インターフェイスの街角– POBox の拡張

増井 俊之

前号で、予測と曖昧検索にもとづくテキスト入力システム「POBox」の Palm 版と Emacs 版を紹介しました。今回は、Palm 版の機能拡張と Emacs 版の詳細について解説します。

POBox の拡張

POBox では、

1. ユーザーが文字を入力する
2. その文字をもとにシステムが候補単語を検索し表示する
3. 必要な単語をユーザーが選択する

という作業の繰返しで文章を作成します。1 や 2 の部分で別の処理もおこなえるようにすると、さらに応用範囲がひろがります。

入力手法の拡張

上記の 1 において、ユーザーが入力した文字を使う代わりに、ユーザーが選択した領域の文字列を使用する場合も考えられます。

選択した領域をもとに漢字変換をおこなえば、未確定のひらがな文字列を漢字に変換する、いわゆる〝遅延変換〟や、「沢」を「澤」に変換するような〝異体字変換〟が実現できます。後者の場合は、図 1 のような異体字辞書を使うことができます。

候補生成手法の拡張

上記の 2 において、日本語の単語を検索する代わりに英単語を検索したり、あるいは別のデータベースを検索したりすることが考えられます。

図 1　異体字辞書の例

```
沢/澤
澤/沢
```

ジャストシステムでは、このような考えにもとづき、同社のかな漢字変換システム ATOK のフロントエンドを汎用的な検索フロントエンドとして使えるようにする AMET (ATOK Multi Engine Transfer) というインターフェイスを公開しています[1]。AMET を利用すると、ATOK と同じインターフェイスを用いて辞書や CD-ROM なども検索できるようになります。

検索にかぎらず、どのような計算でも POBox から呼び出せればさらに便利でしょう。たとえば、入力または選択した文字列が数式である場合、その式を評価して計算する機能を呼び出せば、電卓のように使えます。また、特定の文字列を入力すると日付や時刻が入力されるようにすることもできます[2]。

このように、POBox などの文字入力システムは、すこし改造すればさらに汎用的な**万能テキスト変換システム**として使えるようになります。

Palm 版 POBox の拡張

以下では、ここまでに述べたようなアイデアを Palm 版 POBox に適用した例を紹介します。

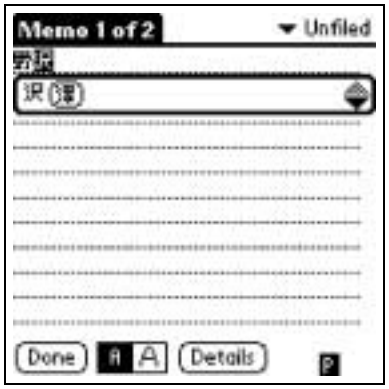
文字入力手法の拡張

Graffiti で文字を入力する代わりに既存のテキストをなぞって選択し、続けて空白文字を入力することによって選

1 http://www.justsystem.co.jp/news/98f/news/j9806301.html

2 たとえば、SKK では〝@〟で日付が入力できます。

図 2　異体字変換の例



択文字列を入力文字列と同様に扱えるように拡張すれば、図 1 のような辞書を使って異体字変換ができるようになります。図 2 の例は、「沢」の字をドラッグで選択してから空白文字を入力したところです。

候補生成手法の拡張

入力文字列が与えられたとき、単語を検索するだけでなく任意の計算ができればたいへん便利です。計算内容を一般ユーザーが指定できれば、さらに有用でしょう。

このためには、計算モジュールを POBox から呼び出す必要があります。さいわい、PalmPilot のアプリケーションには、ほかのアプリケーションからサブルーチンのように呼び出せるメカニズムが用意されています[3]。このメカニズムを用いて、任意のテキストを別のテキストに変換するための汎用のモジュール形式もひろく使われています。これを使えば、POBox を簡単に拡張できます。

拡張 POBox の使用例

図 3〜5 は POBox で計算モジュールを使用した例です。

図 3 は POBox で「1」を入力したところです。辞書には `1/壱´ `100/百´などのエントリが登録されているため、この状態で POBox は `壱´や `百´などを候補として表示します。

「123」まで入力すると、それにマッチする単語は辞書に登録されていないので、候補としては `123´だけが表示

3 この機能は、[find] ボタンによる検索などで使われています。アプリケーション独自のデータはそのアプリケーションでしか検索できないので、別のアプリケーションのデータを検索したいときは、該当するアプリケーションを検索サブルーチンのように呼び出して検索をおこないます。

図 3　「1」を入力

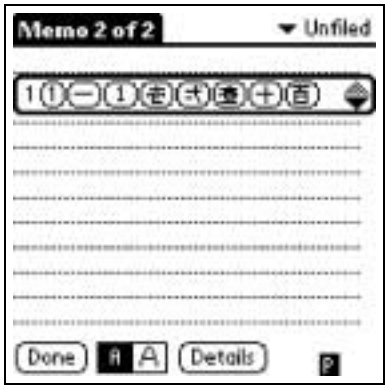


図 4　「123」を入力

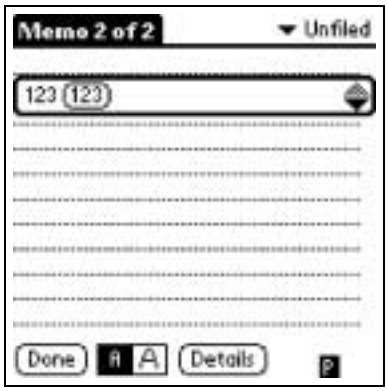
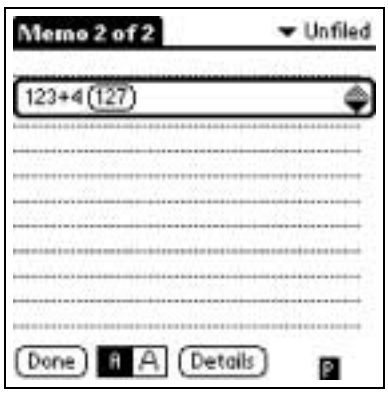


図 5　`123+4´と入力



されます（図 4）。

さらに「+4」と入力して `123+4´とすると、入力文字列全体が計算モジュールに渡され、計算結果の `127´が候補に表示されます（図 5）。

このように、計算式の入力によって自動的に値を計算し、候補として表示させることもできます。

領域をドラッグしてパターンを指定する方法と組み合わせると、さらに便利な使い方ができます。たとえば、〝pi*10〞という文字列があるとき、辞書に、

```
pi/3.14
```

と定義してあれば、〝pi〞を選択して候補となる〝3.14〞に置き換え、〝3.14*10〞という文字列に変換できます。さらにこの文字列全体を選択すると、候補として〝31.4〞が得られます。このように、普通のテキストを表計算システムのように使えます。

拡張モジュールの構造

拡張モジュールとして、福本修仁さんが作成したプログラム「Drag&Drop」[4]で使う形式のものを利用します。Drag&Drop は、Palm のテキスト枠上で領域選択した文字列をドラッグ操作で別の場所に移動したり、別の文字列に変換したりするプログラムで、決められた形式のアプリケーション・モジュールを呼び出すことができます。現在、辞書呼出しモジュール、大文字/小文字変換モジュールなど、数多くのモジュールが公開されています。

モジュールは PalmPilot 用アプリケーションとして実装され、構造の詳細は福本さんの Web ページ[5]で解説されています。

図 3〜5 の例では、計算モジュールとして SRA のほしさんが作成した「CalcPad」モジュール[6]を使用しています。CalcPad は数式の値を計算する PalmPilot のアプリケーションですが、Drag&Drop からも呼び出せます。

拡張モジュールの呼出し

POBox の辞書は、図 6 のようにパターンと単語を並べた形式です。

これを拡張し、単語の代わりにモジュール名を指定すると、候補として計算結果が表示されるようにします。「ds」と入力したとき〝PBTS〞というクリエータ ID[7]をもつ〝TimeStamp〞モジュールを呼び出したいときは、

図 6　POBox 辞書の例

```
kanji/漢字
```

下記のような辞書エントリを使います。クリエータ ID の後ろの数字は、モジュールの引数として使います。

```
ds0/{PBTS0}
ds1/{PBTS1}
ds2/{PBTS2}
ds3/{PBTS3}
```

CalcPad のクリエータ ID は〝CPAD〞なので、上記の計算を実現するには下記のエントリを辞書に記述しておきます。

```
1           /{CPAD}
2           /{CPAD}
3           /{CPAD}
.....
```

1、2、…… の後ろの空白は、いかなる入力文字にもマッチするパターンです。〝123+4〞のような入力は上の辞書の 1 行目にマッチするので、CalcPad モジュールが呼び出されて計算結果として〝127〞が返されます。

拡張モジュールのプログラム例

現在の日付や時刻を計算する TimeStamp モジュールの例を図 7 に示します。

Emacs 版 POBox

前回も書いたように、ネットワークを介していつでもどこでも個人の好みや状況に即した辞書を使えるのが将来の辞書の理想的な形態だと思います。しかし、ネットワークがそこまで進化していない現状では、個人または機械ごとに辞書を用意しておき、機械のあいだで辞書を同期させるほうが現実的です。

UNIX や Windows などでは、携帯端末と共通の辞書を利用する〝POBox サーバー〞を用意し、いろいろなクライアントから呼び出して使えるようにしておけば、すくなくともその機械では 1 つの辞書が共有できます。

前号で POBox サーバーと Emacs 版 POBox を紹介しましたが、今回は Emacs 版 POBox の使い方を詳しく説明します。

4 http://www.umap.net/MacPalm/Soft/DragDrop/index-J.html
5 http://www.umap.net/MacPalm/Soft/DragDrop/module-J.html
6 http://www.sra.co.jp/people/hoshi/palmos/calcpad.html
7 Palm では、アプリケーションを正確に区別するために、各アプリケーションにユニークな ID を割り当てます。

図 7　日付計算モジュール TimeStamp

```
#include <Pilot.h>
#include "DropModule.h" // Drag&Dropモジュール用ヘッダ

DWord PilotMain(Word cmd, Ptr cmdPBP, Word launchFlags)
{
  char buf[100];
  tDropLaunchCmdParamPtr param;

  long t;
  DateTimeType datetime;
  SystemPreferencesType sysPrefs;
  int len;

  switch(cmd){
  case dDropLaunchSupportCmd: // D&Dモジュール検索のときに使用
    return dYes_Support;
  case dDropLaunchCmd: // D&Dモジュールの呼出し
    t = TimGetSeconds();
    TimSecondsToDateTime(t,&datetime);
    param = (tDropLaunchCmdParamPtr)cmdPBP;
    switch(param->selector){
    case 0:
      StrPrintF(buf,"%02d/%02d/%02d",datetime.year,datetime.month,datetime.day);
      break;
    case 1:
      StrPrintF(buf,"%02d:%02d:%02d",datetime.hour,datetime.minute,datetime.second);
      break;
    case 2:
      PrefGetPreferences (&sysPrefs);
      DateToAscii(datetime.month,datetime.day,datetime.year,sysPrefs.dateFormat,buf);
      break;
    case 3:
      PrefGetPreferences (&sysPrefs);
      TimeToAscii(datetime.hour,datetime.minute,
        sysPrefs.timeFormat, buf);
      break;
    default:
      buf[0] = '\0';
    }
    len = StrLen(buf);
    param->resultTextLength = len;
    param->resultText = MemPtrNew(len+1);
    StrCopy(param->resultText,buf);
    param->resultBehavior = eBehavior_Replace;
    return 0;
  default:
    return 0;
  }
}
```

Emacs 版 POBox の使用法

Emacs 版 POBox の本体は Emacs Lisp のプログラム pobox.el です。Emacs のパスが通っているところに pobox.el とローマ字かな変換ライブラリ romakana.el を置き、~/.emacs に以下のようなエントリを記述します。

```
(setq pobox-toggle-key "\C-x\C-j")
(global-set-key pobox-toggle-key 'pobox-mode)
(autoload 'pobox-mode "pobox.el")
(setq pobox-server "localhost")
```

これで、Ctrl-X Ctrl-J を押すと POBox モードになります。再度 Ctrl-X Ctrl-J を押すと、もとのモードに戻ります。`pobox-server´には、POBox サーバーが動いているホスト名を指定します。

図 8　漢字の入力

(a) [k]
(こと)(ことは)(ことも)(ことを)(ことが)(から)(か)(この)(これ)(今日)

(b) [ke]
(検索)(研究)(結果)(研究所)(計算)(計算機)(けど)(研究会)(検出)(研)

(c) [計算]
(計算機)(けど)(研究会)(検出)(研)(見)(系)(結構)(見学)(検討)

(d) 計算[s]
(する)(するのは)(するように)(して)(します)(システム)(さ)(し)

(e) [k]
(計算)(こと)(ことは)(ことも)(ことを)(ことが)(から)(か)(この)(これ)

漢字入力

POBox モードで「k」と入力すると、読みが ``k'' で始まる単語が候補として表示されます (図 8-a)。

さらに「e」を入力すると、読みが ``ke'' で始まる候補単語が表示されます (図 8-b)。

候補を選択するには空白文字を入力します。図 8-b の状態で空白文字を 5 回入力すると図 8-c のようになります。

ここで改行キーを入力すると ``計算'' が確定しますが、続けて文字を入力すると前の候補の確定後に候補の検索表示が続きます。たとえば、図 8-d のように続けて「s」と入力すると、``計算'' の後ろに続く単語が予測表示されます。

このような候補選択をおこなったあとで再度「k」を入力すると、今度は ``計算'' が最初の候補として表示されます (図 8-e)。

単漢字入力

「shi」と入力してから改行文字を入力すると、以下のようにパターンが入力文字に完全に一致する単語のみが候補として表示されます。

```
[し]
(し)(シ)(shi)(市)(氏)(使)(仕)(史)(四)(子)
```

カタカナ入力/無変換

「kana」と入力すると、以下の状態になります。

```
[kana]
(かな)(かなり)(かな漢字)(仮名)(神奈川)(必ず)(必ずしも)
(仮名漢字)
```

ここで Ctrl-K を入力すると、候補の状態にかかわりなく入力パターンがカタカナに変換されて確定します。

```
カナ
```

Ctrl-K の代わりに Ctrl-G を入力すると、

```
kana
```

のように、パターンが変換されずに確定します。

記号

Emacs 版 POBox では、かなだけではなく任意の入力文字列パターンに単語を定義できます。たとえば、``.'' (ピリオド) に各種の記号を割り当てておくと、以下のように記号が候補として表示されます。

```
[.]
(。)(. )(‥)(・)(…)(゛)(゜)(○)(●)
```

パターンとして ``,,'' などの記号列を指定することも可能です (以下の例の ``”'' は HTML の閉じ引用符記号です)。

```
[,,]
(&#148;)(”)(〃)
```

\で始まる単語を辞書に登録しておくと TEX のテキストを書くときに便利です。

```
[\]
(\begin{itemize})(\verb++)(\section{})(＼)(∨)
(¥)
```

連続入力

選択中の候補を確定するには改行キーを押す必要があります。ただし、空白キーによる候補選択や改行キーによる単漢字指定がおこなわれた場合、空白または改行以外の文字を入力すると次のパターン文字とみなされ、その前に選択されていた候補は自動的に確定します。

たとえば、「jidou」と入力すると以下の候補が得られます。

```
[jidou]
(自動的に)(自動的)(自動)(自動化)(自動車)(児童)(自働)
```

続けて空白文字を入力すると、

```
[自動的に]
(自動的)(自動)(自動化)(自動車)(児童)(自働)
```

となります。ここで改行キーを押すと〝自動的に〟が確定しますが、改行キーを押す代わりに「k」を入力すると、〝自動的に〟が自動的に確定し、次の候補が表示されます。

```
自動的に[k]
        (確定)(関する)(関して)(関しては)(改行)(空白)(候補)
```

アルファベットの大文字を使うと、自動的にひらがな確定を指示することもできます。
「kono」と入力すると、

```
[kono]
(このように)(このため)(このような)(この)(このよ)(このよう)
```

と表示されます。ここで「H」を入力すると入力中のパターンがひらがなとして確定され、次の候補が表示されます。

```
この[H]
    (方式)(表示)(ひらがな)(必要)(始まる)(富豪)(はじあ)(法)(方)(比較)
```

こうすると、SKK と同じようなキー操作で連続的に入力していくことができます。

単語登録

Emacs で領域を選択してから Ctrl-O を入力すると、単語とパターンの組を登録することができます。カタカナや英数字の単語については、単漢字変換の指定後、候補から必要な単語を選択すれば自動的に登録されます。

単語の削除

辞書から単語を削除するには、その単語が選択状態にある（[ ] で囲まれている）ときに Ctrl-Q を入力します。

SKK との比較

ここまで紹介したように、Emacs 版 POBox の使い方は SKK とよく似ています。いずれも単純な検索手法を利用し、最近の SKK では補完入力も可能になったため、POBox の予測機能との差はあまりありません。

POBox とくらべると、SKK には次のような利点があります。

- ひらがな/カタカナ/全角文字の連続入力が簡単。
- 再帰的に単語登録ができる。
- 送りがなをきめ細かく制御できる。

一方、POBox は以下の点で SKK より使いやすいように思います。

- ひらがな/カタカナ/英字/全角モードの切替えが不要。
- 単語の読みを完全に指定する必要がない。
- 次単語が予測される。
- ローマ字/かなと記号の区別が不要。
- 無変換確定ができる。
- 送りがなを気にしなくてよい。

私は 10 年以上 SKK を使ってきましたが、英字/ひらがな/カタカナのモード切替えにはどうしてもなじめず、苦労しました。また、TEX で文章を書く場合などは、TEX のプリミティブを入力するために頻繁に英数字モードと漢字入力モードを切り替える必要があるのも面倒でした。辞書を十分に鍛えた POBox では、記号でも英数字でも TEX プリミティブでも、必要と思われるほとんどの単語を候補から選べるので、モード切替えの必要性はかなり低くなります。

たとえば、〝`{\TeX}プリミティブを入力〟という文字列を入力する場合、POBox では、

```
tex purimi wo nyu Return
```

と入力すればよいのに対し、SKK では、

```
l{\TeX}^JqpurimithibuqwoNyuuryoku Return
```

と入力する必要があります。POBox での予測がつねにうまくいくとはかぎりませんが、〝q〟や〝^J〟のようなモード切替え操作が不要なのは利点といえるでしょう。

辞書の同期

複数の計算機で POBox を使っていると辞書の学習が別々におこなわれてしまうため、ときどきこれらを同期させる必要があります。

POBox では、選択・確定した単語をつねに辞書の先頭に追加する学習方式を採用しています。1 台の計算機で学習させている場合はいいのですが、複数の計算機で学習させた辞書をマージするときは注意が必要です。

たとえば、初期状態の辞書が、

```
XXX
aaa
bbb
```

だったとします。このとき、ある計算機で単語 ``aaa''を使うと、辞書は次のように変化します。

```
# 辞書A
aaa
XXX
bbb
```

一方、別の計算機で単語 ``bbb''を使用すると、辞書は次のように変化します。

```
# 辞書B
bbb
XXX
aaa
```

これらの辞書をマージする場合、上または下から順番に単語を追加しようとするとうまくいきません。

たとえば辞書 B に辞書 A をマージするとき、B にない単語だけを付け加えようとしても B は変化しませんし、A にある単語をすべて先頭に追加しようとすると、B が完全に A に置き換わってしまいます。

この場合、マージされた辞書は以下のようになるべきと思われます。

```
# 辞書C
aaa
bbb
XXX
```

辞書をこのようにするにはどうすればよいでしょうか。``aaa''は辞書 A で、``bbb''は辞書 B で学習された単語であることがはっきりしている場合は、それらを別にしてマージすればよいでしょう。しかし、学習した単語をすべて憶えておくのは大変なので、辞書 A と辞書 B の現状から辞書 C を作成できたほうが便利でしょう。

diff による辞書のマージ

ファイルの差分を計算する diff コマンドを使えば、このような辞書のマージを簡単におこなうことができます。

diff は、通常は 2 つのファイルの差分だけを出力しますが、–D オプションを指定すると、#ifdef で相違部分を区別しつつ 2 つのファイルをマージした出力が得られます。

たとえば、上記の辞書 A と辞書 B の場合は以下のようにしてマージします。

```
% diff -DdictB dictA dictB
#ifndef dictB
aaa
#endif /* not dictB */
bbb
#ifdef dictB
aaa
#endif /* dictB */
XXX
%
```

この diff コマンドの出力から#ifdef などで始まる制御行と重複しているエントリを除去すると、

```
aaa
bbb
XXX
```

のようにマージされた辞書が得られます。

おわりに

長年、携帯端末とデスクトップ計算機で辞書を共有するアイデアを温めていましたが、最近になって運用実験を始めました。まだ十分な時間をかけたわけではありませんが、まったく同じ辞書をどこでも使える状態に近づきつつあります。

現在、私は PalmPilot でも Emacs でも 1MB 程度の POBox 辞書を使っています。以前なら辞書に 1MB も使うのはもったいないと思われたかもしれませんが、やはりサイズの大きい辞書のほうが変換効率がかなりよいようです。

PalmPilot とデスクトップ計算機とのあいだのデータ交換に、かなりの時間がかかる点も問題です。携帯端末から 1MB 以上の辞書を転送するには何分もかかりますし、辞書をマージして書き戻すにはその倍の時間がかかります。通信速度が限られている現状では、学習した部分だけを交換するような手法が必要かもしれません。

今回紹介したシステムは、すべて私の Web ページ[8]で公開しています。ぜひご利用ください。

（ますい・としゆき ソニー CSL）

8 http://www.csl.sony.co.jp/person/masui/OpenPOBox/