1　ファイル処理　ファイル処理の概要　226 頁

1.1　学習のポイント

ファイル・オープン、ファイル入出力などのファイル処理を行なう上で基本となる考え方について学びます。

1.2　ファイル・ポインタとファイル・オープン

ファイルはファイル名により識別されますが、そのファイルをアクセスするためには、ファイル・ポインタに対しファイル・オープンしておかなければなりません。ファイル・ポインタはファイル操作するための特別な識別子で、構造体 FILE 型 (stdio.h の中で定義されている) のポインタ変数として次のように宣言します。

```
FILE *fp;
```

こうしておいて、ファイル abc.txt をリード・モード (読み込みモード) でオープンするには、ファイル・オープン関数 fopen を用いて次のように行います。

```
fp = fopen("abc.txt", "r");
```

こうすることによりファイル・ポインタ fp を用いて abc.txt にファイル・アクセスできるようになります。なお、一般的に fopen 関数はファイル・オープンに失敗すると NULL(0) という値を返すことになっているので、ファイルは、次のように行なっておくのが安全です。何故ならファイル・オープンに失敗したまま正常でないファイル・ポインタ fp を用いてファイル・アクセスを行なえば、ファイルを壊す危険があるからです。

```
#include <stdio.h>
void main(void)
{
  FILE  *fp;

  if((fp = fopen("abc.txt", "r")) == NULL){
    printf("File not open\n");
    exit(1);
  }
}
```

構造体 FILE は stdio.h の中で定義されているので必ず stdio.h を取り込まなければなりません。

2 一文字単位のファイル入出力

2.1 学習のポイント

1 文字単位のファイル入出力関数 getc、putc について学びます。

2.2 例題 50 reidai50.c 231 頁

キーボードから入力した文字を 1 文字ずつファイル abc.txt にライトしなさい。

```
/*
  キーボードから入力した文字を 1 文字ずつファイル abc.txt にライトしなさい。

  reidai50.c
*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
  FILE *fp;
  int  c;

  if((fp = fopen("abc.txt", "w")) == NULL){
    printf("Can't open File\n");
    exit(1);
  }

  while((c = getchar()) != EOF){
    putc(c, fp);
  }

  fclose(fp);

  return 0;
```

```
}
```

2.3 練習問題 50 rensyu50.c 231 頁

```
/*
  例題 50 で作ったファイル abc.txt を 1 文字ずつリードして、ディスプレイに
表示して行くプログラムを作りなさい。
  rensyu50.c
*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
  FILE *fp;
  int  c;

  if((fp = fopen("abc.txt", "r")) == NULL){
    printf("Can't open File\n");
    exit(1);
  }

  while((c = getc(fp)) != EOF){
    printf("%c", c);
  }

  fclose(fp);

  return 0;
}
```

3 AI による大規模データ処理入門

3.1 言語処理アルゴリズム

自然言語で記述された大規模データから、その特徴を抽出する方法を考えます。人工知能の歴史においては、自然言語処理の研究は、機械翻訳や自動要約などへの応用を念頭に

置いて、さまざまな側面から取り組まれてきました。

3.2 tf-idf とは

文書の特徴を表現する単語を得るには、冠詞や接続詞など一般によく用いられる単語ではなく、一般にはあまり出現しないのにその文書だけによく登場する単語を抽出しなければなりません。このためには、tf-idf(term frequency, inverse document frequency) という指標が提案されています。tf-idf は、ある単語の特定の文書における出現頻度と、多くの文書のある単語が、その文書の特徴をどの程度表現しているかということを表します。(AI による大規模データ処理入門　220 頁)

3.3 tf-idf の計算プログラム tfidf.c

```
/************************************/
/*     tf-idf の計算プログラム        */
/*        tfidf.c                    */
/************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

/*   記号定数                        */
#define NOFFILES 10 /*データファイル数上限*/
#define NOFWORDS 1000 /*単語数上限*/
#define WLENGTH 32 /*単語長の上限*/
#define LENGTH 256 /*文字列長の上限*/
#define EOD -2 /*データの終わりを示す記号*/

/*  単語の属性を表現する構造体         */
struct tuple{
 int freq ;/*出現頻度*/
 char word[WLENGTH] ;/*単語*/
 double tf ;
 double idf ;
} ;
```

```
/*関数のプロトタイプの宣言 */
void readfile(char *,struct tuple st[]) ;
                /* データ読み込み      */
int sumfreq(struct tuple st[]) ;
                /* 頻度合計の計算      */
void settf(struct tuple st[],int sumoffreq) ;
                   /* settf() 関数      */
void setidf(struct tuple t[][NOFWORDS]
                         ,int i,int argc)  ;
                    /* idf 値の計算      */
int count(char word[]
    ,struct tuple t[][NOFWORDS],int argc)  ;
                     /* 単語出現ファイル数*/
void writefile(int no,struct tuple st[]) ;
                    /* データ書き込み      */


/**********************/
/*     main() 関数     */
/**********************/
int main(int argc,char *argv[])
{
 int i;
 struct tuple t[NOFFILES][NOFWORDS] ;
 int sumoffreq[NOFFILES] ;/*頻度の合計*/

 printf("tfidf 開始\n");

 /*コマンドライン引数のチェック*/
 if(argc<3){/*引数が足りない*/
  printf("エラー\n");
  fprintf(stderr,"使用法  tfidf "
         "(ファイル 1) (ファイル 2) ・・・\n") ;
  exit(1) ;
 }
 else if(argc>NOFFILES){/*ファイルが多すぎる*/
  fprintf(stderr,"ファイルが多すぎます (%d)\n "
         ,argc) ;
  exit(1) ;
```

```
 }


 /*ファイルからのデータ読み込み*/
 for(i=0;i<(argc-1);++i){
  readfile(argv[i+1],t[i]) ;
 }

 /*頻度合計の計算*/
 for(i=0;i<(argc-1);++i){
  sumoffreq[i]=sumfreq(t[i]) ;
 }

 /*tf 値の計算*/
 for(i=0;i<(argc-1);++i){
  settf(t[i],sumoffreq[i]) ;
 }

 /*idf 値の計算*/
 for(i=0;i<(argc-1);++i){
  setidf(t,i,argc) ;
 }

 /*ファイルへのデータ書き込み*/
 for(i=0;i<(argc-1);++i){
  writefile(i,t[i]) ;
 }

 printf("tfidf 終了\n");

 return 0 ;
}

/**********************/
/* writefile() 関数    */
/* データ書き込み      */
/**********************/
void writefile(int no,struct tuple st[])
{
 FILE *fp ;/*ファイルポインタ*/
```

```
 char filename[LENGTH] ;/*ファイル名*/
 int i=0 ;

 /*ファイル名は数字とする*/
 sprintf(filename,"%d.txt",no) ;

 /*ファイルオープン*/
 if((fp=fopen(filename,"w"))==NULL){
  /*ファイルオープンのエラー*/
  printf("%s が開けません\n",filename) ;
  exit(1) ;
 }

 /*ファイルへの書き込み*/
  while(st[i].freq!=EOD){
   fprintf(fp,"%lf %s\n",st[i].tf*st[i].idf
                                  ,st[i].word) ;
    ++ i ;
  }

 /*ファイルクローズ*/
 fclose(fp) ;
}

/**********************/
/* setidf() 関数       */
/* idf 値の計算        */
/**********************/
void setidf(struct tuple t[][NOFWORDS]
                       ,int no,int argc)
{
 int i=0 ;

 while(t[no][i].freq!=EOD){
  t[no][i].idf=log(argc/
    (double)count(t[no][i].word,t,argc)) ;
  ++i ;
 }
}
```

```
/**********************/
/*  count() 関数       */
/* 単語出現ファイル数 */
/**********************/
int count(char word[]
    ,struct tuple t[][NOFWORDS],int argc)
{
 int cnt=0 ;/*出現数*/
 int i,j ;

 for(i=0;i<argc-1;++i){
  j=0 ;
  while(t[i][j].freq!=EOD){
   if(strcmp(word,t[i][j].word)==0){
    ++cnt ;
    break ;
   }
   ++j ;
  }
 }
 return cnt ;
}


/**********************/
/* settf() 関数        */
/* tf 値の計算         */
/**********************/
void settf(struct tuple st[],int sumoffreq)
{
 int i=0 ;

 while(st[i].freq!=EOD){
  st[i].tf=st[i].freq/(double)sumoffreq ;
  ++ i ;
 }
}

/**********************/
/* sumfreq() 関数      */
```

```
/* 頻度合計の計算      */
/**********************/
int sumfreq(struct tuple st[])
{
 int i=0 ;
 int sum=0 ;

 while(st[i].freq!=EOD){
  sum+=st[i].freq ;
  ++ i ;
 }

 return sum ;
}

/**********************/
/* readfile() 関数     */
/* データ読み込み      */
/**********************/
void readfile(char *filename,struct tuple st[])
{
 int i=0 ;
 FILE *fp ;/*ファイルポインタ*/
 char buffer[LENGTH] ;/*入力バッファ*/

 /*ファイルオープン*/
 if((fp=fopen(filename,"r"))==NULL){
  /*ファイルオープンのエラー*/
  printf("%s が開けません\n",filename) ;
  exit(1) ;
 }

 /*ファイルからの読み込み*/
 while(fgets(buffer,LENGTH,fp)!=NULL){
  if(sscanf(buffer ,"%d %s",&(st[i].freq)
              ,st[i].word)!=0)
   /*変換できた*/
  if(strlen(st[i].word)!=0)
   ++ i;/*次の位置へ*/
  if(i>=(NOFWORDS-1)){/*単語数の上限を超えた*/
```

```
    fprintf(stderr,"単語数が多すぎます "
          "(%s)\n",filename) ;
    break ;/*読み込みの打ち切り*/
   }
  }
  /*ファイルの終わりの記号*/
  st[i].freq=EOD ;

  /*ファイルクローズ*/
  fclose(fp) ;
 }
```

3.4 プログラムのコンパイルの方法

```
gcc tfidf.c -o tfidf -lm
```

3.5 tfidf.exe の使い方

```
./tfidf s?.txt
```

tfidf 計算結果は、0.txt、1.txt、2.txt に書き込まれています。