

テキストファイルをレコード型ファイルへ 変換する手法について

高 萩 栄一郎

Oo a conversion method from a text file into record type file

Eiichirou TAKAHAGI

ABSTRACT

We introduce a conversion method from a text file (for example, a file made by word processor) into the record type file which has fixed length records and fixed field location. The conversion process has 4 stages, that is (1) pickoing up strings from the text file; (2) separation of the storings into tokens; (3) recognition of fields consisting of the tokens; (4) assignment the fields into records. We explain the method using the example of an adress book.

1. はじめに

現在、多くの事業所では、O A化が進んでいるとはいえ、他の事業所間での情報の受渡しは、紙の帳票で、行われることが多い。現在、事業所間の電子的方法での受渡しは、系列間の取り引きなど少数である。電子的方法での取り引きは、事業所の戦略やコンピュータメーカの戦略などがからみ、統一した形で行われることは将来も難しいだろう。

しかしながら、これらの紙の帳票は、近い将来、光学技術の発達や文字認識技術の進歩により容易にテキストファイルとして、電子媒体に置き換えることができるだろう。また、事業所には、ワードプロセッサで作成された名簿や電子メールで送られてきたファイルなど多くの電子媒体のテキストファイルが存在する。これらのテキストファイルは、プリントアウトされたり、ディスプレイに表示されたりして、人間には認識できる。

しかしながら、このテキストファイルの意味情報対して、そのままデータ処理を施すことはできない。例えば、図1の左のようなテキストファイル中の金額(¥12,500)は、そのままでは集計等のデータ処理には使うことはできない。通常、図1の右のような、固定長のファイルのあるレコードの決った位置に「00012500」のように格納されていないと使うことはできない。本論文の目的は、図1のように、テキストファイル(左)を項目が定義されているレコード(右)に変換することである。

以下では、図1の領収書の例と図2の住所録ファイルの例を用いて説明していく。

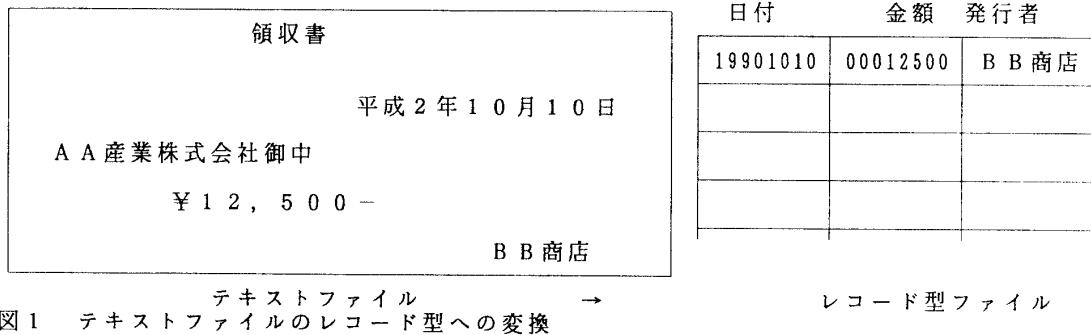


図1 テキストファイルのレコード型への変換

2. 処理の概要

処理の流れは、1) 1つのテキストファイルに1つのレコードの場合(例えば、図1のような領収書のテキストファイル进行处理する場合)と2) 1つのテキストファイルに複数のレコードが含まれている場合(例えば図2のような住所データ)は、処理の方法が異なる。それぞれの処理の方法について述べる。

1) 1つのテキストファイルに1つのレコードの場合は、次の5つの段階に分かれる。

(1) 前処理

(2) 文字列の切出し これは、空白をデミリットとして、テキストファイルを文字列の集まりに変換する作業である。

(3) 文字列のトークンへの分解 これは、1つの文字列をトークンに分解する。ここでトークンとは、この処理システム扱う文字列の最小単位をいう。例えば、都道府県名、姓、電話番号などがトークンである。

(4) トークンを結合して、項目を認識する。例えば、都道府県名、市町村名等を結合して、住所という項目を認識する。

(5) レコードの種類の認識 レコードの種類、すなわち、テキストファイルの種類を認識する。例えば、図1の領収書のテキストファイルは「領収書」という文字により領収書ファイルに格納されるべきだと認識する。

(6) レコードの出力 各ファイルについてあらかじめ定義した必要な項目を項目を出力する。(領収書の場合、日付、金額、発行者を出力する。)

2) 1つのテキストファイルに複数のレコードが含まれている場合。これは、図2のように1つのテキストファイルに複数のレコードを含む場合である。この場合、あらかじめ、レコードの種類(図2の例では、住所録)は、わかっているものとする。

大場浩之	福井繊維大学 TEL 0776-12-3456 FAX 0776-21-6543 〒910 福井県福井市銀座5-8-4
中西友行	北陸工業大学 TEL 0769-99-9877 〒910 福井県敦賀市大手3-9
福島英樹	福島鉄工所 TEL 0781-98-9876 FAX 0781-98-9999 〒910 福井県武生市西大田5-4 福島ビル1F

図2 住所データ

- (1) 前処理
- (2) 文字列の切出し
- (3) 文字列のトークンへの分解
- (4) トークンを結合して、項目を認識する。
- (5) 項目のレコードへの割り当て。 1つのテキストファイルに複数のレコードが含まれているので、(4)で認識した各項目を、いくつかのレコードに割り当てることである。
- (6) レコードの出力

3. 前処理および文字列の切出し

まず、前処理は後の処理の準備段階で、次の2つのことを主に行う。

- (1) 罫線の空白への変換
- (2) 全角の空白の半角の空白2文字への変換

これらの処理は、後の処理を容易にするためである。

つぎに、文字列の切出しを行う。空白をデミリットとして、文字列の抽出を行う。後の分析で、文字列がテキストファイルのどの位置から始めているかが重要な情報になるので、文字列の開始位置(X座標およびY座標)と長さをあわせて抽出する。

この分析は、AWKというプログラミング言語を利用して行った。AWKは、C言語ライクな記述ができるファイル処理言語であり、プロットタイプの作成に大変有用である。正規表現が使えるなど、文字列処理に優れているので、この分析に使用することにした。AWKについての詳しい説明は、参考文献[1]にある。主な処理の中心となるプログラムは、付録に掲載したので、参考にしていただきたい。

図2のテキストファイルを付録の「tomoji.awk」というプログラムで処理すると表3のような結果を得る。

1	1	8	大場浩之	15	8	8	〒910
15	1	12	福井繊維大学	25	8	22	福井県敦賀市大手3-9
23	2	24	0776-12-3456	1	10	8	福島英樹
15	3	6	FAX	15	10	10	福島鉄工所
23	3	24	0776-21-6543	15	11	6	TEL
15	4	8	〒910	23	11	24	0781-98-9876
25	4	26	福井県福井市銀座5-8-4	15	12	6	FAX
1	6	8	中西友行	23	12	24	0781-98-9999

15 6 12 北陸工業大学	15 13 8 〒9 1 0
15 7 6 TEL	25 13 24 福井県武生市西大田5-4
23 7 24 0 7 6 9-9 9-9 8 7 7	25 14 12 福島ビル1 F

スペースで区切られた欄は左からX座標、Y座標、長さ、文字列をあらわす。
表3 文字列を抽出した結果

以下の分析で、処理を容易にするため、全角文字のうち、半角文字に変換できるものは、変換しておく。

4. 文字列のトークンへの分解

文字列をさらに細かい単位のトークンに分解する。トークンとは、この処理システム扱う文字列の最小単位である。ほぼ単語に相当する。例えば、都道府県名、姓、電話番号などが当てはまる。

このトークンへの分解は、正規表現を利用すると便利である。また、トークンは、1つの文字列が1つのトークンになる場合もあるし、複数のトークンに分割される場合もある。このことも認識に役立てることができる。例えば、電話番号は、1つの文字列が1つのトークンになり、住所の番地の部分（例えば「5-8-4」）と識別することができる。電話番号は、正規表現で表わすと $/^0[0-9]+-[0-9]+-[0-9]+$/$ とか、 $/^0[0-9]+Y([0-9]+Y)[0-9]+$/$ となる。

次に、辞書の利用である。都道府県名、地名、姓、名などの切出しには、辞書を用いる。文字列を辞書の単語に照らしあわせていき、一致したものをトークンとする。例えば、表4のような辞書が存在するとき、「福井県福井市銀座5-8-4」という文字列は、福井県という都道府県名のトークンと「福井市」という「市町村名」のトークン、「銀座」という地名のトークンを認識する。

福井県	都道府県名	大阪府	都道府県名	銀座	地名
福井市	市町村名	大富山	都道府県名	大手	地名
敦賀市	市町村名	石川県	都道府県名	西大田	地名
武生市	市町村名	新潟県	都道府県名	学園	地名
鯖江市	市町村	滋賀県	都道府県名	大名町	地名

表4 辞書の例

これらで、認識できない部分は、「不明」というトークンにしておく。

この例題を処理するために実験的に作った「totoken.awk」を付録にのせた。このプログラムを使って、トークンの抽出を行うと、表5のようになる。

1 1 1 1 8 姓	大場	14 1 25 8 22 都道府県名	福井県
1 2 1 1 8 名	浩之	14 2 25 8 22 市町村名	敦賀市
2 1 15 1 12 地名	福井	14 3 25 8 22 地名	大手
2 2 15 1 12 一般名詞	繊維	14 4 25 8 22 番地	3-9
2 3 15 1 12 事業所キーワード	大学	15 1 1 10 8 姓	福島
3 1 15 2 6 TELキーワード	TEL	15 2 1 10 8 名	英樹
4 1 23 2 24 電話番号	0776-12-3456	16 1 15 10 10 姓	福島
5 1 15 3 6 FAXキーワード	FAX	16 2 15 10 10 事業所キーワード	鉄工所
6 1 23 3 24 電話番号	0776-21-6543	17 1 15 11 6 TELキーワード	TEL
7 1 15 4 8 郵便番号	〒910	18 1 23 11 24 電話番号	0781-98-9876
8 1 25 4 26 都道府県名	福井県	19 1 15 12 6 FAXキーワード	FAX
8 2 25 4 26 市町村名	福井市	20 1 23 12 24 電話番号	0781-98-9999
8 3 25 4 26 地名	銀座	21 1 15 13 8 郵便番号	〒910
8 4 25 4 26 番地	5-8-4	22 1 25 13 24 都道府県名	福井県

テキストファイルをレコード型ファイルへ変換する手法について

9 1 1 6 8 姓	中西	22 2 25 13 24 市町村名	武生市
9 2 1 6 8 名	友行	22 3 25 13 24 地名	西大田
10 1 15 6 12 地名	北陸	22 4 25 13 24 番地	5-4
10 2 15 6 12 一般名詞	工業	23 1 25 14 12 姓	福島
10 3 15 6 12 事業所キーワード	大学	23 2 25 14 12 地名	ビル
11 1 15 7 6 TELキーワード	TEL	23 3 25 14 12 番地	1F
12 1 23 7 24 電話番号	0769-99-9877		
13 1 15 8 8 郵便番号	〒910		

スペースで区切られた欄は左から文字列の番号、文字列内の何番目のトークンか、文字列のX座標、文字列のY座標、文字列の長さ、トークンの種類、トークンの文字列をあらわす。

表5 トークンを抽出した結果

5. トークンの結合

次にいくつかのトークンを結合して、項目を作成する。この項目は、レコードの項目に対応する。例えば、都道府県名、市町村名、地名、番地等を結合して、住所という項目を認識する。また、FAXキーワードと電話番号を結合して、FAX番号を認識する。

まず、項目がどのようなトークンから成り立っているかを定義する。その際、必ず必要なトークンとそうでないトークン（項目の一部となっているトークン）を区別して考える。また、だぶってはいけないトークンも定義しておく。住所データの場合の必要な項目をまとめると表6のようになる。

項目名	必ず必要なトークン	項目の一部になるトークン	だぶってはいけないトークン
氏名	姓、名		姓、名
電話番号	電話番号	電話番号キーワード	電話番号
FAX番号	電話番号 FAXキーワード		電話番号
郵便番号	郵便番号		郵便番号
事業所名	事業所キーワード	一般名詞、姓、地名	
住所	市町村名	都道府県名、地名、番地、姓	市町村名

表6 項目の定義

1つの項目が2つ以上の文字列に分断されている可能性がある。その時は、その文字列を連結しなくてはならない。連結する文字列は、まず、1つの文字列の左右の文字列が考えられる。何らかの原因で、トークン間に空白が入っている場合である。次に、文字列が2行にわたる場合を考えなくてはならない。表3では、最後の2行のデータである「福島県武生市西大田5-4」と「福島ビル1F」は、結合しなくてはならない。文字列が2行にわたる場合、人間がテキストファイル、もしくはその元となる帳票を作成する際、常識的に文字列のX座標の開始位置をそろえるので直前、直後の行で同じX座標で始まっている文字列は、結合できる文字列の候補である。

次のようなアルゴリズムにしたがって、項目の認識処理を行なう。

WHILE (項目として認識できる可能性がある文字列が存在) BEGIN

A := (ある文字列のトークンの集合)

```

PERFORM SUB
K := KK
WHILE (結合する文字列の候補が存在) BEGIN
  AA := A + (結合する文字列の候補の1つ)
  PERFORM SUB
  K1 := KK ; A1 := AA
  AA := AA - (項目K1を認識するのに必ず必要なトークンの集合)
  PERFORM SUB
  IF ((FE == 1) かつ (必要なトークン数(K1) >= 必要なトークン数(K)))
    A := A1 ; K := K1
  END IF
END WHILE
AをKとして認識
END WHILE
SUB :          # 文字列AAがどれかの項目に当てはまるかどうか調べるサブルーチン
FOR (KWに全ての項目を順に代入) BEGIN
  AAとKWの当てはまり具合を調べる
    だぶってはいけないトークンがだぶっている → FD(KW) = 1 ELSE FD(KW) = 0
    必ず必要なトークンがすべて存在          → FT(KW) = 1 ELSE FT(KW) = 0
NEXT
IF ((FD(KK) = 0) かつ (FT(KK) = 1) となるKKが存在)
  KK := ((FD(KK) = 0) かつ (FT(KK) = 1) で、FT(KK) が最大となる
        KK)
  EF := 0
ELSE
  EF := 1
END IF
RETURN

```

結合する文字列の候補とは、採用されている文字列のすぐ右、もしくはすぐ左の文字列と開始X座標が等しい前後のY座標の文字列をいう。

この処理を行うときに、各項目毎に、標準的なデータ形式に変換しておく。例えば、電話番号は、半角の数字と'-'で構成される形に変換する。

この処理を行うと表5のデータは、表7のデータに変換される。

1	1	8	1	氏名	大場浩之
15	1	26	1	事業所名	福井繊維大学

テキストファイルをレコード型ファイルへ変換する手法について

15	2	46	2	電話番号	0776-12-3456
15	3	46	2	FAX番号	0776-21-6543
15	4	22	4	郵便番号	910
25	4	50	4	住所	福井県福井市銀座5-8-4
1	6	8	6	氏名	中西友行
15	6	26	6	事業所名	北陸工業大学
15	7	46	7	電話番号	0769-99-9877
15	8	22	8	郵便番号	910
25	8	32	8	住所	福井県敦賀市大手3-9
1	10	8	10	氏名	福島英樹
15	10	24	10	事業所	福島鉄工所
15	11	46	11	電話番号	0781-98-9876
15	12	46	12	FAX番号	0781-98-9999
15	13	32	13	郵便番号	910
25	13	48	14	住所	福井県武生市西大田5-4福島ビル1F

スペースで区切られた欄は左から、項目の開始X座標、開始Y座標、終了X座標、終了Y座標、項目の種類、文字列をあらわす。

表7 項目を認識した結果

6. レコードの種類の認識

1つのテキストファイルに1つのレコードの場合、レコードの種類があらかじめ与えられていないので、レコードの種類を認識しなくてはならない。

レコードの種類は、通常、テキストファイルにキーワードが書かれている。このキーワードの存在を利用して、認識を行う。図1の例では、「領収書」という文字列が書かれている。トークンの抽出で、このキーワードを帳票種キーワードして認識する。（辞書に加えることで、簡単にできる。）例えば、「領収書」「領収証書」「領収」等は、「帳票種キーワード（領収書）」として認識される。これは、項目の認識についてもそのまま引き継がれる。レコードの種類の認識は、この帳票種キーワードの存在で判断される。

この認識では、キーワードが複数存在することを考慮していない。この場合、キーワードの座標や文字列がこのキーワード単独で成り立っているかどうかということによって、特定することが考えられる。

7. 項目のレコードへの割り当て

1つのテキストファイルに複数のレコードが含まれている場合は、認識された項目が何番目のレコードに所属するのかを決定しなくてはならない。

まず、1つのレコードに2つ以上含まれない項目を決めておく。例えば、住所データでは、すべての項目が1つのレコードに2つ以上含まれない項目とする。

テキストファイル中のレコードの切れ目は、行単位である。すなわち、はじめから何行目までがレコード1で、次の行から何行がレコード2であるといった具合である。また、空白行が、存在するとその前後がレコードの切れ目である可能性が高い。

項目をテキストファイルの上（Y座標の小さい方）から、眺めていき、1つのレコードに2つ以上含まれない項目が2回出現した時点で、2回目に出現した項目は次のレコードの項目であることがわかる。この2回目に出現した項目とこの直前の1つしか含んではいけない項目間の2つ以上含んでもよい項目が、認識中のレコードに含まれるか次のレコードの含まれるか不明の項目

である。このような項目は、そのY座標、空白行の存在やその項目の出現の仕方、で、判断する。
結果は、表8のようになる。

1 氏名	大場浩之	2 電話番号	0769-99-9877
1 事業所名	福井繊維大学	2 郵便番号	910
1 電話番号	0776-12-3456	2 住所	福井県敦賀市大手3-9
1 F A X 番号	0776-21-6543	3 氏名	福島英樹
1 郵便番号	910	3 事業所	福島鉄工所
1 住所	福井県福井市銀座5-8-4	3 電話番号	0781-98-9876
2 氏名	中西友行	3 F A X 番号	0781-98-9999
2 事業所名	北陸工業大学	3 郵便番号	910
		3 住所	福井県武生市西大田5-4福島ビル1F

スペースで区切られた欄は左から、レコード番号、項目種類、文字列をあらわす。
表8 項目のレコードへの割り当てをした結果

8. おわりに

テキストファイルをレコード型のファイルに変換する手法を提案した。住所データ（図2）で、実験を行い、表8のような結果を得た。今後、より多くのデータに適用して、この手法の有用性を示していきたい。

以上で紹介した手法についてまだまだ不十分な点がある。改善すべき点として以下の点が考えられる。

（1）項目の認識において、トークンの順序を考えていない。項目の定義においてトークンの順序も指定できるようにし、順序にしたがったもののみを認識するようにしなくてはならない。その際、文脈自由文法にもとづいたコンパイラコンパイラの利用も考えられる。

（2）項目の認識において、項目のテキストファイル内の位置を考えていない。（1）で、ある文字列内だけで認識した後、表頭の文字列などを利用して、項目の種類をより細かいものにする。例えば、「電話番号」という項目を、同じX座標の「勤務先」という表頭を利用して「勤務先電話番号」として認識する。

（3）段組されたテキストファイルを考えていない。この場合、項目のレコードへの割り当てで、この段組を認識するようにしなくてはならない。

参考文献

- Aho, Alfred V., Brian W. Kernighan, Peter J. Weinberger, 足立高德訳 「プログラミング言語 A W K」, トッパン, 1989
- Aho, Alfred V., Ravi Sethi, Jeffrey D. Ullman, 原田賢一訳 「コンパイラ 原理・技法・ツール」, サイエンス社, 1990

付録（AWKでのプログラム）

（１）文字列の切出し

```
{
    for ( i = 1 ; i <= NF ; i++ ) {
        match( $0 , $i )
        printf("%5d %5d %5d %s\n", RSTART, NR, RLENGTH, $i)
    }
}
```

（２）トークンへの分解

```
BEGIN {
    while ( getline <"jisyo.dic" > 0 ) {
        tango[++p] = $1
        tango_syu[p] = $2
    }
    tango[++p] = "([0-9]+-)*[0-9]"
    tango_syu[p] = "番地"
    tango[++p] = "[0-9]+(F|階|号|号室)"
    tango_syu[p] = "番地"
    tango_n = p
    NR = 0
}
# 1つの文字列が1つのトークンになる場合

$4 ~ /^0[0-9]+-[0-9]+-[0-9]+$ / {
    printf( "%5d %5d %5d %5d %5d %-20s %s\n", NR, 1, $1, $2, $3, "
電話番号", $4 )
    next
}

$4 ~ /^〒?[0-9][0-9][0-9](-[0-9][0-9])?$/ {
    printf( "%5d %5d %5d %5d %5d %-20s %s\n", NR, 1, $1, $2, $3, "郵便番号",
$4 )
    next
}
```

#複数のトークンに分割される場合

```
{
    tt = $4
    count = 0
    while( tt != "" ) {
        count++
        max_point = 1000
        max_len = 0
        max_i = 0
        for ( i = 1 ; i <= tango_n ; i++ ) {
            if ( match( tt , tango[i] ) != 0 ) {
                if ( ( RSTART < max_point ) || (( RSTART == max_point ) && ( RLENGTH >= max_len ))) {
                    max_i = i
                    max_len = RLENGTH
                    max_point = RSTART
                }
            }
        }
        if ( max_i == 0 ) {
            printf("%5d %5d %5d %5d %5d %-20s %s\n", NR, count, $1, $2, $3, "不明", tt)
            tt = ""
        }
        else {
            if ( max_point != 1 ) {
                printf("%5d %5d %5d %5d %5d %-20s %s\n", NR, count, $1, $2, $3, "不明", substr( tt , 1 , max_point - 1 ))
            }
            printf("%5d %5d %5d %5d %5d %-20s %s\n", NR, count, $1, $2, $3, tango_syu[max_i], substr( tt , max_point , max_len ))
            tt = substr( tt , max_point + max_len )
        }
    }
}
```

(平成 2 年12月20日 受理)