TM-1118

Symmetry上の電子英和・和英辞書について

平田 圭二

October, 1991

© 1991, ICOT

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Symmetry 上の電子英和・和英辞書について
Ver. 1.0

平田 圭二
(財) 新世代コンピュータ技術開発機構
第 1 研究室
hirata@icot.or.jp

1991 年 9 月 30 日

現在, icot22, 21 (Symmetry) の上では, 電子英和・和英辞書をオンラインで利用することができます[1]. UNIX shell のレベルからでも nemacs の中からでも英和・和英辞書が引けます. この辞書データの中身は, 三省堂の「ニューセンチュリー英和」と「新クラウン和英」と同じもので、富士通 FM Towns の CD-ROM, OASYS, Sony の Datarnan にも同じ物が乗っています[2].

1 セットアップ及び使い方

1.1 nemacs からの使い方

まずセットアップの方法を説明します. まず icot21 か icot22 のユーザを想定しています.

1. .cshrc に
   set path=($path /usr/edict/bin)
   という行を加えます. nemacs からしか使わない人はこのステップを省いても OK です. ここで /usr/edict/bin というのは辞書を引くコマンド eluw, jluw [3]を置くディレクトリです.

2. .emacs に

```
(global-set-key "\e\C-L" 'look-up-eword)
(global-set-key "\e\C-D" 'dict-spell-word)
(global-set-key "\e\C-J" 'look-up-jword)
(setq exec-path (append exec-path (cons "/usr/edict/bin" nil)))
(autoload 'look-up-jword "~/nemacs/edict" nil t)
(autoload 'dict-spell-word "~/nemacs/edict" nil t)
(autoload 'look-up-eword "~/nemacs/edict" nil t)
```

[1]ICOT 内での英和辞書のリリースは 1990 年 11 月 9 日, 和英辞書のリリースは 1991 年 3 月 13 日でした.

[2]今なら CD-ROM ドライブと辞書の CD を買ってきて, junet に流れていた PDS のドライバソフトでアクセスする方がお手軽かも知れません.

[3]eluw というコマンド名は English look-up-word の気持ちで, jluw は Japanese look-up-word の気持ちです.

という行を入れます. ここで ~/nemacs というのは, nemacs のための辞書フロントエンドプログラム edict.elc の置いてあるディレクトリです.

すると次の 3 つのキーマップが使えるようになります.

ESC C-l : nemacs 中で, 意味を知りたい英単語の上にカーソルを持って行き[4], ESC C-l と打つと, window が split してその単語の意味を表示します.

ESC C-d : 意味を知りたい英単語の上にカーソルを移動してこのコマンドを実行すると, 語尾の ing や ed や s 等の変化を自動的に取り除いてからその英単語を引きます (ispell のフロントエンドを流用しています).

ESC C-j : 日本語の英訳を知りたい時は, region を指定して本コマンドを実行します. window が split してその単語の意味を表示します.

また英和, 和英とも nemacs の echo area (スクリーンの最下段) からの入力が可能です. 英和を呼びたい場合は,

```
C-u ESC C-l あるいは M-0 M-x look-up-eword
```

と打ちます. すると echo area に word? というプロンプトと, region で指定されている単語がデフォールトで表示されるようになっています. 和英の場合も同様で,

```
C-u ESC C-j あるいは M-0 M-x look-up-jword
```

と打って下さい.

1.2 UNIX シェルからのコマンド起動

UNIX シェルレベルから直接辞書を引くコマンド eluw, jluw を起動することもできます (実際 nemacs の中からこれらのコマンドを呼んでいます). 英和辞書を引きたい場合は,

```
eluw 大文字の英単語 PRTX m | nkf
```

と打ちます. 第 1 引数が引きたい英単語で, 全部大文字で打ち込んで下さい. 第 2 引数はおまじないだと思って PRTX, 第 3 引数も同じくオマジナイで m と打って下さい. nkf は eluw の出力が EUC なのでコード変換のために必要です.

シェルからのコマンド起動で和英辞書を引く場合, 次の 2 通りの呼び方ができます.

```
jluw 日本語の単語 TX m | nkf
jluw TX m < file | nkf
```

(この時, 日本語の単語 の漢字コードは, (漢字 in/out 付き) JIS か EUC です). 上の呼び方は基本的には eluw と同様ですが, シェルからの起動では特殊キャラクタが邪魔をして, 希望するキャラクタシーケンスを jluw に渡してくれないことがあり, たいがいはエラーになってしまいます. そこで, jluw は標準入力を受け付けられるようにもなっています. 例えば, 予め file の中に引きたい日本語の単語を書いておき, 下の例のように起動した方が安全です (JIS でも EUC でも受け付けます).

eluw, jluw は当然 remote shell でも使えます. nemacs の中から remote shell で使用する場合は, UCB 系と System V 系の両方のルーチンを用意していますので, edict.el ファイルの 80 ~ 90 行目当たりを適宜書き換えてコンパイルし直して下さい. nemacs をインタフェースに使う限り, remote shell で使用してもレスポンスを除けば操作はまったく同じです.

[4] 単語の上ならどこでも OK です.

1.3 その他の機能

語義だけの出力

eluw に関して, 前節では第 2 引数はおまじないだと思って PRTX と書いて下さいと述べましたが, ここを KJ とすると, 語義 (意味のキーワード) だけ引くことができます. 例えば,

```
eluw LONELINESS KJ m | nkf
```

と打つと,

```
☆ loneliness 孤独, 寂しさ, 心細さ
```

のような出力が得られます[5]. 次のような alias が便利かも知れません.

```
alias gogi '/usr/edict/bin/eluw `echo \!* | tr a-z A-Z` KJ m | nkf'
```

読み方の分からない漢字

日本語の単語で読み方の分からない漢字が出てきたりした場合, 和英辞書で読み方を引いてみることができます. 例えば, 「杜鵑」を引いてみると,

```
☆ ほととぎす【杜鵑】 <<鳥>>a cuckoo <<pl. -s>>.
```

のように読み方だけでなく英訳を知ることもできます.

2 使い方のコツ

eluw, jluw には受け付ける単語のフォーマットがあります. 正しいフォーマットの単語が与えられたら, まずキーワードファイルを検索し, 該当する単語があるならその意味を表示します. eluw, jluw は, 人間が辞書を引く時のように, 知りたい単語が見つからなかった時にその前後を探したり, 別の探し方に切り換えたりと言った intelligent なことは一切していません[6]. そこで, 探したかった単語が見つからなかった時のワンポイントレッスンをします. はっきり言って引き方にはちょっとしたコツが必要です (特に jluw は).

2.1 eluw

- ハイフンを含んだ単語は, カーソルの乗っている単語の項目として調べられます. 例えば data processing の意味を知りたい場合, data で引いても processing で引いても出てきますが, data で引いた場合は, data の項目の一つとして出力されます.
- 熟語はキーとなる重要な単語の項目として登録されています. 人間が普通に辞書を引く場合と同様に, まず重要そうな単語を引いて, それから熟語を探して下さい.
- 綴りが怪しい時には, ESC C-d を使って ispell のフロントエンドを利用するのも一つの手です.

2.2 jluw

駄目だと思っても一応引いてみる

和英辞書のインデックスファイル[7]の中の項目が, 引ける単語一覧表になっているわけですが, 「げげっ, こんな項目もあるの?!」というような単語が時折見受けられます. 例えば「ひくひくする / ひくひく / ヒクヒク」, 「ビキニ姿の女」「屠所に引かれる羊のように」「親

[5]何故 PRTX や KJ なのかは第 7.1 節を参照して下さい.

[6]後でちょっと触れますが, 平仮名で見つからなかった場合は, 片仮名で検索してみる位のことはしています.

[7]第 6 章インストールを参照. GO ファイル中の data_dir 変数で指定されるディレクトリ中の jIDX ファイルのこと.

知らずが生える」「生兵法は大けがのもと」「先客はみんな帰ったあとだった」等です. とにかく引いてみると良いと思います.

漢字で駄目なら平仮名か片仮名で試してみる

和英辞書のインデックスファイルの見出しは 53043 項目もありますが, 実際は同じ言葉が平仮名, 片仮名, 漢字で登録してあったりします. 例えば,

あい / アイ / 愛

肺門りんぱせん炎 / 肺門りんぱ腺炎 / 肺門リンパせん炎 / 肺門リンパ腺炎 / 肺門淋巴腺炎

などで, どのパターンでも検索できます. 一方, 平仮名でしか登録されていない言葉もありますので, 漢字で駄目な場合は, 一度平仮名か片仮名に直してその読みで引いてみることをお勧めします. 但しその場合は同音異義語が出力される可能性もありますが. また検索プログラム中では, 平仮名で検索して失敗した場合は, 自動的に片仮名に変換して再度検索するということをしています.

適当に語尾を付けてみる

例えば「ふしだら」という単語は「ふしだら」でも「ふしだらな」でも引くことができます. ところが, 「総舐めにする」は「総舐め」では引くことができません. ただ「総舐め」で引くのを失敗しても「にする」という語尾を付けることは (日本人なら恐らく) 容易にできるでしょう. 同じような言葉に「平身低頭する」というのがあります (「平身低頭」では引けません).

このように, 付加する語尾が分かり易い場合はまだ良いのですが, 例えば「下剋上」という見出しは無いのに「下剋上である」という見出しはあります. この時「である」という語尾を付ける / 見つけるのはちょっと難しいかも知れません. 同様の例として, 「踏んだり蹴ったり」というのは引けませんが, 「踏んだり蹴ったりだ / ふんだりけったり[8]」というのは引けます. ここら辺については systematic な規則がないので, その場その場で, いろいろな語尾を試してみるしかないと思われます[9].

ハイフンを使ってみる

「ナニナニごっこ」「ホニャララもどき」「反ベケベケ」「暴れナニナニ」という表現を知りたいときには, ナニナニやベケベケのところをハイフンにして, 「- ごっこ」「反 -」という形にすれば和英辞書を引くことができます. ハイフンは ASCII のマイナス記号 (0x2d) に限ります. またハイフンに続く仮名や漢字の間に空白を入れてはいけません.

数字やアルファベットで始まる言葉も引ける

項目数は少ないですが数字やアルファベットを含む表現を引くこともできます. 例えば「1つ」「2次会」「3DK のアパート」「FM 放送」「X 線」等です. ここで, 数字やアルファベットは ASCII キャラクに限ります. また, 数字やアルファベットに続く仮名や漢字の間に空白を入れてはいけません.

2.3 多少のカストマイズ

`edict.el` (nemacs のフロントエンドのプログラムが入っているファイル) 中の変数 `dict-buf-erase-mode` が t (デフォールト値) の場合は, 辞書を引くたびに `*Dictionary*` バッ

[8] 平仮名の時は「だ」がありません.

[9] 最終兵器は, 辞書データのインデックスファイルをエディタなどで直接検索してしまうことです. この手の ispell 風のツールを nemacs のフロントエンドに組み込みたいと言う要望はあるのですが.

ファがクリアされてから, 単語の意味が表示されます. nil の場合, 単語の意味は append モードで表示されていきますので, 以前に引いた単語の意味は *Dictionary* バッファに残っています[10].

3 三省堂との契約について

三省堂は辞書データをある料金で研究開発用としてユーザ (例えば ICOT) に「貸し出し」ます. というのも, 辞書の著作権を持つ三省堂は, (当然ですが) 辞書の著作権・所有権をユーザに売らないという方針を取っているからです.

この辞書を購入する際, ICOT と三省堂の間では,

辞書データを研究対象には使わない, ICOT 以外には絶対出さない

という契約を結んでいます. つまり, 辞書データ自身を研究対象にすることはできませんが, 辞書データを検索して研究支援のために使うのは構わないということです. 具体的に言えば, 従来の「辞書をひく」という手作業は, 辞書データを計算機上に置くことでオンライン化できますが, 辞書データはそのためのみに使用できるということです.

辞書データの取り扱いには十分注意して下さい.

ICOT のユーザの皆さんへ : 成果物 (文書, ソフトウェア, ハードウェア) と辞書データそのものが関連することのないように願います.

4 Known Bugs

4.1 JIS コードにない特殊記号, 特殊文字

元々の辞書データには, 英和・和英辞書用の特殊な記号や滅多に使われない漢字が含まれています. それら特殊記号や漢字は, 大日本印刷にて ICOT 向けにできるだけ変換して頂きましたが, 一部どうしても JIS コードに対応しきれない (存在しない) 記号や文字が残りました. それらは外字として JIS コードが割り当てられていない領域[11]に割り当てられ ICOT 向けの辞書データの中に存在しています.

本辞書ソフトではそのような特殊記号が出てきた時, 似たような記号で代用したり, 普通の記号の組み合わせで代用したりしています. 例えば, i というアクセントの付いた特殊文字は 0xa8b5 (EUC) に割り振ってあり, nemacs のウィンドウには i~ という風に表示されます. また, cut という単語の発音記号は [cʌt] ですが実際は [c ∧ t] (∧は 0xa2ca (EUC)) のように表示されます. アクセント関係では, ó は o', ò は `o という風に表示されます.

JIS コードの空白領域に割り当てられた第一水準, 第二水準に無い漢字についても, 似たような漢字や省略形の漢字を当てて代用しています. ところがやはり対応し切れない漢字がいくつか残ってしまいました. 例えば「もがく」という単語です. 無理に書けば「足宛」という雰囲気なのですが, 出現頻度が低いのでそこまで頑張らなくてもということで, 一律に「〻」という記号で置き換えてしまいました. 従って「もがく」を引いた場合,

☆ もがく【〻く】 struggle; wriggle; writhe; [あせる] be impatient <<for>>.

が出力されます. このような漢字が全部で 28 文字あります. 第 1.3 節に読みが分からない時に和英辞書が使えることがあると書きましたが, これら 28 文字については当てはまりません.

---

[10] 老婆心ながら, デフォールト値を変更したら edict.el を再コンパイルして下さい.

[11] 実際には 0xa8a1 ~ 0xaaa1 (EUC). 一部 JIS コードの領域を潰しています.

4.2 ispell フロントエンド

ispell フロントエンドが変に賢いことが却って災いすることがあります. 例えば honest という単語を ESC C-d で引くと ispell が起動され, honest を hone の最上級だと思って hone を引きに行きます. 出力は,

```
☆ hone [houn]
<名> (特にかみそり用の) 砥石 (といし).
--<動> を砥石でとぐ; [技術など] を磨く.
```

のようになります. もちろん honest を ESC C-l で引いた場合には意図した通りに動いてくれます.

ESC C-d で起動された ispell プロセスは, 後の M-x ispell-buffer 等の実行で起動されるべき ispell と兼用にしたい所です. 現在は中身がほとんど同じであるにもかかわらず別々のプロセスとして走り, 計算資源を多少無駄使いしています.

4.3 辞書データ自身に含まれるバグ

辞書データ自身がバグっているために, 検索結果がおかしくなってしまうことがあります.
例えば arouse という単語を引くと以下のような出力が得られます.

```
☆ *arouse ə ra'uz]
<動> (a・rous・es[~ə z] ....
```

これは発音記号を表す左鍵カッコが欠けているように見えます. しかし, 辞書データ中の見出し語の入っているレコードのデータ内容を見ると, 見出し語と発音記号が arouse ə ra'uz] のようにくっついて入っており, 代わりに発音記号のレコードが存在していません. この影響で, 検索のためのキーを保持しているレコードやシラブル情報のレコードも間違ったデータを保持しています.

arouse 以外にも同様のバグが存在していると思われます. 発見されたかはお手数でも平田までお知らせ下さい.

5 辞書の機能

大日本印刷から提供される辞書データを使って実現できる (等の) 検索機能一覧を示します. ここで ○ が現在リリースしているサービスで, ● がまだリリースされていないサービスです (リリース予定であるとは違います).

○ **見出し語検索** (英和・和英)

● **見出し語部分検索** (英和・和英) 見出し語の一部に対する前方・後方一致検索.

● **表記形検索** (和英のみ) 漢字 + かなに対する前方・後方一致.

● **逆引き検索** 指定された和訳語をもつ英和見出し, 及びその逆.

● **条件検索** (英和・和英) 指定された語 (一語以上) を含む用例, 例文の検索.

author の時間不足のため, 取り敢えず一番需要が多いと思われる英和・和英の見出し語検索のサービスのみリリースしております. 今後, いろいろな検索をサポートしていく予定です.

何か不明の点, 御意見, 御要望, 辞書データの取り扱いに関する質問, バグ情報等々何でも平田 (hirata@icot.or.jp) までご連絡下さい。

6 インストール

6.1 辞書データファイルを MT からダンプして生成する場合

三省堂の英和・和英辞書を使いたい方は，まず三省堂[12]と契約して下さい．契約はサイト毎です．インストールされるデータやプログラムは，MT や CMT 等の実費を払って大日本印刷[13]から受けとって下さい．

1. 辞書加工プログラム，辞書検索プログラム，インストール用コマンドファイルをごっそりどこかにコピーしてください．

2. 英和・和英辞書の生データも (デフォールトで) DNP-EJ, DNP-JE という名前でどこかのディレクトリにダンプして置いて下さい．GO の中の MT_dir をそのディレクトリに書き換えて下さい．

3. 生成される辞書データファイル (全部で約 18MB; EJ, eIDX, eKEY, JE, jIDX, jKEY の 6 つのファイル) を置くディレクトリを決めたら data_dir をそのディレクトリに書き換えて下さい．

4. 辞書を引くコマンド eluw, jluw を置くディレクトリを決めたら bin_dir をそのディレクトリに書き換えて下さい．辞書フロントエンドプログラム edict.elc を置くディレクトリを決めたら nemacs_lib_dir をそのディレクトリに書き換えて下さい．ここでの設定値とセットアップの所の設定値 (第 1.1 節) が食い違わないようにして下さい．

5. GO と打って下さい．辞書データファイルを生成するのは大変時間がかかるので，すでに生成されている辞書データファイルを remote mount して使うかどうか聞いてきます．また，ディスクのワークスペースとして (DNP EJ と DNP-JE も含めて) 155 MB 必要なので，それだけ余裕があるかどうか聞いてきます．適当に正しく答えて下さい．

6. ひたすら待って下さい[14]．

6.2 すでにどこかにある辞書データファイルを remote mount して使う場合

1. 第 6.1 節と同様，必要なファイルをごっそりどこかにコピーしてください．

2. 辞書データファイル (EJ, eIDX, eKEY, JE, jIDX, jKEY) のあるディレクトリをとにかく remote mount して，そのディレクトリを data_dir に書き換えて下さい．

3. 辞書を引くコマンド eluw, jluw を置くディレクトリを決めたら bin_dir をそのディレクトリに書き換えて下さい．辞書フロントエンドプログラム edict.elc を置くディレクトリを決めたら nemacs_lib_dir をそのディレクトリに書き換えて下さい．ここでの設定値とセットアップの所の設定値 (第 1.1 節) が食い違わないようにして下さい．

4. GO と打って下さい．辞書データファイルを remote mount して使うかどうか聞いてきますので，y と正しく答えて下さい．うまく行けばあっと言う間にインストール終了です．

---

[12]〒 101 千代田区三崎町 2-22-14 (株) 三省堂出版局 電子出版部 山田 薫 様 (電話 03-3230-9470) が窓口です．

[13]〒 162 新宿区市ヶ谷加賀町 1-1-1 (株) 大日本印刷 CTS 事業部 開発 2 部 4 課 高田 和彦 様 (電話 03-3266-4587) が担当です．

[14]ICOT の Symmetry で英和・和英をインストールした時の様子が Script ファイルに残っていますが，全部で約 1 時間かかっています．

7 実装

本章では,辞書データのデータフォーマットや検索アルゴリズム等について述べます. また実際に,三省堂辞書データの加工ソフト及び辞書検索ソフトのソースコードを付録として本 technical memorandum に添付します.

7.1 辞書の生データのフォーマット

ブロックフォーマット, レコードフォーマット, 識別子: 大日本印刷から送られてくる辞書の生データは, ブロック構成をしています. 1 つのブロックは任意個のレコードを含み,最大ブロック長は 2008 バイトです. 1 つのレコードは 6 つのフィールドから成ります. 各フィールドには,全見出し語の通し番号や, 1 つの項目を構成する複数個のレコードの通し番号,テキストデータ等が入ります. ある英単語の意味を調べると, その内容は例えば発音記号, 複数形, 1 番目の語義, 2 番目の語義等といったサブ項目から成り立っています. このサブ項目毎に 1 つ以上のレコードが割り当てられています. あるレコードがどんな種類のサブ項目のデータを含んでいるのかは,識別子と呼ばれるフィールドによって知ることができます.

識別子にはどのようなものがあるのかを表 1, 表 2 に示します. これら識別子の種類から, どのような情報が辞書データとして提供されているかが分かると思います. これらの識別子をフルに用いると, 単純な英和・和英検索だけでなく様々な種類の検索が可能になることもお分かり頂けると思います (第 5 章参照).

また, データ中 (EUC) に含まれる数字, アルファベット, 記号等はすべて全角で入力されていますが,本辞書ソフトはできるだけ対応する ASCII コードに変換しています.

特殊記号, 特殊コード列: 第 4.1 節にも述べましたが,英和・和英辞書の元データには電子写植機用の特殊記号やコマンドが埋め込まれています. ICOT 向けデータ作成のため, それら特殊記号やコマンドは大日本印刷にて適当に変換されたり削除されます. 現在 ICOT 向けデータに含まれる特殊記号, コマンドは次の 2 つです.

`0xcffe:`

特殊ブランクコードであり,本辞書ソフトではすべて読み飛ばしました.

添字指定のコード列:

基本的には次の 3 パターンがあります[15].

＜行＞添字＜行＞
ル　　ル
添　　添

添字は 1 ～ 9 の数字です. ところが, 左右の 行 / ル / 添 の部分がアンバランスに 改行 / 行送 等となったり,添字の左の 3 文字 (6 バイト) だけ欠落したり, またその逆に右の 3 文字だけが欠落したりすることがあります. 本辞書ソフトでは, 端末への出力が主であることを考慮して,添字指定のコード列は一切削除しています. 従って, 本来は ABC_1 や ABC^2 と出力されるべきところが, すべて $ABC1$, $ABC2$ のように出力されます.

[15]実際に全角の “＜”, “＞” が入っていて, 全体で 13 バイト長です.

識別子	対応するデータの内容
ADD	見出し番号
-W1	見出し語
-K1	見出し語から切り出したキー (英単語大文字, 単語単位)
-GR	見出しに付いている * の数
-SY	シラブルのデータ
-PR	(変換表による) 発音記号
-PL	複数形
-KJ	語義から切り出したキー (漢字)
-TX	本文
-CG	用例 (-EX) かイディオム (-OM) の区別
-EX	用例
-K2	用例から切り出したキー (英単語大文字)
-K3	用例から切り出したキー (漢字)
-K4	用例から切り出したキー (カタカナ大文字)
-OM	イディオム
-K5	イディオムから切り出したキー (英単語大文字)
-K6	イディオムから切り出したキー (漢字)
-K7	イディオムから切り出したキー (カタカナ大文字)

表 1: 英和辞書識別子一覧表

識別子	対応するデータの内容
ADD	見出し番号
-W2	見出し語 (ヨミ【表記】)
-E1	見出し語
-EG	-TX の語義から切り出したキー (英単語大文字)
-TX	本文 (見出しを除く)
-CG	用例 (-R1) かイディオム (-R2) の区別
-R1	用例 (データ中 '-' の後)
-E2	用例から切り出したキー (英単語大文字)
-E3	用例から切り出したキー (漢字)
-E4	用例から切り出したキー (カタカナ大文字)
-R2	イディオム ('¶', '——' の後)
-E5	イディオムから切り出したキー (英単語大文字)
-E6	イディオムから切り出したキー (漢字)
-E7	イディオムから切り出したキー (カタカナ大文字)

表 2: 和英辞書識別子一覧表

7.2 辞書検索ファイル及びアルゴリズム

7.2.1 英和 (eluw)

英和辞書を引くコマンド eluw は 3 つのファイル eKEY, eIDX, EJ を順に読みだして単語の意味を検索します. 各ファイルに格納されているデータフォーマットは図 1のようになっています[16]. eKEY, eIDX は ASCII ファイルです. アルゴリズムのアウトラインは次のようになってい

[16] これらのファイルフォーマットは, 大日本印刷の辞書データフォーマットを元に平田が設計したものです.

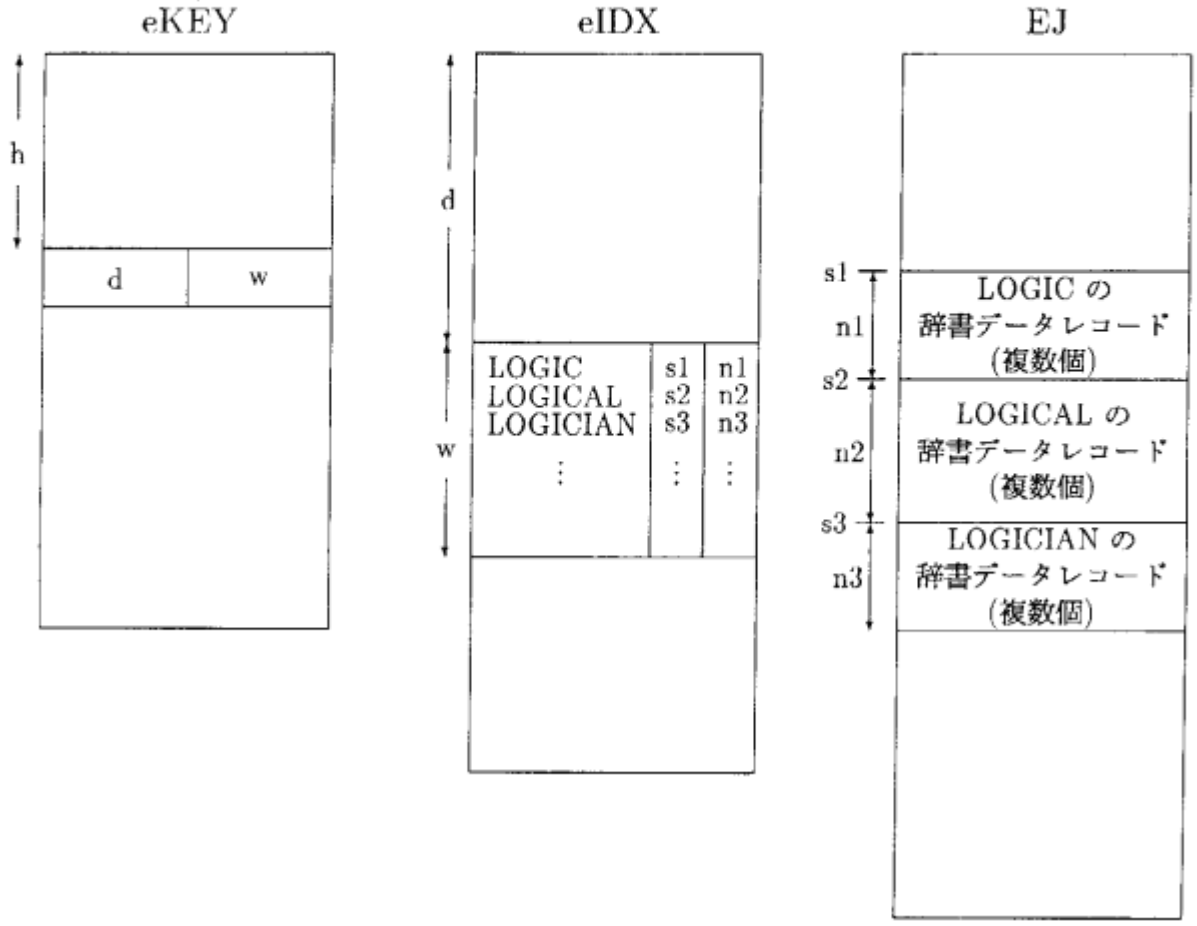

図 1: 英和辞書ファイル eKEY, eIDX, EJ のフォーマット

ます.

1. 英単語の先頭 3 文字を 3 桁の 27 進数と見なしてハッシュ値 h を計算します.
2. eKEY の先頭から h バイトの所の 1 行 (改行まで) を読みこみ d と w を得ます.
3. eIDX の先頭から d バイトの所より始まる w バイト分のデータを読み込みます.
4. その中から該当する単語をシーケンシャルに (ストリングのサーチで) 検索し, 見つかった場合 s と n を得ます.
5. EJ ファイルの先頭より s バイトの所から n バイト分のレコード (複数個) を読み込みます.
6. 読み込まれた個々のレコード中に, 調べたい単語の発音記号, 語義, 例文等が格納されているので, 適当に整形して出力します.

尚, 現バージョンの英和辞書の場合, EJ ファイルに含まれるレコードの識別子は -W1, -PR, -KJ, -TX だけです[17]. 1 つのレコードは, 表 3 のようなフォーマット構成になっています.

[17]表 1 を参照.

	識別子	データ長	データ
コード体系	バイナリ	バイナリ	EUC
バイト長	1	2	データ長で指定

表 3: EJ ファイルのレコードフォーマット

識別子	データ長	データ
W1	8	*logical [a]
PR	19	l ɑ 'd ʃ ikl/l ɔ 'd ʃ -
TX	200	< 形 > 1[説明, 結論などが] 論理的な, 合理的な. a logical argument 筋の通った議論 /The man has a logical mind. あの人は論理的な頭を持っている. [類語] 「理路整然として論法に誤りがない」 こと; → reasonable.
TX	77	2(論理上) 当然の. the logical result 当然の結果 /a logical choice 当然の選択.
TX	29	3 論理 (学) 上の. ⇔ illogical.

[a] 見出し語の先頭にある * は重要度を表しています. *** は中学学習語 (1000 語), ** は高校学習語 (3000 語), * はそれ以外で高校生に必要な語 (3000 語) の意味です.

表 4: LOGICAL に対応する EJ 中のレコード

図 1 は LOGICAL という単語を引いた例を示しています. まず先頭の 3 文字 LOG でハッシュ値 h を計算して cKEY を引いて d, w を得ます. 次に d, w で eIDX を調べると 2 行目に LOGICAL を見つけることができ, s2, n2 を得ます. 最後に s2, n2 で EJ の該当部分にある複数個のレコード (この場合 5 個) を読み込みます (表 4). さらにこれらの情報を次のようにユーザに読み易い格好に整形して出力します.

```
☆ *logical [l ɑ 'd ʃ ikl/l ɔ 'd ʃ -]
<形> 1[説明, 結論などが] 論理的な, 合理的な. a logical argument 筋の通った議論 /The man has a logical mind. あの人は論理的な頭を持っている. [類語] 「理路整然として論法に誤りがない」 こと; → reasonable.2(論理上) 当然の. the logical result 当然の結果 /a logical choice 当然の選択. 3 論理 (学) 上の. ⇔ illogical.
```

例えば man about town, sandwich man, best man, stunt man のように, ある一つの単語 (man) が他の単語と組み合わさって幾つもの見出し語を構成しているような場合は, eIDX ファイルの中に man で始まるような行が複数個現れます.

7.2.2 和英 (jluw)

和英辞書を引くコマンド jluw も eluw 同様に 3 つのファイル jKEY, jIDX, JE を順に読みだして単語の意味を検索します. 各ファイルに格納されているデータフォーマットは, eluw のそれとほぼ同様です. しかし jluw の場合は, 検索したい言葉の文字コード中に EUC (平仮名, 片仮名, 漢字) と ASCII (アルファベット, 数字, 記号) が混在しており, さらに字種がアルファベットよりはるかに多いので, 1 文字目 (漢字・平仮名・片仮名なら 2 バイト, ASCII なら 1 バイト) と 2 文字目の上位バイトの (0 相対で) 4 ～ 6 ビット目からハッシュ値 h を計算します. ハッシュ値を計

算した後のアルゴリズムは eluw のそれと全く同じです.

jluw で計算されるハッシュ値は, UNIX の sort コマンドのデフォールトの順序[18]に従っています.

7.2.3 nemacs フロントエンド

eluw, jluw を nemacs 中から呼び出すためのプログラムは cdict.el ファイルに入っています. 以下, 各関数の機能を概説します.

`look-up-eword` は, 空白やハイフンなどのデリミタを手掛りに自動的にテキスト上の英単語を切り出し, 関数 `access-dictionary` を呼び出します. echo area (スクリーンの最下段) からの入力を行うための関数 `read-from-minibuffer` を呼び出すこともできます.

`look-up-jword` は, 指定された region に含まれるストリングを関数 `access-dictionary` に渡します. echo area からの入力を行うために `read-from-minibuffer` を呼び出すこともできます.

`access-dictionary` は, 辞書の検索結果を表示するための `*Dictionary*` という名前のバッファを create して eluw 或いは jluw を呼び出します. そして eluw, jluw の出力を `*Dictionary*` バッファに書き込みます. eluw, jluw の検索出力をユーザが読み易いように整形するために, `my-fill-individual-paragraphs` と `display-dict-buf` を呼び出します.

`my-fill-individual-paragraphs` は, 右揃えや適当な改行の挿入を行います. 検索結果が複数項目になっても, 整形がうまく行われるようになっています. テキスト自体の整形はこの関数が行います.

`display-dict-buf` は, どれ位のサイズのウィンドウをどこに開いたら良いかを決定します. `my-fill-individual-paragraphs` で整えられたテキストの行数に従い, 適した大きさのウィンドウを生成したり, すでにあるウィンドウのサイズを変更したりします.

`dict-spell-word` は, emacs のライブラリ ispell のルーチンをそっくりそのまま英単語検索のフロントエンドとして流用しています. 誤った綴りの単語を検索しようとした場合は, ispell が代わりの候補を提示します. そしてユーザが選択した新たな英単語を `access-dictionary` に渡します.

[18] lexicographic by bytes in machine collating sequence (UNIX オンラインマニュアルより抜粋).

謝辞　最後にご協力を頂いた関係者方々のお名前を挙げさせて頂き，お礼の代わりとさせて頂きたいと思います．

三省堂の山田善さん，大日本印刷の高田和彦さんには，本辞書プロジェクトの最初から最後まで，契約，辞書の生データの作成等に関して大変御世話になりました．

稲村雄さん，屋代寛さん，市吉伸行さん，山本礼己さんには辞書引きソフトに関して，近山隆さんには Makefile に関して，田中裕一さんには電子辞書全般に関して，今井明さん，川合英夫さん，平野喜芳さん，富士通 SSL の中越靖行さん，西崎慎一郎さん，福島靖浩さん，三露哲司さんには和英辞書の加工プログラムに関して御世話になりました．

バグのお知らせや使用感のレポートに付いて，田中秀俊さん，岩山登さん，堀内謙二さんに御世話になりました．

注意　本辞書加工ソフト，検索ソフトに関連して生じたトラブル，損害等に付いて，ICOT は一切責任を負いませんので予めご了承下さい．

付録: 三省堂辞書データの加工ソフト及び辞書検索ソフト

これら辞書ソフトは，リスティング (印刷出力) または電子媒体 (CMT, フロッピー等) の形態にて提供され，本 technical memo に添付される．