日本語文書処理 8－5
（1986・9・17）

日本語文書校正支援システムCRITAC

鈴木恵美子　武田浩一　藤崎哲之助

日本アイ・ビー・エム株式会社　東京基礎研究所

我々は従来より漢字複合語の短単位分割や文節切りの手法について研究しており、べた書きの文書の文節切り、読みや品詞情報の付加が高い精度で実現できる。これにより、従来知的処理に向かないべた書きの文書形式を単語や文節、その読みや品詞という付加情報をもった文書に構造化できる。我々はこれを構造化文書と呼んでいる。

文書校正のための知識は論理型プログラミング言語Prologで表わされており、構造化文書上で働く校正ルールには、構造化文書の2種類の外部表現形式「ソース表現」とキーワード/文脈(KWIC)表現」の各々に対して、ソースルールとKWICルールがある。

ルールの数は全体で30個と少なく、現在のところまだ簡単なルールしか実験されていいが、簡便な対話的校正処理が実現できたので、報告する。

CRITAC - A Japanese Text Proofreading System

Emiko SUZUKI　Koichi TAKEDA　Tetsunosuke FUJISAKI

Tokyo Research Laboratory, IBM Japan, Ltd.
5-19 Sanban-cho, Chiyoda-ku, Tokyo 102, JAPAN

In this paper we describe a prototype system CRITAC (CRITiquing using ACcumulated knowledge) for proofreading Japanese text.

We have studied a method to decompose Kanji compound words into primitive words and to decompose continuously typed Japanese text into a sequence of tokenized primitive words and particles. This allows us to convert text into an internal form with added information such as word boundaries, parts-of-speech, reading, semantic markers for each fragment (called segment) of Japanese text. The internal form is a set of Prolog facts which is referred to by CRITAC proofreading rules.

The knowledge base contains Prolog-coded heuristic rules which are designed to detect possible errors.

The user interface is built upon an editor and SQL/DS on-line dictionary server. The editor offers two views of the text - a simple text view and a keyword extraction (KWIC : Key Word In Context, for short) view.

1 まえがき

ワード・プロセッサの普及に伴い、日本語文書を作成することが容易になってきた。しかし、できあがった文書の校正・推こうを行うといった高度のテキスト処理はまだ実用化に至っていない。ワード・プロセッサの、文書を作成したり印刷したりという基本的な機能は向上しているようであるが、英文では既にパソコン上で機能するスペルチェックさえなく、ユーザーはすべての文を自分の目で確かめているのが現状である。

これは主として、日本語のこう着性や日本語文書の正書法が確立していないこと［11］等による。従って英文並みのスペル・チェックやスタイル診断［14］といった校正機能を実現するためには、日本語の性質を考慮した上での校正方法を研究する必要がある。ユーザーにもよるが、ワード・プロセッサの使い方として、「とりあえず原稿を全部入力してしまい、仮名漢字変換の誤りやミスタイプはあとでまとめて修正することにする」という使い方が考えられる。もちろん、原稿を見ながらも、注意深く画面に注目するユーザーもいるだろうが、これは現在のワード・プロセッサの機能が十分でないためであり、もし、仮名漢字変換の誤りや表記のゆれなどを容易にチェックできる手段が提供されれば原稿と画面と両方に注目するといった非生産的な作業は行なわれなくなるであろう。

我々は従来より漢字複合語の短単位分割［7］や日本語文の文節切り［6］等の手法を研究しており、これらをもとに、文書を高度に構造化した構造化文書というものを提案した［9］。この構造化文書では、日本語の文章は文節に区切られ、読みや文法やカテゴリといった情報が付加されている。我々は構造化文書上に、「べた書き表現」と「キーワード／文脈表現」というユーザー用の2種類の文書表現と文書校正環境を実現することを目指している［8］。本論文では、ワードプロセッサで作成した文書に現れやすい誤りを検討し、構造化文書上で校正知識を表現する方法について検討したので、それについて述べる。また校正知識のデバッグ環境、現在までに開発された校正知識とその評価についても報告する。

2 日本語文書の校正技術

2.1 現状

最近になっていくつかの日本語文書校正支援システムが試作されているが［1,2,5,9,10］、ワード・プロセッサの誤りのなかで最も現れやすい仮名漢字変換の誤りや表記のゆれの検出・訂正などについての系統的な手法はまだ得られていないようである。

建石ら［10］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、ユーザーに注意を促す。また、一つの文の長さの最大値、平均値等を用いて、文章の読みやすさの測定も行っている。

石井［1］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、言い替えた方がよい言葉使いなどについての情報が出力される。

牛島ら［2］は辞書も文法解析も行わずに文章を字面だけで解析し、推こうするツールを開発している。ここでは句点などに注目して文を切りだし、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ーに出力させたりすることができる。

絹川［5］は仮名漢字変換レベル、語・句のレベル、文のレベル、段落のレベルの各レベルにおいて、文章の均質化を図ろうとしている。これは複数の人が分担して一つの文書を作成するような場合に、書き方やことば使いの不統一を防ごうとするもので、あいまいさの少ない表現をするために、ワード・プロセッサに必要になる機能の検討を行っている。

以上日本語文書の校正のために行なわれている研究のうちいくつかを挙げてみたが、これらは一般的な校正技術や語用法の取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明らかにしておらず、スタイル以前の誤りを指摘するには至っていない。

2.2 ワード・プロセッサで作成された文書の誤り調査

日本語文書中に現われやすい誤りを調査し、分類を行なうために、ワード・プロセッサで作成された3種類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(文書A)が、約9380文字、もう一方(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は同一人により入力されたもので、文書Cは別の人の手による。誤りを分類した結果を表1に示す。

表1. 文書に現われた誤り

	文書A	文書B	文書C
文法的な誤り	1	0	0
仮名漢字変換の誤り	0	0	14
ミスタイプ	0	2	14
言葉づかいのおかしいもの	2	0	1
一文が長過ぎるもの	2	2	0
句点の打ち方の誤り	1	0	0
読点を入れた方が良い文節	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仮名漢字変換の誤りが非常に少なく、ユーザーがかなり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は、今まで自分がどのような文章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く、読みなおしの際に仮名漢字変換の誤りやミスタイプに気がついて修正が行な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、ユーザーが発見できない誤りがあったのである（例「1 入力：二入力」）。

反面、文書Cでは、文書入力者はブラインド・タイプができるため、原稿を見ながらローマ字漢字変換を行っ

ている。その結果ワード・プロセッサの画面にあまり注目せず、仮名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（例「文例終←文例集」）。この文書作成者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つしか現れなかった。しかもこれ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、もともとユーザーが文法的におかしな日本語を書いたとは考えにくかった。

この調査をまとめてみると、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以外に少なく、
2. むしろミスタイプや仮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い、
3. 長過ぎたり同じ言回しを繰り返し用いたりといったことによる読みにくさが生じやすい、

のではないかと考えた。

そして、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考え、文書を校正する知識は文法チェックとは異なる独自のものを実際の文書にあたって、経験的、発見的に獲得することとした。我々はこのためCRITAC(CRITiquing using ACcumulated knowledge) と呼ぶ実験システムを開発することにした。

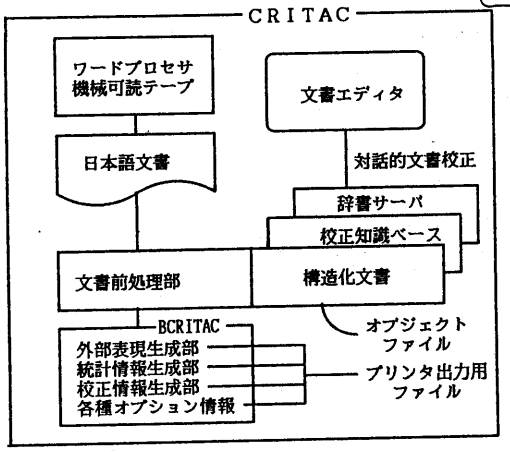

図1. システム構成

3 CRITAC

3.1 システムの構成

ここではCRITACの概要と、基本的な概念について説明する。CRITACのシステム構成は図1のようになっており、大きく分けて次の三つの主要部分：文書前処理部分、ユーザーインターフェイス部分、校正用知識ベースからなる。

［文書前処理部分］

文書前処理部分では、入力されたべた書きの文書に、

1. 文節切り
2. 自立部と付属部に分離
3. 漢字複合語に短単位分割
4. 付属語の接続検定

といった処理を施して図2に示す構造化文書というPrologの節集合に変える。Prologの節にしたのは、後に行う校正処理の入力としてそのまま使用するためである。ワード・プロセッサから得られる文書にはこれらの情報を含むものがあるが、文節区切りや短単位は文書によりばらつきが生じやすく我々の処理には不適当なため、使用しない。

```
IPSJ8610 PRINT    A1  V 283  TRUNC=283 SIZE=2011 LINE=217 COL=1 ALT=0
  206 head(4, 1, 9,' 誤 ',nil,' あやま ',nil,(9),'',nil).
  207 tail(4, 1, 9,' り '.' は '.' でき '.' る '.' だけ ',nil,(87)).
  208 seg( 4, 1,10,' 文書の '),
  209 head(4, 1,10,' 文書 ',nil,' ぶんしょ ',nil,(19),'12',nil).
  210 tail(4, 1,10,' の ',nil,(76)).
  211 seg( 4, 1,11,' 作成時に '),
  212 head(4, 1,11,' 作成時 ',nil,' さくせいじ ',nil,(19),'12S',nil).
  213 tail(4, 1,11,' に ',nil,(78)).
  214 seg( 4, 1,12,' 検出されることが '),
  215 head(4, 1,12,' 検出 ',nil,' けんしゅつ ',nil,(13,19),'12',nil).
  216 tail(4, 1,12,' さ '.' れ '.' る '.' こと '.' が ',nil,(75)).
  217 seg( 4, 1,13,' 望ましく '),
  218 head(4, 1,13,' 望 ',nil,' のぞ ',nil,(8,19),'',nil).
  219 tail(4, 1,13,' ましく ',nil,nil).
  220 punc( 4, 1,13,'、 '),
  221 seg( 4, 1,14,' 将来的には '),
  222 head(4, 1,14,' 将来的 ',nil,' しょうらいてき ',nil,(19),'12S',nil).
  223 tail(4, 1,14,' に '.' は ',nil,(85)).
  224 seg( 4, 1,15,' 仮名漢字変換等の '),
  225 head(4, 1,15,' 仮名 '.' 漢字 '.' 変換等 ',nil,' かな '.' かんじ '.' へんかんとう ',nil,(19),'12','12','12S',nil).
  226 tail(4, 1,15,' の ',nil,(76)).
```

図2. 構造化文書

文節切り：

入力された文はまず文節に分けられる。文節切りのためのルールは約100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。現在、文節を正しく区切る精度は97.5% である。なお、ここでは文節は、大河内ら［3］の拡張文節を指す。

自立部と付属部に分離：

文節ごとに辞書を引き、文節内の自立語を取り出す。

漢字複合語の短単位分割：

取り出された自立語が漢字複合語の場合には、さらにそれを短単位に分ける。日本語はたとえば「不安定」のような語を短単位に分割する場合、辞書の引き方により、「不安・定」「不・安定」のどちらにも分割可能ということがある。このようなあいまいさを解消する方法として、我々は確率的なアプローチをとり、確率付きの漢字短単位辞書を用いている。この方法による漢字短単位分割の精度は約96.5%である［6］。そして最も確からしい分割が決まった時点で漢字に読みがふられ

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ない、正しい付属語列に「五段活用・カ行・連用形」などのカテゴリを表わすカテゴリ番号が付けられる。

以上の処理により、入力された日本語文書は図3に示す4つのPrologの節に変換される。

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)
head(I,J,K,U,Y,G,L)	Uは文節中の自立語のリスト。Yはその読み、Gはその品詞のリスト。LはUが漢字複合語のときその語基のパターンを表わす。
tail(I,J,K,V,H)	Vは文節中の付属語のリスト。Hは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がその文字列となる。

図3．構造化文書の構成要素-1-

また、これら4種の節から、階層的な文書の構成要素である図4に示すような3種の節が定義される。文書中のこの他の構成要素はこれらの基本的な節から定義できる。このようにして得られた、Prolog の節集合によって表わされた文書を構造化文書とよぶ。

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

図4．構造化文書の構成要素-2-

CRITAC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扱わないことにした。これは、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。

〔ユーザーインターフェイス部分〕

ユーザーインターフェイスでは、構造化文書から図5のような外部表現を作成して、ユーザーに提供する。

「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。

「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（漢字複合語の場合はその各基本語）をキーワードとしてその読みの順番等によって並べ代え、前後の文脈とともに表示したものである。従って構造化文書がN個の自立語／基本語をもつと、KWIC表現はN論理行からなる(図6参照)。

これら2つの外部表現は、それぞれエディタによって編集することができ、エディタ上で行なわれた変更等は直ちに構造化文書と、もう一方の外部表現に反映される。

```
CRITAC   SOURCE   A1  F 72  TRUNC=72 SIZE=211 LINE=78 COL=1 ALT=0
 68
 69  ＣＲＩＴＡＣの校正知識は２種類あり、１つはソース画面上で、もう１つはＫ
 70  ＷＩＣ画面上で働く。ソース上ではたらくルールというのは文節、自立語、付
 71  属語などに直接関与して、ミスタイプや不均一な語の使用をみつける。例えば
 72  、ある文が終止形で終わっているのに句点が打たれていないとき、前節で述べ
 73  た文書の内部表現を用いて次のようにして警告する。
 74
 75  ＫＷＩＣ上のルールは隣接するキーワードの関係をみたり、同音意義語の使用
 76  状況を見たりするのに使われる。ＫＷＩＣでは、単語を読み順に並べることが
 77  できるので、同じ読みで異なった語が１つだけ現れていたようなときに（例２
 78  ）に示すようなルールにより、ユーザーに警告を与えることにする。
 79
 80  以上述べてきたように、パターン・マッチングとバック・トラッキング機能を
 81  もつＰｒｏｌｏｇの採用により、校正知識を宣言的かつ単純な形で実現するこ
 82  とができる。また、文書そのものもＰｒｏｌｏｇのファクトで掛かれているこ
 83  とで、校正知識の開発、修正のための良い環境が与えられる。今後、簡単な文
 84  法を備えることにより。果たしてそのくらい文法的なチェックが可能になるの
 85  か、そしてそれによってどのくらい校正能力が向上するのかをみたうえで、そ
 86  のあとの意味処理につなげる方法を検討するつもりである。
 87
```

図5．ソース表現

```
CRITAC   KWIC     A1  F 116  TRUNC=116 SIZE=1138 LINE=786 COL=1 ALT=0
 776  ず、構文解析も行わずに、          入力          文書と不適当な表現を集
 777  Ａと文書Ｂは同一人により          入力          されたもので、文書Ｃは
 778  一がかなり注意深く文書を          入力          したと考えられる。
 779  点字化や検索システムへの          入力          として利用できる。
 780    反面、文書Ｃでは、文書          入力者        はブラインド・タイプが
 781              また、石井は          入力文        をリスト形式に変換し、
 782                                    日本語        文書を作成することが容
 783            これは主として、        日本語        の文書では単語が隣に区
 784                                    日本語        の性質を考慮した校正方
 785  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
 786  ロセッサの普及にともない          日本語        文書の作成は近年急速に
 787            これは主として          日本語        ぶんしょにおける語のこ
 788                                    日本語        文書校正を支援する試作
 789    個のような出力は例えば          日本語        文書の音声か、点字化や
 790              本報告では            日本語        文書校正を支援するため
 791  表記のゆれの検出機能等は          日本語        の独自性を扱う上で効果
 792  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
 793  れによってどのくらい校正          能力          が向上するのかをみたう
 794  文書にあたって、経験的、          発見的        に獲得することとした。
 795                                    果            たしてそのくらい文法的
```

図6．KWIC表現

ソース表現上では、語単位、文節単位の挿入・削除・更新を行なうことができる。そして校正機能は表示画面上で、画面単位にユーザーに誤りを指摘するようになっていて、校正メッセージを見ながら対話的に文書を更新できるようになっている。

KWIC表現では、キーワードや文脈を、その文を単位として更新できる。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部分への切り換えにより、ソース表現上で行なわれるようになっている。KWIC表現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れの検出である。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われる。この特徴を生かして隣り合ったキーワードを調べていくような校正規則が作られている。

ユーザーインターフェイスの提供するその他の機能としてテキスト・コンパイラと辞書サーバーがある。

テキスト・コンパイラ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。テキスト・コンパイラ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。このような付加情報としては[2]で考察しているようなものが有用であると考えている。テキスト・コンパイラの出力結果を図7に示す。ユーザーはこの出力を見ることにより、誤変換を見つけたり、言回しを変えた方がよいような表現について、推こうしたりできる。

```
BCRITAC  PRINT    A1  V 255  TRUNC=255 SIZE=897 LINE=332 COL=1 ALT=2
  321
  322    * ERROR 31 on line 23 position 6 : 『機械翻訳システム野ためには』
  323        適当でない漢字が使われています。
  324        誤変換ではありませんか？
  325
  326    * ERROR 33 on line 40 position 35 : 『実用かと』
  327        未変換の可能性はありませんか？
  328        この「か」は漢字で書かれるはずではありませんか？
  329
  330    * ERROR 31 on line 41 position 12 : 『束変には』
  331        適当でない漢字が使われています。
  332        誤変換ではありませんか？
  333
  334    * ERROR 15 on line 25 position 10 : 『たてられる』
  335        受身のような表現が使われています。
  336        他の言い回しができませんか？
  337
  338    * ERROR 17 on line 24 position 28 : 『よって』
  339        1つの文の中に同じ言葉が繰返し使われています。
  340        他の言い回しができませんか？
  341
  342
```

図7. コンパイラ出力

```
 CRITAC   KWIC     A1  F 116  TRUNC=116 SIZE=1138 LINE=575 COL=1 ALT=0
+ ------------------------------------ +
|  更生 :   こうせい          :000820  |
|  公正 :   こうせい          :000060  |
|  硬性 :   こうせい          :000020  |
|  恒星 :   こうせい          :000020  |
|  抗生 :   こうせい          :000020  |
|  構成 :   こうせい          :000020  |
|  攻勢 :   こうせい          :000020  |
|  後世 :   こうせい          :000020  |
|  校正 :   こうせい          :000820  |
|  鋼製 :   こうせい          :000020  |
|  厚生 :   こうせい          :000020  |
|  高声 :   こうせい          :000020  |
+ ------------------------------------ +
  575              という表は『    構成          』に変えられる。
  576                            構成          校正：呼び出し呼出し：
  577  ようであるが、このような    校正          については、英文では既
  578    日本語の性質を考慮した    校正          方法の研究が重要となる。
  579        最近になって、文書    校正          のツールとして、すでに
  580  ような構造化文書のうえで    校正          知識をルール形式で実現
```

図8. 同音語の出力(SQL/DS)

辞書サーバーは、ユーザーがエディタから対話的に同義語・同音異議語や関連語等を検索できるように、関係データベースシステムSQL/DS[14]の関係として国語辞書を管理したものである。ユーザーは校正機能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、言回しを変えたいときに辞書から同義語を引いてみたりすることができる。現在のところでは漢字の基本語約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8 参照）。辞書サーバーへのアクセスは、あらかじめ用意された同義語等の検索以外にも、質問言語SQL を用いてユーザーが動的に表現できる。これは通常の電子辞書と違う、関係データベース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ーに提供することが可能となる。

[校正用知識ベース]

CRITACの校正知識は2種類あり、1つはソース表現上で、もう1つはKWIC表現上で働く。ソース表現上で働くルールというのは、文節、自立語、付属語に関する述語で、ミスタイプや不均一な語の使用などをみつける。KWIC表現上のルールは隣接するキーワードの関係をみたり、同音異議語の使用状況をみたりするのに使われる。以下で、この校正知識について詳しく述べる。

3.2 CRITACの校正知識

CRITACの校正知識はソース表現上で働くものと、KWIC表現上で働くものと2種類あることについては前にも述べた。ここではその各々について例をあげて説明する。

[ソース表現上のルール]

ソース表現上のルールは、複雑であい味な名詞句に警告を与えたり、ミスタイプや不均一な語の使用をみつけたりする。日本語の文法はある種の助詞を繰返し用いて一つの長い名詞句を作ることを許してしまうため、一度読んだだけでは判らないような、係り受けのあい味な表現がある。文章中のそのような個所に警告を与えてユーザーに書き替えを促したり、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ーに尋ねたりするのである。ここでは不均一な語の使用を見つけて警告するルールを例に挙げて説明する。

<例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:上記以外の場合→ 警告する
case3: 上記以外の場合→ 何もしない

この規則により、例えば文書中のあるパラグラフでは「編集画面」と使っている語が別の場所で「編集用画面」と使っていた場合、「用」は漢字接尾語なのでcase1により警告を受ける。また同じく、「編集のための画面」もcase2-2により警告される。これは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は同じ重みをもつ対等な語として扱われていると考えられる。つまり「編集および画面」「編集または画面」だが、このような場合は「編集画面」とは全く異なる概念を指しているわけなので、警告はしない。そしてＣが付属語で等位接続詞以外の場合（例えば「編集のた

めの画面」など）は、「編集画面」と同じことを表現しているとみなし、表記の統一のために警告することにした。

次にもう一つ、誤変換の例について見てみる。

<例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種類の語基とする。ここでCはある助詞と読みを同じくする漢字である。

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

このルール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ルールである。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、もちろん「・・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「、」によって繰り返されている可能性もあるが、ここではCは一つの独立した語基と仮定しているので、そのような場合はない。とすると、「野」は単独に一語で現れてそこで「、」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。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」が単純な名詞であるとすると、接続不可能なので同じく警告する。それ以外のたとえば「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、これについては何もしない。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、つまり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」で始まる外来語とも考えられ、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」が接頭辞的に働くことになり、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はずなので警告する。

［KWIC表現上のルール］

KWIC表現上のルールは、正書や読み等で順序づけられたキーワードに対する述語である。日本語文書によく現れる、誤変換や表記のゆれのほとんどは、適当なキーワードの順序を与えることにより、KWIC表現のうえで隣接したキーワード間の関係として捕えることができる。例えば、例1のソース表現上のルールは、キーワードが正書の順に並んだKWIC表現のもとで次のように表わせる。

<例3> KWIC表現のルール：
今正書の順に並べられたKWIC表現の第i行のキーワードをK(i)、K(i)に後接する付属語連鎖をKf(i)、そのあとに現れる自立語をKj(i)と書く。Kf(i)は『による』や『の』といった単純なものか、接続詞『および』等からなるものとする。直観的には、例1のX、Y、Cはそれぞれ K(i)、Kj(i)、Kf(i)に対応する。

case1: ある接尾語Cがあり、
K(i+1) = K(i)C、Kf(i)とKf(i+1)は空、およびKj(i) = Kj(i+1)
が成り立つ。→ 警告する
case2: K(i) = K(i+1)、Kf(i)が空、
Kj(i) = Kj(i+1) で、Kf(i+1)が空でない場合
case2-1: Kf(i+1)が接続詞の場合→ 何もしない
case2-2:上記以外の場合→ 何もしない
case3: 上記以外の場合→ 何もしない

```
CRITAC   SOURCE   A1  F 72  TRUNC=72 SIZE=211 LINE=84 COL=1 ALT=0
 79
 80  以上述べてきたように、パターン・マッチングとバック・トラッキング機能を
 81  もつＰｒｏｌｏｇの採用により、校正知識を宣言的かつ単純な形で実現するこ
 82  とができる。また、文書そのものもＰｒｏｌｏｇのファクトで扱かれているこ
 83  とで、校正知識の開発、修正のための良い環境が与えられる。今後、簡単な文
                                                    15========
 84  法を備えることにより。果たしてそのくらい文法的なチェックが可能になるの
       23================
 85  か、そしてそれによってどのくらい校正能力が向上するのかをみたうえで、そ
 86  のあとの意味処理につなげる方法を検討するつもりである。
 87
 88  ワード・プロセッサの普及にともない日本語文書の作成は近年急速に機械化さ
```

図9. 校正ルール実行画面

KWIC規則のもう1つの利点は、誤りが隣接したキーワードの上で表示できるため、人間にとって非常に理解しやすいことである。例1のソース表現の規則では、文書に点在した誤りを検出しても、それを分りやすく示すのは容易でない。　同様にして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意義語を検出する規則を得ることができる。この場合はキーワードが読みの順に並べられなければならない。図8に読み順に並んだKWIC表現で、同音意義語を検出した例を示す。誤変換はこの同音意義語がさらにある条件を満足した場合として表現できると考えている。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正書や読みを組み合わせることにより、かなりの範囲で誤りを指摘できると考えている。現在のところルールの数は30個で、種類は表2に示すとおりである。校正ルールの実行の様子を図9に示す。

表2．ルールの種類とカテゴリ

カテゴリ	ルール番号	ルールの説明
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
	110	ことば使い
	111	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　校正ルールのデバッグ

CRITACの校正ルールのデバッグには、Prolog の視覚的デバッガPROEDIT［12］［13］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、ユーザーはPrologプログラムのエディット、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、我々は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の実行モニターは、図10に示すように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、ユーザー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出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つの校正ルールが正しく文章中の誤りを見つけることができるかどうかを、確かめながら、ルール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よう。この例は文の終わり方をみて、誤った句読点の打ち方をしているもの（ミスタイプ）をみつけるルールである。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語基、PはXの直後の句読点とする。

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
case3: 上記以外の場合→ 何もしない

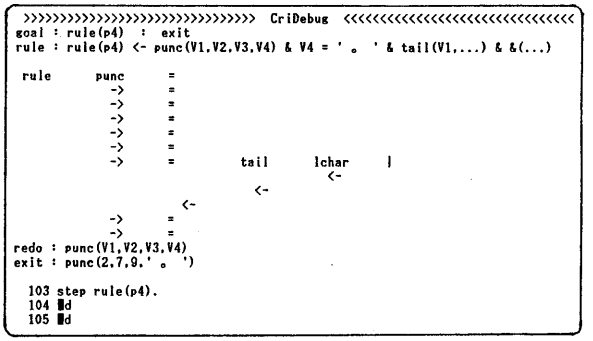

図10．PROEDIT -1-

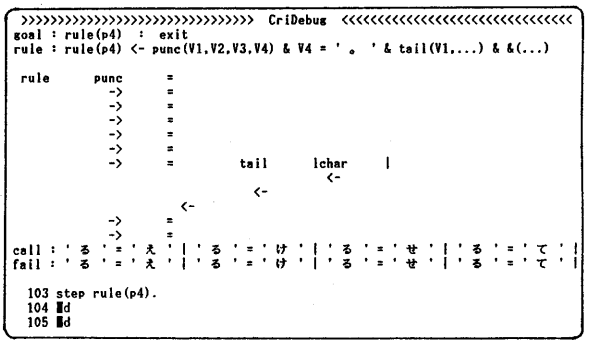

図11．PROEDIT -2-

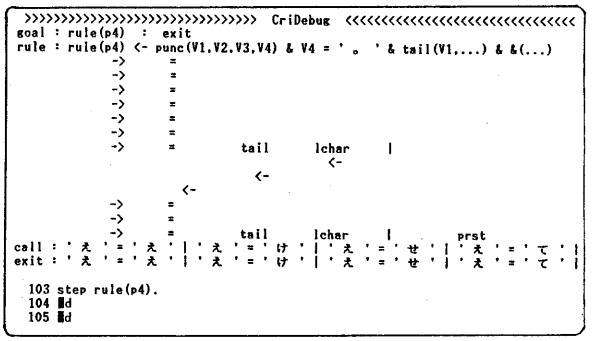

図12．PROEDIT -3-

このとき、このルールを試行するテキストには1ケ所「・・を行なえ。・・」という部分がある。このルールが正しく働けば「え」の後の「。」は「、」の誤りではないかと警告することができる。図12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」の段でもないことによって失敗している様子がわかる。そしてどんどんサブゴールの実行を進めていって、最後に図12の画面中央に成功したサブゴールが示され、このルール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。

ルールが正しい場合についてのみ、説明したが、ルールに誤りがある場合、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（上の例ではたとえば、「・・を行なえ。・・」の部分）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なおして再びPROEDIT 上で実行してみる。この操作を繰り返すことにより、ルールが整備されていく。

5 実験結果・評価

5.1 誤りの分類と調査

我々はワード・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われる誤りを第4節で述べたルールがどの程度検出することができるかを調査してみた。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変換入力を行ない、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、2文書とも情報工学分野の論文である。文書Sも文書Tも筆者のうちの1 人が入力したもので、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3 に示す。

表3．サンプル文書に現われた誤り

	文書S	文書T	文書S
誤変換α	24	20	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
変換されすぎ	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、いわゆる同音異議語による変換誤りで、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」を打とうと思っているのに、左手中指の「e」を打ったことによる誤り)や「コーコ←コード」（本当は左手中指の「d」を打とうと思っているのに、右手中指の「k」を打ったことによる誤り）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思いのほか少なく、全体で3種しか現われなかったが、これらのうちの2つはカタカナ語の後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。

『変換され過ぎ』というのは、長い単位で変換したときや、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。

表記のゆれは、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書かれる場合にも起こりやすい（例「PROLOG」と「Prolog」、「インターフェース」と「インターフェイス」など）。

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(例「辞サーバ」←「辞書サーバ」)。

5.2 実験結果と評価

構造化文書では、各自立語には読みがふられ、自立語や付属語の品詞といった情報も付与されたProlog の節が与えられている。この'読み'と段落(章、文書)内の統計的情報、そして文献［9］でも述べた、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われていたときに警告するようなルールを働かせることで、誤変換の内、現在48.9%は検出できる。そして、例えば文書中「多い」という言葉を1回しか使用せず、しかもそれが「覆」という漢字になっていた場合、あるいは気付かずに同じ語を何度も同じように誤っている場合の確率は全体の誤変換αの内のごくわずかであろうと予測するためである。

誤変換βについては「救って←作って」や「苦情←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入力で、「tsukutte」と打つべきところを「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「kouj(y)ou」の最初の「o」が抜けて「kuj(y)ou」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［4］、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、日本語ワード・プロセッサでは、一番多くの場合、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段階の変換が行なわれており、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かなり難しい。CRITAC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。

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、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個の内の1個については正しく文節切りできない。残り7個については文節切りは正しいので、文節内の構造をうまくとらえられれば辞書を引くことで辞書の正書を取り出せる。一方未変換βの内文書Tに現われた「ひょうげんん←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に現われた「ねべてきた←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カ所にしか現われず、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。

ミスタイプについては英語で行なわれているようなスペルチェックを応用することで将来は正しい語を推定できるようにしたい。

文字種キーの押し忘れによる誤りはCRITAC では、付属語接続検定の折に接続に失敗し、かつそのような自立語(ワード・プロセッサニオイテ、チェックサエナク)も辞書にないことから、検出し警告することができる。

以上の結果を、現在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13参照）。

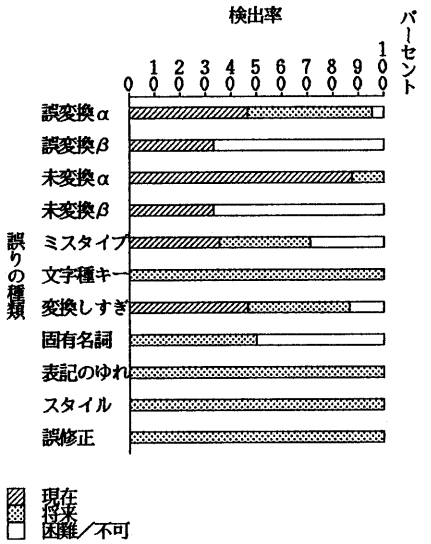

図13．誤りの検出率

6 あとがき

本稿ではワード・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し、日本語文書の校正を支援するための試作システムCRITACについて述べた。ここでは、入力された文書をPrologの節に変換して構造化文書として取り扱い、校正機能はエディタから呼び出されるPrologプログラムとして実現した。

［謝辞］

本研究を遂行するにあたって、日本アイ・ビー・エム株式会社の幾多の同僚、先輩から助力や助言を得たことに感謝する。特に校正ルールのインプリメンテーションおよび討議に加わって頂いた西野哲朗氏、沼尾雅之氏、丸山宏氏、尾関正俊氏に感謝する。

［参考文献］

1. 石井ほか：日本語の漢字・用字の校正のための知識、ICOT Technical Report：TR-0092、1985。
2. 牛島ほか：日本語文章推こう支援ツールのプロトタイピング、ソフトウェア工学、40-8、pp43-48、1985。
3. 大河内：仮名漢字変換のための形態素接続規則、東京サイエンティフィック・センター・レポート、N:G318-1560-1、19281。
4. 角田：多層テキスト構造を持つ日本語エディタ、pp75-84、第27回プログラミングシンポジウム、1986。
5. 絹川：高品質日本語文章作成支援機能の一考察、pp1389-1390、情報処理学会第31回全国大会。
6. 鈴木：漢字かな混じり文に現われるひらがな列の文節推定方法について、pp.1383-1384、1985。
7. 武田、藤崎：統計的手法を用いた漢字複合語の短単位分割、自然言語処理研究会、48-2、1985。
8. 武田ほか：日本語文書作成支援システムCRITAC、pp.1691-1692、情報処理学会第32回全国大会。
9. 武田ほか：日本語文書作成支援システムCRITACの校正知識、pp1693-1694、情報処理学会第32回全国大会。
10. 建石ほか：DictionとStyleの日本語化について、pp1381-1382、情報処理学会第31回全国大会。
11. 長尾(監修)：日本語情報処理、電子通信学会編、1984。
12. 沼尾：PROEDIT-A Screen Oriented Prolog Programming Environment システムの構成と実現、pp403-404、情報処理学会第31回全国大会。
13. 沼尾：Prologの視覚的デバッガ、記号処理研、32-1、1985。
14. Cherry,L.L.：Writing Tools-The STYLE and DICTION Programs、Bell Lab. Technical Rep.、No.9、1980。
15. IBM Corp.：SQL/Data System Concepts and Facilities、GH24-5013。