解　説

日本語テキストエディタ†

荻　野　綱　男††

1. はじめに

最近，各研究機関で日本語のデータ，特に漢字かなまじり文で書かれたデータが計算機処理されるようになってきた．その裏にはもちろんハードウェア＝日本語入出力装置の進歩がある．一方，こうして大量の日本語データが計算機処理向きに蓄積されればされるほど，入力されたデータの校正・修正の作業が重要になってくる．入力データが正確であればあるほど，出力結果の価値が高まるのは当然なのである．

日本語データの修正には，TSS で動くエディタが有効であると思われる．この視点から，すでにいくつかの日本語テキストエディタが作られているが[1]，それを実際に大量データに対して応用した例はまだ少ない．

以下では，筆者の作成した日本語テキストエディタを事例としてとりあげ，大規模なデータに実際に使ってみた経験から問題点を述べてみたい．

2. システム構成

電子技術総合研究所のディスク・ファイルには，新明解国語辞典（三省堂）が一冊全部入っている[2]．空白以外の文字数は約428万字である．このデータの校正作業のため，筆者は TSS で動く EDIT 2 という日本語テキストエディタを作成した．このエディタは，のちに分類語彙表や中学校の理科の教科書のデータの校正にも使われるようになった．

システム構成は図のとおりである．計算機 TOSBAC-5600 に専用回線で結ばれた TSS 端末からファイルにアクセスするようになっている．端末としては，ソニーテクトロニクスの蓄像管グラフィック・ディスプレイなら何でも使える．そのうち一つ（4014型）には漢字タブレットが接続されていて，漢字の入力ができるようになっている．ただし漢字タブレットなしでも EDIT 2 は充分使えるようになっている．

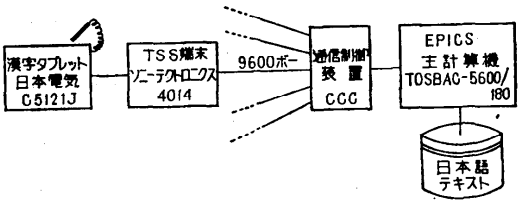

図　EDIT 2 を使うためのシステム構成

EDIT 2 の大きさは，FORTRAN のソースの行数で約2,600行であり，実行時には 35K ワードほど主記憶を必要とする．漢字の出力には，電総研ですでに開発されていたプログラムとデータを利用した．漢字の字種は約8,000字で，ドット型の文字パターンがファイルに入っている．このうち約4,000字はベクトル型の文字パターンも用意されていて，グラフィック・ディスプレイを出力媒体にした場合，ベクトル型のほうがかなり速く文字を書けるので，なるべくそちらを使うようにしてある．4014型のディスプレイを使うと，縦29cm×横38cmの画面の中に縦19行×横32字（一画面608字）表示するのが標準になり，一文字のサイズは縦10mm×横8mmになる．エディタ内ではコマンドによって文字の大きさが変えられるようになっているが，実用上充分な品質を保つためには標準の大きさの半分が最小限度になる．この場合は一画面に縦36行×横64字（2,304字）が表示される．

3. 日本語テキストエディティングの特徴

エディタの仕様の設計には，だれが（どんな人が），どのくらい（時間的・量的に），どんな環境（ハードウェア・ソフトウェア他）で使うのかを考慮して，それにふさわしいものを作っていくことが大切である．EDIT 2 の場合は，その主たるユーザが TOSBAC のテキストエディタに慣れていたこと，それに EDIT 2 がそもそも TOSBAC の TSS 上にインプリメントさ

† Japanese Text-editor by Tsunao OGINO (Department of Linguistics, Faculty of Letters, University of Tokyo).

†† 東京大学文学部言語学科

れることから，EDIT 2 のコマンド体系を TOSBAC のエディタのそれに合わせるようにした．また，EDIT 2は，はじめから大規模使用が予想されたので，少しでも使いやすくすることに心がけた．そのため，ユーザの意見をなるべく取り入れて，たとえ小さくとも次々に改良を加えるようにした．

EDIT 2 は約2年使われてきたが[3]，この間，どんなコマンドがよく用いられるかをカウントしてみた．その詳細は別の機会に報告した[4][5]が，現在約 10 万個のコマンドが入力されたところである．これを時期的に三分して集計すると，使われるコマンドがしだいに変化してきていることがわかる．行単位の挿入・削除のコマンドは第一期に多く，文字列単位の修正コマンドは第二期に多く，文字列の探索や数行にわたる印字のコマンドは第三期に多い．また，EDIT 2 の一回あたりの平均使用時間は，だんだん短くなってくる．これらの現象は，通常のテキストエディタとちがって EDIT 2 があくまでデータの校正・修正のためにだけ使われたせいである．つまり，データの校正はまず行単位の大幅変更が多く，次にそれからもれ落ちた小さな変更を行うのである．データが完全に近づくにつれて，データの検索・表示といった使用が多くなってくる．今後は，データ（特に文字列）の検索用のコマンドを充実させていくか，あるいは検索専用のプログラムを新しく作成するようになっていくだろう．

日本語テキストエディタのもっとも大きな特徴は，漢字の入力ができることであろう．逆に，漢字の入力がうまくできないと，大変使いにくいエディタになってしまう．電総研には KEDIT* という漢字エディタがあるが，今回の校正作業でこれを使わなかった理由の一つは，KEDIT での漢字の入力がわずらわしく思われたことであった．KEDIT では，漢字の入力には4ケタの十進数を使うしか方法がなく，ひらがな文字列の入力にすら労力を必要とする．さらに，一つのファイル内に漢字コードと ASCII の英数字コードとの共存を許すため，漢字コード列の前を4ケタのアポストロフィ（′′′′），後を4ケタの引用符（″″″″）で区切らなければならないのである．

エディタでは，コマンドとデータを区別しなければならない．データの入力には大別して2種類の形式が考えられる．一つは，コマンドといっしょにデータも並べて入力する方式で，データはデリミタと呼ばれる何らかの文字で両端を区切り，データの範囲を明示する．たとえば，EDIT 2 のコマンドでは，文字列の置換を行うコマンド

-RS:/XYZ/:/STU/

は，文字列 “XYZ” を文字列 “STU” に置き換えることを表わす．こういう形式をデータ埋込型と呼ぶことにする．

もう一つの形式は，コマンドを入れるとエディタがユーザにデータを入れるようにうながし，それからデータが入力できる方式で，コマンドとデータを別々に入力するものである．たとえば，EDIT 2 のコマンドでは，先ほどのコマンドと同じことを指定するのに，

-RS:/XYZ/

とだけ入れると，エディタは

enter

*

と打ってくるので，“*” の次に “STU” と入れるのである．こういう形式をデータ独立型と呼ぶことにする．

たいていのエディタに，データ埋込型とデータ独立型の両形式のコマンドがあるが，頭の中で考えたことを直接コマンドとして表現するためには，データ埋込型が使いやすい．筆者の使用経験では，比較的長いデータを入れるときに独立型，短いデータには埋込型を多く使うようである．

図のようなシステム構成で使うエディタは，どんなコマンド形式にしたら使いやすいだろうか．次の理由により，グラフィックディスプレイ端末についている英数字キーボードをおもに使うようにしたほうがよい．

（1） 操作性は，漢字タブレットより英数字キーボードのほうがはるかによい．特に，TSS に親しんでいる人はそう感じる．

（2） 英数字キーボードだけでも，英字の大小文字が使えるし，かなりの特殊記号があるので，これらはそのまま入力して，エディタの内部で 16 ビットの漢字コードに変換すればよい．

（3） 漢字データといっても，字数から見れば漢字とかなの比率は同じくらいで，ひらがな・カタカナは英数字キーボードからでもローマ字で入力できる．

（4） 漢字の入力は，漢字コードで直接入力してやればできないわけではない．ただし，この点は漢字タブレットの利用を考えるべきである．

英数字キーボードから多くの「字種」を入れるため，

* 電総研内で東芝作成の手書きのマニュアルのゼロックスコピーが手に入る．

シフト文字を使わなければならないが，何とかできるようである．

ここで EDIT 2 のデータ埋込型の文字列の表現のしかたについて規則をまとめて述べておこう．

- 文字列の両端は，文字列中に含まれない“\”以外の任意の一字でくくり，その直前に“:”を置く．
- 文字列は1個以上の副文字列の並びから成る．副文字列の間は“\”で区切る．副文字列は，その先頭の1字を見て，ひらがな，カタカナ，ASCII，漢字コードのうちのいずれかに判定される．
- ひらがな副文字列は，Hまたはhで始まり，そのうしろにローマ字つづりが続くものである．ローマ字のつづり方は，ヘボン式，訓令式，新日本式のいずれでも，あるいはそれらのまざったものでもよい．
- カタカナ副文字列は，Kまたはkで始まることを除いてひらがな副文字列と同じである．
- ASCII 副文字列は，Aまたはaで始まり，そのうしろに端末が持っている“\”以外の文字が続くものである．
- 漢字コード副文字列は，十進数字列である．4ケタより長い数字列は4ケタごとに一つの漢字コードを表わすものと見なす．4ケタより短いコードを使用するときはカンマか“\”で区切る．
- H, h, K, k, A, a, 数字以外で始まる副文字列は，EDIT 2 によってその前にAがあると見なされる．
- 漢字コード列の中に数字以外で出てきたり，ひらがなやカタカナ列の中にローマ字として使われない英数字記号が出てきたとき，EDIT 2 によってその前に“\”があると見なされる．

これらの規則にしたがってデータ埋込型のコマンドの例は次のようである．

-RS:/KSA 239/:/HSA 197/
(“サ・”を“さ”に置き換える)
-IS#9:/*␣\HATCHI␣*/
(9カラムの次に“*␣あっち␣*”を挿入する)

さて，漢字タブレットを使用するときは，データ埋込型のコマンドはむずかしい問題を含んでいる．つまり，漢字タブレットで漢字1字を入力すると，プログラムで受け取るときには英数字キーボードから適当な2文字を入力したことと同じなので，その中にデリミタに一致してしまうものがあるかもしれないのである．これを解決するのは容易でないので，現在，EDIT 2 は独立型のデータ入力の場合だけ，漢字タブレットの使用が可能になっている．

4. 日本語テキストエディタの問題点

EDIT 2 を使った経験から，日本語テキストエディタというもののもつ問題点をいくつか述べたい．

まず，何といっても漢字の入力，およびそれから派生するさまざまの問題がある．エディタを使うような人は，漢字入力に習熟した専門オペレータではない．専門家はむしろ第1次入力を担当するものである．だとすれば，当然漢字入力に不慣れな人がエディタを使うことになる．それに，汎用の機器で構成する限り，端末としてはグラフィックディスプレイを使うしかなく，せいぜい漢字タブレットを付加する程度である．そうなるとコマンドとデータをどう入れればいいのかが大きな問題になる．それに対する一つの解答は，コマンドもデータも基本的に英数字キーボードから入れるようにすることであり，上に述べたように EDIT 2 もこの方向に沿っている．こうすれば，英数字キーボードに慣れた人ならまあまあ使えるようになる．

エディタでは文字列$_1$を文字列$_2$に置き換えるコマンドというのが必須である．ところが，日本語テキストエディタでこういうコマンドを用意すると，文字列$_1$をそのまま指定することは，それに相当する漢字を入力することと同じ手間がかかるのである．もちろん，文字列$_1$が英数字やひらがな・カタカナなら入力が簡単だし，頻出する特殊記号ならそのコードを覚えるようになるのでよいが，8,000 種もある漢字のコードはとうてい覚えていられない．そこで，文字列$_1$の代わりに別の方法で指定することを考えなければならない．EDIT 2 ではカラム番号を指定し，カラム$_1$〔からカラム$_2$まで〕を文字列$_2$に置き換えるというコマンドを用意した．ライトペンがあればそれで指定することもよかろうが，その場合，コマンド形式に充分注意を払わないと，かえって使いにくくなる恐れもある．漢字の入力の困難さによって，置き換える対象を指定するのにも一工夫しなければならないのである．

普通のエディタでもそうだが，日本語テキストエディタではデータ破壊のダメージが一層大きい．つまり破壊されたデータを回復するのに，漢字を入力しなければならないからである．データ破壊の最大の原因はコマンドの誤入力だが，エディタのユーザが人間である以上，誤入力は避けられないものである．また，エディタのコマンドには，プログラムのテストランに当たるものがなく，いつでも即実行される性格をもっている．こうして，うっかり数文字消してしまうような

小さな事故はしょっちゅう起こるのである.

これには,バックアップ用コピーを作っておくなどの対策も必要であるが,EDIT 2 ではさらに二つの対策が用意された.一つは,変更後のデータを行単位で変更前に戻すコマンドを用意したことである.一行に数回の変更が加えられることもあるので,何回前のデータに戻すか指定できるようにしてある.また,一度ログオフしたあとで誤修正に気づいた場合も,ログオンして回復できる.この機能は,EDIT 2 が,行単位にきちんと分けられたすでに存在するファイルの修正用に開発されたから可能なのであり*,一般のエディタでの実現は少々むずかしいだろう.しかし使ってみるとこれは実に強力であった.

もう一つの対策は,コマンドの“テストラン”を可能にしたことである.すなわち,TEST コマンドを入れると,次に NOTEST コマンドを入れるまで,端末上の印字はあたかもコマンドを実行しているかのようによそおい,実はファイルには何の変更も加えていないのである.特に,はじめて EDIT 2 を使う人は,慣れるまでこうして練習できる.また,特定のユーザ名以外のユーザが EDIT 2 を使うと,テストモードにしかならず,ファイルをこわされる心配なしにデータの検索だけを許すようになっている.

データの破壊はエディタの虫によっても起こり得る.エディタの虫は早く確実に取り除かなければならない.特に重要なことは,エディタの虫とユーザの誤りとをはっきり区別することである.このため,EDIT 2 では,入力されたコマンドのイメージを保存しておき,必要なときに HISTORY コマンドを入れると最近のコマンド100個が見られるようになっている.これは EDIT 2 の虫取りとユーザの教育とに大いに役立った.

漢字入力に関連するこれらの問題のほかに,漢字表示速度が遅いことが問題になる.電総研では 9,600 ボーの専用回線で端末を結んでいるが,これでも,標準の半分の文字サイズで一画面(36行×64字ただし約半分は空白)を表示するのに1分以上かかり,カラム番号も表示するようにすると2分半もかかる.毎秒10～20字くらいの表示速度というのは,蓄像管グラフィックディスプレイという汎用のハードウェアの限界かもしれないが,これでは修正の能率があまりよくない.漢字パターンをセンター側で生成することなく,漢字コードで伝送して端末側で漢字を表示すれば,大幅な速度向上が期待できる.しかしこうなるといわゆる漢字ディスプレイ端末ということになってくるので,エディタの設計方法も根本的に変える必要があるだろう.ともあれコマンドの集計結果から見ると,EDIT 2 のユーザはいろいろ工夫してデータの表示量を少なくしようとしているようである.

一般のエディタについても,どういうのが使いやすいエディタなのか,まだ話題になることが少ない[6].しかし,人間と計算機を結ぶともいえるエディタにおいては,「使いやすさ」こそがもっとも重要なのである.日本語テキストエディタでは,漢字入力という困難な問題がからむだけ,もっと「使いやすさ」を追求することが望まれる.使いにくいエディタでは,長時間使いこなすことはできない.現在,専用の漢字処理システムなども作られるようになってきたが,まず,多くの人が自分なりの漢字処理システムを作り,実際の使用経験を積み重ね,それに基づいて一歩一歩改良していくことが必要なのではないだろうか.そのために,より多く「日本語テキストエディタ」が作られ,使われるようになってほしいものである.

* 編集対象となる日本語データファイルは数個に限定されているため,それぞれに専用のデータ保存用永久ファイルを別に用意し,修正が行われてポインタが移動するたびに,その旧レコード1行分をそこに格納するようにしている.

参 考 文 献

1) 人間機械系としてのプログラム作成環境, p. 185, ソフトウェア産業振興協会, 東京 (1979).
2) 横山晶一: 国語辞典データベース化の準備, 電総研彙報, Vol. 41, No. 11, pp. 855-863 (1977).
3) 荻野孝野: 大量漢字データの TSS による校正, 日本語情報処理シンポジウム報告集, pp. 99-104 (1978).
4) 荻野綱男: 漢字エディタの作成と使用, 日本語情報処理シンポジウム報告集, pp. 90-98 (1978).
5) 荻野綱男: TSS 漢字エディタの設計, 情報処理学会計算言語学研究会, 16-1 (1978. 11. 17).
6) 荻野綱男: TSS エディタの使い勝手, bit, Vol. 11, No. 1-3, 4-8 (1979).

(昭和54年6月15日受付)