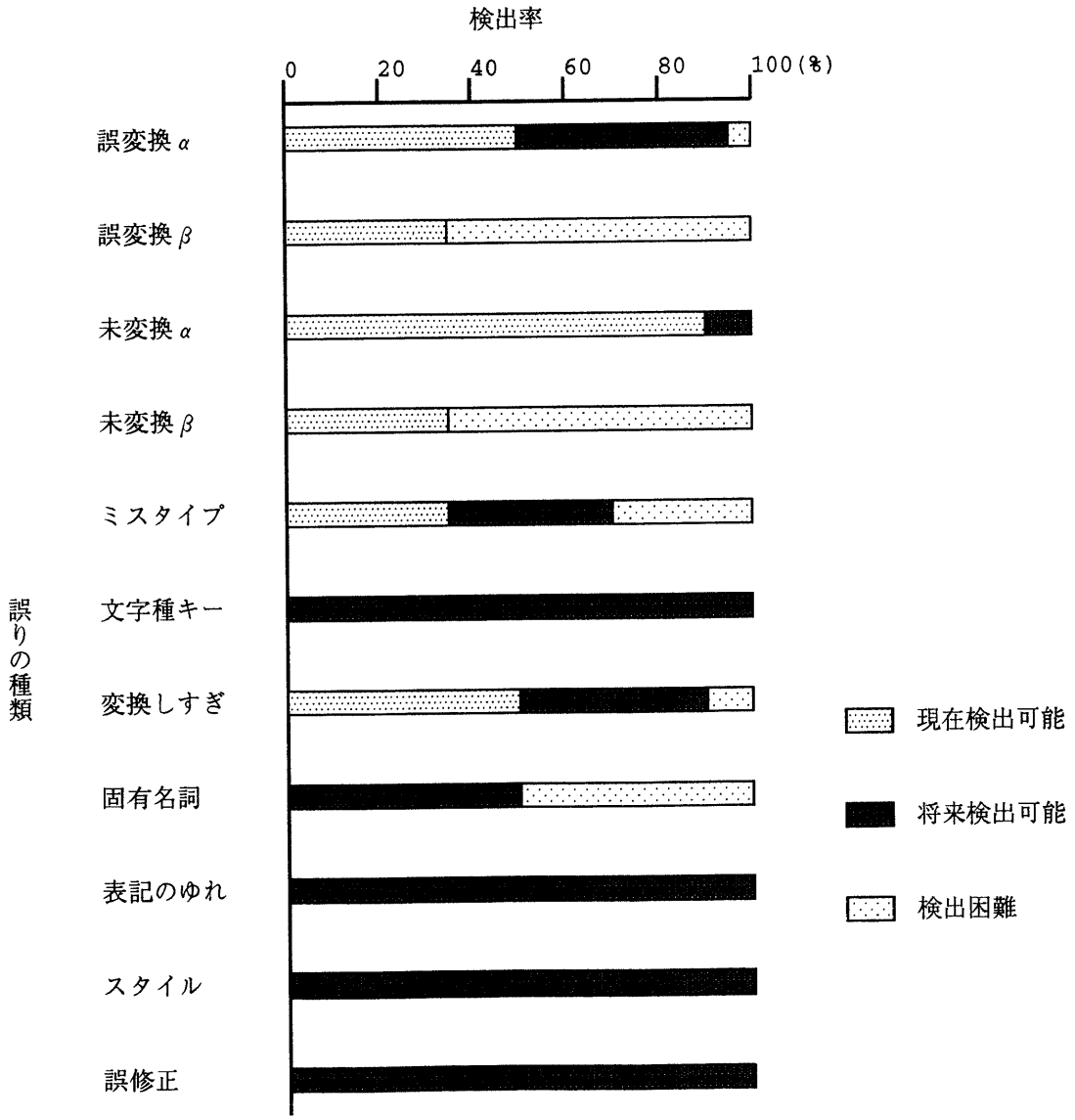

図 4.14: 誤りの検出・訂正率

4.9 まとめ

ワードプロセッサが大量に普及し、日本語文書を電子的に作成、配布、印刷することが日常的になってきた。しかし、このような状況の中でも計算機上でできあがった文書の校正・推敲を行なうといった高度のテキスト処理は、最近になってやっと研究が行なわれ始めたところで、まだ実用化の段階には至っていない。本章では、機械可読な日本語文書を対象として文書中の誤りや用語の不統一、言い換えたほうがよい表現などを検出し、文書の校正支援を行なう手法について考察し，実際に試作システムを構築したので、それについて述べた。この結果，構造化された文書表現 (構造化文書) とその上でのルール形式の校正知識表現を用いることが有効であるという結果を得た。すなわち、

1. 文書を前処理段階でモデル化することにより、日本語文書のための応用プログラム実行時には字句解析を行なうことなく、単語や文節、段落や文書全体といった単位を扱うことができる、

2. 校正知識は構造化文書上の高レベルの述語として記述できる、

3. 文書校正知識を複数の段階で (入力時と作成時それぞれにおいて) 利用できるように、対話的文書校正とバッチ的文書校正が提供できる、

といった特徴を実現した。

ここでは、ワードプロセッサによって作成された文書に現れやすい誤りを指摘するため、まず実際の文書に現れる誤りを調査・分類した。さらにそれらの誤りのうちどの程度が機械的に検出可能かについて検討した。また、より高度な文書処理機能の一例として重要語検出機能を追加し、試作システム上に実現した。

本章で述べた機能の一部はワードプロセッサの高機能化によって代替できるものもあり、一部実現されている。しかし、文書中に出現する同じ読みを持つ語を一度に表示する機能 (KWIC機能)、初出語と文末表現の提示機能や、重要語表示機能など、文書作成中対話的に行うよりは、作成後の文書を有効に活用するための機能もあり、今後のインターネット社会における高度文書処理技術に有効な構造化文書という概念の提案と今後の可能性について十分な検討を行なうことができた。