北海道大学工学部研究報告
第167号（平成6年）

Bulletin of the Faculty of Engineering
Hokkaido University, No. 167 (1994)

自然言語処理研究向き文書エディタの作成

堤　正勝[t]　荒木　健治[tt]
宮永　喜一[t]　栃内　香次[t]

（平成5年8月31日受理）

Impremention of Document Editor for Natural Language Processing.

Masakatsu TSUTSUMI Kenji ARAKI
Yoshikazu MIYANAGA and Koji TOCHINAI

(Received August 31, 1993)

Abstract

In this paper, we discuss about the implementation of a document editor which has three purposes.

The first purpose is following. Voice recognition is a simple way to input documents. However its method is not established. On this problem, we had supposed a method to recover original character strings from voice-recognized erroneous strings. This method is somewhat statistic one. So we need statistic information of morphic words from a large number of existing document files. In order to get the statistic information, we cannot use full-automatic morphic analysis, because those files must have some errors. Therefore we propose an editing algorithm, and implement it as a half-automatic analizer.

The second purpose is following. It is destiny of voice recognition to include error. So, a voice input system needs to provide not only the voice recognition, but also keyboad input correct errors. In addition, using the editor's function of gradual entry enables to cope with unentried words.

The third purpose of the editor is following. Since edited error-free documents can be useful for various studies of natual language processing, we consider to prepare a morphic saving format. And we refer to its description and grammar.

1．はじめに

1.1　ビジョン

計算機には多くの機能があり，基本機能は入力，変換，出力，蓄積，通信である。近年，あらゆる分野において計算機の利用はますます一般的になってきており，従って，マンマシンインターフェースの立場から，上記基本機能を統合してより使いやすくすることが強く望まれている。

t　電子工学科　電子機器工学講座
tt　北海学園大学工学部

1.2 入力機能，入力変換機能

現在の計算機の入力手段は絵や画像ならスキャナー，音声ならマイクロホン，文字ならばキーボードであろう。これらの入力手段は，そのままのトランスペアレントな形であれば，殆んどハードウェアの機能で実現され，ソフトウェアはその転送と名前の管理を行なうのみである。しかし，それらが情報として役立つには，その内容について計算機が何らかの介入を行なって処理できることが重要である。例えば音声認識を行なうことにより文書化することができる。書かれた文字をスキャナー等で入力した画像データならば，文字認識によりやはり文書化できる。即ち，一旦文字として記号化すれば，人工知能関連の方面への適用が期待でき，また，著しい省スペース化が行なわれ，記憶域の効率的利用に繋がる。また画像データからは，別の形の認識も可能であろう。例えば，形の配置を表現できる言語がそこから得られる情報であっても良い。また文書として得る形では，図や表を整形する機能が既に実現され市販されている。この様な情報の加工，抽出ができる段階を入力変換として考える。

このような誤りを含む認識処理の結果を修正するには二通りの形態が考えられる。一つは認識結果をファイルにセーブし，エディタで読み込んで修正する方法。もう一つは，認識処理そのものをエディタ内部の入力変換機能として装備する方法が考えられる。ここで，認識率の向上を図ることを考えた場合，エディタの修正操作に関する情報を利用することが考えられるが，この情報を効率良く認識率に反映するのは後者である。修正情報を内部でインタラクティブに利用でき，認識のための登録情報を逐次で更新できる。このことは情報の有効利用において重要である。前者は修正情報が即座に利用できないばかりではなく，一度エディタ外部形式に変換されて再度修正情報の抽出を行なって認識のための登録情報を更新することになるので無駄が多い。

1.3 自己表現や対話の手段としてのエディタ

自分の表現したい内容を忠実に映し出す鏡としての役割を果たせる機能を持つエディタを，究極の目標に考えている。現在の計算機システムに許される表現能力といえば，文字，音声，画像である。とくに，画像が追加されたことにより，昔の計算機に比べて格段に表現力が増したといえる。しかし，その画像自体は計算機には何の意味もない単なるデータである。こういった情報を人間が文書を中心に結び付けてやることにより何らかの意味の創造を行なう。エディタはそういった作業を助けるものでありたいと考える。

遠隔地へのコミュニケーションは種々ある。たとえば電話のように，音声に関して透過な通信を行なってくれるものもあれば，手紙やメイルシステムのように，メッセージを蓄積し一括して送出するものがある。人と人とのコミュニケーションの間に機械が介在してインターフェースを行なう。その際，例えば送信機能に翻訳を介在すれば国語に依存しないコミュニケーションが可能であろう。我々は自分の意図を一度計算機に蓄積し，推敲したものを相手に伝えるものを目標としている。エディタはそういった作業の助けになるものでありたい。

1.4 文書入力の現状

現在のエディタは多くをキーボードから入力するために非常に不便でありもどかしいものがある。全くの初心者ならば，入力している間に折角のアイディアを忘れてしまうなどということもままあるのではないだろうか。入力に多くの時間を要し，手間が掛かる。手数の少ないキータッチで複雑な操作をすることは実現されているが，問題の本質は操作よりも文書そのものの入力にある。

例えば，人の話しを聞きながらタイプする場面を考える。かな，あるいはローマ字でタイプするのみなら次々と話されるスピードに間に合いながら，入力することのできる人もいるかもしれない。しかしながら，仮名漢字変換を行いながら入力する場合を考えると，候補を選択し確定する作業が必要になるため，話す速度での入力は不可能であろう。

また，キーボードによる打鍵入力は一定の姿勢を強いるので長時間作業に向かないことや，ＣＲＴ表示画面からの有害な電磁波による人体への悪影響なども指摘されている。

1.5 音声による文書の入力

このように，音声による文書入力が望まれて久しい。しかしながら，音声認識による入力は，精度や語彙の点で単独で正しい文章を得る手法は確立されていない。

一方，音声だけで全ての入力が済む場合というのは非常に限られると考えられる。例えば句読点の挿入等の様にある意味で人間の編集作業が必ず必要となるのが文書の本質であろう。

この様なことから文書作成における音声認識の効果的な位置づけは，まず音声で発声し，それを計算機が認識し，表示する。人間はそれを確認し，誤っていれば修正するといったような形態が最適な的なスタイルとなると考えられる。

人間が手で修正する作業も，同じ修正を何度も繰り返すようでは，やはり辛い作業といえる。そこで，音声の誤りの傾向の統計を取り，それを教師情報として，認識部にフィードバックして学習させて以後の誤りを減少させる機能を持たせるとよい。また，人間の修正作業から，音声認識部の本質的な誤り以外の修正のパターンを蓄えることにより，修正箇所の候補を挙げることができるので，誤りを探す手間が減少し，修正のパターンを適用するかどうかだけの判断を入力とするのでワンタッチで操作できるはずである。

なお，音声認識手法としてワードスポッティング法と，音素のセグメンテーションを行なう手法があるが，必要となるデータの規模を実用的なレベルに考え，また，あとの処理の都合を考慮し，後者を想定する。

1.6 研究者向き内部表現

現在のところ，認識により得られる精度は甘く見積もっても９割程度で，あり，その多くが何らかの制約のもとにこの数字を得ている。音素や単語のレベルの認識だけではこれ以上の精度は求められないといっても過言ではない。これに対し何らかの統計情報や意味情報，文脈情報などの上位の情報を用いて原文を求めようとする自然言語処理の研究がある。これらの研究では，自然言語の性質に関するさまざまな情報を調べる必要があり，またそのために形態素に分けてあると都合が良い。そのために形態素解析処理を行ない，分割を行なったデータを対象として実験を行なうのが普通であり，研究者はそのデータをロードする部分を記述する必要がある。本エディタは，その様な入出力の記述の手間を省き，ポインタの連鎖で表された内部表現をもち，その上で，データを処理するプログラムを追加することで実験が可能である。また，本システムはエディタなので，実験中に本質的に関係ない問題がテキスト中に含まれている場合にはプログラムからエディットモードを呼び出し，その部分を削除して実験を続行すればよいという利点がある。プログラムの追加は，システムにリンクすればよく，未定義のキーから呼び出せば良い。我々は，音声認識の誤りを修正するアルゴリズムを求めるために，このような機能を持つエディタが必要になり，現在作成している。

なお，音声認識によって得られる誤りを含む文字列を，正しい文章に直す処理を復元処理と呼

んでおり，復元処理が完成すれば，このエディタに組み込む方向で考えている。この復元処理を次章でふれておく。

2．認識文書の復元

音声や文字認識によって得られる文書には，認識の本質上，必ず誤りが含まれることを想定しておかねばならない。つまり，誤りがあるものとして，正しい文章を得ようとするフェーズを入力機能の一部に設ける必要がある。この問題に対して我々は，べた書きかな漢字変換のために開発した，多段階分割法[1]をさらに音声認識によって得られるような，誤りを含む文字列に適用した多段階分割復元法[1]を提案し，その有効性を確認している。そこでまず，この手法を簡単に説明しておく。なお，ここで音声認識により得られる誤りを含む文書を認識文書と名付けておく。

2.1 多段階分割法の概要

現在かな漢字変換手法として，２文節最長一致法など，幾つかの手法がパーソナルコンピュータのフロントエンドプロセッサとして実用化されているが，左から順に変換して行くため，変換精度にはあまり良くない。長い文章を一括変換しようとすると，時間がかかるばかりでなく，誤った箇所を修正するのが多大な手間となり，結局単語単位で入力した方が早い。

この問題に対し多段階分割法では，単語を確実性の高い語から順に当てはめてゆく手法をとるので，より正確な変換が行なえる。確実性の高い語は数が少なくパターンマッチングが容易である。また，ひとたび語が当てはまると，入力行はそこで二つに分割され，より短い未確定部分列の処理を行なうことになるので，単語が当てはまる毎に，当てはめのコストが著しく減少してゆく。こうすることにより，確実性の低いレベルでの負荷を軽減している。このレベルでは，単語を特定するには，拘束条件として，単語の尤度として，頻度，履歴，誤り数や，連接情報を用いている。

語の確実性に関しては，文献独立キーワード，文献依存キーワード，キーワード拡張処理，カタカナ語，助詞候補，一般語，助詞確定，接辞，一字漢字語の順としている。なお，全ての語に優先順位をつける手法も現在検討中である。

2.2 多段階分割復元の概要

かな漢字変換では入力された文字列は正しいとして扱う。しかしながら，認識文書では誤った文字を含む入力文字列を扱わなければならない。

認識文書では，文字単位の誤りがある。これを正しい文に復元するためには，まず，文字レベルで正しいと考えられる文字を統計情報として前持って取得（学習）しておき，正しい文字の候補を，幾つか挙げる。対象は文字列であるから，その候補の組合せ（パス）は膨大な数に上る。この膨大なパスをｎ文節最長一致法で検索することは実用上問題がある。我々はここに多段階分割法の拡張し，適用の有効性を見た。多段階分割復元法では数少ない確実性の高い語から順にを当てはめることにより，そのパスの可能性を格段に減らした。そうすることにより確実性の低い語の段階での検索すべきパスの長さを短くし，計算コストを著しく減少する。

3．自然言語処理指向文書エディタ

3.1 はじめに

そうした自然言語処理に関する研究では，前処理として形態素解析が必要となる。著者らはこ

れまで，形態素を入力の段階でわかち書き（ＫＫＨ表記）[2] を行なったり，文書データの形態素解析の結果を編集して未知語や解析誤りに対処してきた。本エディタは第一に，形態素解析と同時に未知語の登録を行なって作業効率を向上させること，第二には，解析結果を効率よく使用することを目的としている。

形態素解析の結果は主記憶上の辞書項目へのポインタの列として保持されるので，その後の形態素に関する実験手続きをエディタに組み込むことにより容易に実験ができる。本エディタは，以上二点について自然言語処理の環境を提供する。

3.2 エディタの構成

本エディタはフルスクリーンエディタの形態をとり，入力は二系統ある。一つはキーボードから入力するものである。行管理部を通じて入力と同時に形態素解析が行なわれる。形態素解析アルゴリズムの不都合と未知語対応のために半分ち書き入力を行う。もう一つは，既に編集された文書データを入力するもので，文書の読み込みと同時に形態素解析を行なう。入力文書の中に未知語が存在すると，逐次切り出し範囲と読みを要求する。解析結果が複数存在し，一意に決定できない場合は，その部分をレジスタに格納して最初の候補に設定しておき，会話モードにおいてそれらの部分を最少の操作で次々と検索，編集できるようにした。

形態素に分割された文書は行管理部において形で管理される。文書要素は，単語と非単語（未解析部分）に分けられる。表示管理はキーボード入力と各種管理部を結び付けてそれぞれの機能の呼び出す。ファイルのセーブには形態素分割の情報を残すため，形態素を保存する形式を用意している。これについては次章（4.6）に述べる。文書は形態素列として表される。

3.3 単語管理と行管理

辞書は単語管理において起動時に主記憶にロードし終了時にセーブする。単語文字列は単語管理の下に文字列管理を設け，同じ文字列が重複して登録されないように，検索してから登録している。これにより，同音語，同表記語は，同じ文字列を指すポインタとして管理されるため，おなじ文字列であるかどうかはポインタを比較するだけでよいという利点をもつ。

入力したテキストは，行に分割され，形態素解析により更に単語に分割する。単語の定義は資料[3] に従った内部表現で管理する。大部分の用言は子音で終るので叙述表現[3]（一般の助動詞に相当）の並びとともに表示部で仮名に変換している。非単語文字列は，テキストバッファ内の文字列へのポインタと長さの情報で未処理の部分として管理される。なお，記号や，罫線文字，スペース，句読点，改行等の文要素も語として登録してる。これにより音声による文書入力および編集にも対応している。

行は，単語へのポインタのリストで構成される。また，ファイル通常のエディタと同様には，行のリストとして管理される。

3.4 機能追加について

本エディタはＣ言語で書かれたプログラムを本体にリンクすることで機能の追加が容易に実現できる。具体的には，校正処理，翻訳処理，また，単語にファイルをリンクすることによりハイパーテキストエディタなどが考えられる。なお，我々は上述のように復元処理をアルゴリズムの追加を考えている。

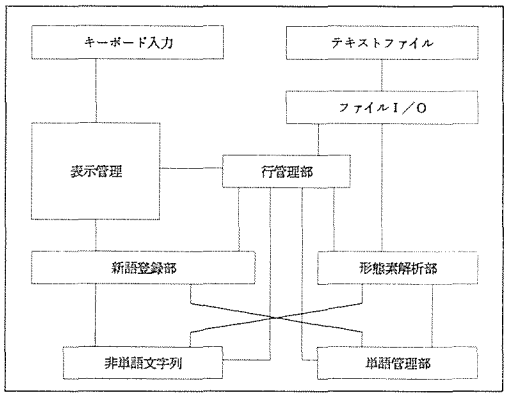

図１：エディタの構成

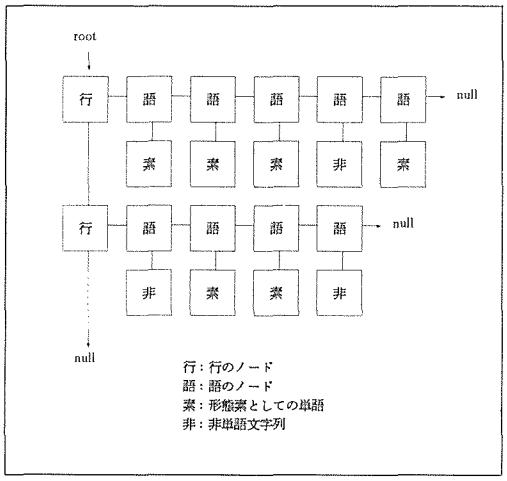

図２：行と単語の管理

４．単語管理について

前章で紹介した内部構造において単語をどのように取り扱うかを以下で説明する。

4.1 辞書の構築とバイナリ辞書

辞書を構築するにあたり，辞書の書式，辞書のアクセス，辞書の編集に関して，効率化，扱い易さなどに配慮している。

これまで我々の研究室で採用してきた辞書としては，ＰＬ／Ｉの言語仕様に合わせ，１レコードのフォーマットを決めて，同じ形式で入出力を行なう方法をとった。この方法で辞書を構築すれば，そのまま可読，編集可能な辞書ができる。

これに対し，UNIX 環境下でＣ言語で記述擦る場合，それ以外に種々の方法をとることができる。その意味では，Ｃ言語で記述する場合，記述のしかたによってかなり効率化が行なえる。

辞書を，主記憶のイメージのままにセーブすれば，変換時間のオーバヘッドが少なく辞書のロード，セーブの効率が向上する。この機能をサポートしたことにより起動，終了時の待ち時間が著しく減少した。具体的には，辞書を文字列と管理情報に分けて扱っている。文字列部分は単語を全て一つのバッファにナル文字を区切り記号として詰め込んで行くき，以後の単語文字列の参照は全てこのバッファへのポインタで行い，表示管理部や入出力管理部から参照することになる。主記憶イメージは可変長レコードの形式でセーブするために，詰め込みの際にバッファのサイズを管理している。単語へのポインタはユニークな値なので文字列ＩＤとして用い，文書中の単語文字列の一致検出において文字列を直接比較する代わりにＩＤの比較を行なうことで効率化している。一方辞書としての情報として，読み文字列と，表記文字列の対応もセーブしなければならない。また，対応づけを行うことによってはじめてそれが語として成立し，そこに品詞などの付加情報が必要となる。そこで，単語はひとつの固定長のレコードとして管理できる。つまり，単語レコードの長さの可変な要素を排除した形式をとる。このレコードも，レコード数を先頭にして，レコードの配列ごと，一括してセーブできる。このレコードは文字バッファと同様，出現順に登録して行くので，そのままでは検索には適さない。このため辞書の検索情報として，辞書の

エントリ番号を読みや，表記でソートした配列も同時にセーブする。現在，辞書アクセス機能として二分木検索を行なっているため，木構造の情報をセーブしている。なお，このイメージをセーブする形の辞書をバイナリ辞書と呼ぶ。

効率化を追求したこの辞書は，しかしながらこの形式は可読ではないので指定により可読な状態での入出力も指定により行なえるようにしてある。可読形式の場合，ソーティングして出力すると，編集する人間にとって都合がよい。しかしこのようにすると，次の入力時点に辞書の偏りが最悪のリスト構造になり，木構造の意味がなくなってしまうという問題がある。この問題に対処するアイデアについて，木構造を動的に繋ぎ替える手法を提案した[6]が，入力時に項目数が分かっている場合には，ノードの深さが分かるので，均一な木を作ることが可能である。簡単のために後者を採用した。検索木は，読みによる検索と，表記による検索の二通り用意している。

辞書のアクセスは上記のソートキーや二分木を通じて行なう。新語登録は，上記の様に文字列での比較をポインタ比較に置き換えるため，登録時にユニークな文字列であることをチェックする。そして，その上位の機能として同音語，同表記語を管理している。

4.2 辞書のロード

本エディタは辞書を起動時に読み込んで常駐する。辞書名は，リソースファイルに名前一覧を書いておくことにより複数の辞書を容易に指定できる。辞書は空でもよいが，非保存形式テキスト（後述）の解析ができないので逐次登録することになる。入力辞書は上記バイナリ辞書形態の他，括弧形式（後述）や，ＫＫＨ辞書形式に対応した。

4.3 語彙の拡大

辞書の編集に関しては幾つかの方法を考えている。そのひとつに可読形式の辞書を編集して読み込ませる方法がある。これは最も基本的だが，資料としては各出版社から出ている国語辞書などを用いることになり，使わない語の登録が大半を占めるので，効率的な方法ではないと思われる。

第二の方法として，本稿のような原稿の編集中に必要な語を逐次登録して行く方法がある。これは実際有効な方法で通常のかな漢字変換システムがユーザに提供している方法でもある。

本エディタではある程度未登録語の出現率が減ってた段階において，形態素解析機能を利用し，既に存在している文書から単語を抽出することを考える。もちろん，そのまま登録してしまうと誤りの原因になるため緩衝機構として単語候補プールを作成して，頻度，誤り率などにより評価して，辞書形式の単語一覧を作成する。最終的な登録は，その一覧を目視にて確認をしたうえで辞書として使用するようにする。

これら登録の問題に対して，一方で辞書中の単語を削除したい場合が考えられる。ここに問題が生じる。登録に関しては，出現するたびに辞書に追加して行けるが，削除については注意すべきことがある。削除しようとする語はそれまでに編集した文書の何処かに存在する可能性があり，削除された語は，その文書をロードした時に出現し，

上記の登録機能に従い，再度登録候補とされてしまう。

そこで，単語削除は，実際に取り除いてしまうのではなく，不使用のフラグを立てて管理し，文書再ロードの際に，該当箇所の編集をするための，マーキングを行ない，編集者が訂正をする。

なお，入力ミスによる誤登録もあり得るので，登録直後や，文書に使われない語は削除して構わない。

4.4 辞書項目の追加

本エディタは研究環境を提供するものなので，C言語によるアルゴリズムの追加が容易であるように内部のデータ構造を考慮した。単語文字列や，基本的な単語構造体（単語レコード）はすべてポインタで管理されているので，新しいレコードを作成する時はそれらを適当に編集すればよい。ポインタで扱うことはメモリの複写の必要がないことを意味する。基本機能として単語検索をサポートしてある。

概念などによる単語のグループ化なども可変長ポインタ配列を動的メモリに作ることで簡単に実現できる。追加項目に関する入出力ルーチンは必要に応じて書くことになるが，基本機能として，レコード情報を文字列に変換するルーチンを提供しておいたので，可読形式への変換もそれほど手間ではない。

4.5 辞書の内容

文書の内部表現は通常のテキストファイル以上の記述能力を必要とする。辞書はエディタで使用される内容の全て，入力されうる全てのものを表現するための要素を内容として持たせる。

現在目標としている記述力の範囲は，

ページ記述言語（ＴＥＸなど）の内容や，文書に関する実験のために，特定の部分にマークをつけたりすることもできるような機能もサポートする。音声タイプライタのような用途を考慮し，入力操作のようなメタな機能も記述する。なお，音声による入力を考えるので全ての要素に読みを与えている。このような目標に対して以下のものを考えている。

(1)通常の単語で，形態素として扱われる。(2)句読点や鍵括弧などの記号，数式などで一定の文法がある。(3)文書の書式を制御する要素。例えば，改行，改頁，行組など。(4)その他として，マクロ，文書解析用のマークや，フラグの類。

編集したテキストは，これらの要素に分割された表現形式で保存する。これを「保存形式」と名づけ次章の形式で扱う。

4.6 保存形式

保存形式は読みの順にソートして同一のものを含まない形にすれば可読形式辞書の一つの形態であり，また，語の出現順にセーブすればテキストの保存形態である。これを以下のように定義して使用している。

一つの要素あたり下の形式を用いる。
(〈読み〉/〈表現〉/〈属性〉[/〈補助〉]…)

読み：ローマ字で記述する。
表現：漢字や，印刷制御の情報。
属性：要素の種類を示す。形態素ならば品詞。

文書はこの要素を繰り返すことにより，成立する。まず，各々の例を以下にしめす。

(1)の例：

(situ/質/n) (no/の/Z) (yok/良 k/j) (-ki/い/V)
(kana/かな/n) (kanzi/漢字/n) (henkan/変換/s) (wo/を/Z)

(2)の例：(＊は出力すべき版組ソフトのソースに依存する。)

(ee/E/SN) (ekooru/＝/SN) (emu/M/SN) (sii/C/SN) (zizyou/＊/SN)

(3)の例：

キーボードから読みなしで入力するもの。

(#/title) (#/paragraph) (#/newline/^J) (#/newpage/^L)

読みを定義する場合，以下の様になる。

(kaigyou/(#/newline/^J)/C) ……改行

(kaipeezi/(#/newpage/^L)/C) ……改頁

(danraki/(#/paragraph)/C) ……印刷制御に読みを与える。

(4)の例：

(#/flag/A) ……フラグ定義

(#/paragraph) ……印刷制御

(#/def/.PA/(#/paragraph)) ……マクロ定義

(#/def/.FL/(#/mark/A)) …… 〃

(.PA)……マクロ使用

なお，辞書内部でユニークな読みがユニークであれば，次の様に読みだけで書いてもよい。また，同音語が存在しても最後に参照されたものと同じであれば，再度全部を記述する必要はない。そこで実際のテキストは次のようになる。

(#/paragraph)
(kono) (you/様) (ni) (sur) (eba) (zyouki/上記) (no) (fairu) (no) (yom/読 m) (i) (ni) (hituyou) (na) (ziku) (syori) (ga) (fuyou) (ni) (nar) (u) (.) (kaigyou/^J/S)

以下に属性の凡例を示す。なお，復元処理を指向しているため，学校文法には必ずしも対応していない。このことに関して次の章で述べる。

属性：

H：接頭語	E：英字	T：接尾語	O：古語
n：名詞	S：記号	g：連体詞	@：感動詞
s：名詞サ変	N：数詞	u：副詞	V：助述表現要素
v：動詞	r：カタカナ語	f：特殊活用動詞	C：制御情報
j：形容詞	Z：助詞	o：一字漢字語	
a：形容動詞	z：助動詞	J：接続詞	

4.7　形態素の取り扱い

以前のシステム[2]では形態素の定義はユーザが行ない，ローマ字ひらがな変換を行ないながら，漢字の入力を助けるものであった。それゆえ，個人辞書の中を見ると，字を引くだけの読みや，活用語尾を含むもの，最悪の場合などは，読みは記号と化して，漢字さえ引ければ何でもよいな

ど，辞書としてのスペース効率や，それ以上の自然言語処理に使用可能な，情報の取得が望めないという問題があった。

そこで特に活用語に着目[5]し，以下の方針で形態素を定義した[3]。

(1)活用語の無駄な登録はしない。
(2)ローマ字あるいは音声による入力を想定する。
(3)意味処理等のより上位の自然言語処理を指向する。
(4)その他は，学校文法を基準とする。

ここに(1)は主に動詞の語幹で登録することを意味し，動詞の変化部は助述表現要素として別の語として取り扱う。助動詞も活用するのでこれに準ずる。形容詞や形容動詞も同様である。ローマ字単位で取り扱うことにより，動詞の活用行の問題が非常に楽に処理できる。

そこでまず，学校文法との比較をまず以下に示す。

	学校文法	ローマ字	本方式
未然	書か - ない	kaka - nai	kak - a - nak - -ki (*)
	書こ - う	kako - u	kak - ou
連用	書き - ます	kaki - masu	kak - imas - u
終止	書く - 。	kaku - .	kak - u
連体	書く - とき	kaku - toki	kak - u - toki
仮定	書け - ば	keke - ba.	kak - eba
命令	書け - 。	kake - .	kak - e
その他	書かせる	kaka - seru	kak - as - er - u.
	書かす	kaka - su	kak - as - u.
	書かれる	kaka - reru	kak - ar - er - u.
	書ける	kake - ru	kak - er - u.
	書いた	kai - ta	kak - -kit a. (*)
他動詞	折る	o - ru	or - u.
	飛ばす	toba - su	tob - as - u.
自動詞	折れる	ore - ru	or - er - u.
	飛ぶ	to - bu	tob - u.

(*) -ki は直前の k を除くことを意味し，音便処理を行なうもの。

なお，上図は，(kak)(er)(u) などの様に表されるが，(a)，(i)，(u)，(e)，(o)，(er)，(it)，(ou)，(ar)，(eba)，(as)，(er)，(es)などは助述表現で，ある一定の種類である。この助述表現にはある種のパターンがあるので接続頻度によっては一つの語にまとめる方針である。

5．おわりに

本稿では，我々が現在作成中の自然言語処理指向文書エディタの概要と，そのシステムについて，その内部構造と，考え方について述べた。

今後このエディタを完成して機能の充実を図り，続編として，報告したいと考えている。

参考文献

1）荒木：多段階分割復元法による日本語文書処理に関する研究：北海道大学工学部，学位論文，昭和63年
2）栃内，伊藤，荒木，鈴木，永田：研究者向き日本語ワードプロセッサＫＫＨ：北海道大学工学部研究報告，No.109，pp.119-126(1984)．
3）堤　正勝，荒木健治，宮永喜一，栃内香次：自然言語処理における実験用文書データの取得及び保存について：平成4年度電気関係学会北海道支部連合大会，北見（1992)．
4）堤　正勝，荒木健治，宮永喜一，栃内香次：多段階分割復元法を用いた連続音声認識の誤りの復元：電子情報通信学会，音声研究会資料，SP89-92（1989)．
5）久光　徹，新田義彦：漢字かな混じり文形態素解析における非サ変動詞の分割単位設定について：情報処理学会論文誌，Vol.30，No.2，pp.159-167(1991)．
6）堤　正勝：学習機能を有する木構造辞書について：北海道大学工学部，卒業論文，昭和63年