3.1 PHPだけで自然言語文を解析してみよう

PHPが提供する文字列処理の関数を組み合わせてアルゴリズムを実装することで、自然言語文を形態素の塊に分割することができます。使う関数の種類、アルゴリズムのパターンを学ぶことにより、形態素解析処理が内部にどのようなロジックを持っているのか理解することができるでしょう。

なお、欧米のアルファベットを基礎とする言語と日本語では、形態素解析処理のアルゴリズムが大きく異なります。まず英語の形態素解析処理を、次に日本語の形態素解析処理の順に解説をしていきます。

3.1.1 英語の文章を単語に分解してみよう

英語は、シンプルで明快な構造の言語です。使用される主な文字は、アルファベットと記号と数字であり、プログラミングでもおなじみの文字です。まずは英語の文章を題材として、形態素に分解する方法について学びましょう。

3.1.1.1 文章を単語に分割してみよう

英語は、単語と単語の間に空白を設ける言語です。英語の文を空白文字で分割するだけでも、単語を切り出すことができます。では次の英語の文を例文にして考えてみましょう。

●例文：「自然言語処理の定義」

> Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.
> （訳：自然言語処理（NLP）は、コンピュータ科学、人工知能、コンピュータと人間（自然）言語間の相互作用に関する言語学の分野である。）

出展：Wikipedia（http://en.wikipedia.org/wiki/Natural_language_processing）

この英語の文は、Wikipedia英語版「自然言語処理」の解説文の１つです。文を構成する英単語同士の間には空白が設置されていることがわかります。空白は多くの場合、コンピュータ上では半角スペースで表されます。

半角スペースを境目として、英文を英単語の集合に切り離すことができます。ここでいう半角スペースのことを「区切り文字(separator)」と呼びます。この「区切り文字」を使って、英文のテキストを英単語に分割するには、PHPの文字処理関数explode()関数を使います。explode()関数の使い方は、以下のとおりです。

```
分割結果 = explode( 区切り文字 , 分割したい文字列 , 最大分割数 );
```

〈解 説〉

- 分割結果は、配列変数です。配列変数の要素数は、分割数です。
- 区切り文字は、文字列型変数です。今回の場合、半角スペースを格納した文字列変数です。
- 分割したい文字列は、文字列型変数です。今回の場合、英文のテキストを格納した文字列変数です。
- 最大分割数は、デフォルトでは指定なしとなっています。通常は指定しないでおきます。設定ファイルの記述の解析など、分割数に制約がある場合のみ使います。

サンプルプログラムとしてcode-3-1-1-1.phpを使います。WinSCPを使い、開発サーバのディレクトリ/home/nlp_ai/training_codeにcode-3-1-1-1.phpをコピーします。コピーしたら、Tera Termで開発サーバにログインして、コマンドラインからcode-3-1-1-1.phpを実行して下さい。

●英文を単語に分割するサンプルプログラムcode-3-1-1-1.phpの実行

```
[nlp_ai@localhost ~]$ cd training_code/
[nlp_ai@localhost training_code]$ php code-3-1-1-1.php
[ 元の英文 ]
Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics
concerned with the interactions between computers and human (natural) languages.

[ 分割で得られた単語数 ] 24
( 1 番目 ) Natural
( 2 番目 ) language
( 3 番目 ) processing
( 4 番目 ) (NLP)
( 5 番目 ) is
( 6 番目 ) a
( 7 番目 ) field
( 8 番目 ) of
( 9 番目 ) computer
( 10 番目 ) science,
( 11 番目 ) artificial
( 12 番目 ) intelligence,
```

```
( 13 番目 ) and
( 14 番目 ) linguistics
( 15 番目 ) concerned
( 16 番目 ) with
( 17 番目 ) the
( 18 番目 ) interactions
( 19 番目 ) between
( 20 番目 ) computers
( 21 番目 ) and
( 22 番目 ) human
( 23 番目 ) (natural)
( 24 番目 ) languages.
```

与えられた英文が24個の単語に分割されたことがわかります。それでは、サンプルプログラムcode-3-1-1-1.phpの中を見ていきましょう。

❶分割結果は、配列変数 $chunkResult に格納されています。配列の要素数は24個です。

❷文字列変数 $targetText には、"Natural language processing (NLP) is ..." という英文がセットされています。

❸文字列変数 $separator には、半角スペース1文字がセットされています。

❹最大分割数は指定しないので、explode()関数の引数は2個（$separatorと$targetText）です。もし最大分割数を指定してしまうと、単語数が多い場合に支障をきたします。具体的には、（最大分割数）番目以降の単語は全てくっついた状態で結果が返ってきてしまうのです。含まれている単語数がわからないことがほとんどなので、通常は最大分割数は指定しないようにしましょう。

```
$chunkResult  = explode( $separator , $targetText );
```

❺配列変数$chunkResultの要素数は、下記のようにsizeof()関数を使うことで取得することができます。

```
printf( "[分割で得られた単語数] %d¥n" , sizeof( $chunkResult ) );
```

❻foreach構文を使って配列変数 $chunkResult の内容を画面に表示します。print_r()関数で表示するより、コンパクトで目に優しい表示にします。

❼foreach文の個所において、"$eachPos => $eachWord" とあります。$eachPosには配列変数の要素番号（0, 1, 2...）がセットされ、$eachWord には各要素の内容（英単語）がセットされます。

❽要素番号は0から始まる数字なので、＋1して補正しています。

```
/* 分割で得られた単語を表示する */
foreach( $chunkResult as $eachPos => $eachWord )
{
    printf( "( %d 番目 ) %s¥n" , $eachPos + 1 , $eachWord );
}
```

こうして見てくると、結局、実質的に 1 行で英文を単語に分割できたわけですが、分割された個々の単語を見ると単語の前後に括弧記号がくっついていたり、単語の末尾にカンマがくっついているものがあります。

ここで使った explode() 関数で得られた単語の集合は、鉱石から切り出したばかりの宝石の原石のようなものです。周りには砂や岩石といった不純物（単語の前後に付着している余計な文字）がこびりついているわけです。不純物をできるだけ取り除き、綺麗な単語の状態にする処理が必要になります。

そこで次項以降では、上記のような単語に付着した不純物を取り除くための処理について解説します。

3.1.1.2 取り出した単語をクリーニングしよう

前項のサンプルプログラムでは、与えられた英文を半角スペースで単語の集合に分割しました。しかし、単語の前後に余計な文字が付着しているケースが散見され、対処する必要がありました。

自然言語処理においては、目的とする文字列（本節においては単語）の前後に付着した余計な文字を掃除する処理のことを文字通り「クリーニング」と呼んでいます。

地味な処理に聞こえるかもしれませんが、このクリーニング処理は非常に重要です。コンピュータは人間とは違って、融通のきいた処理ができません。入力した文字列は厳密な判定処理にかけられます。そのため、余計な文字が付着していると、本当は同じ単語なのに別の単語だと見なしてしまうのです。

入力するデータ（単語の集合など）に、ゴミのように不要な文字が混じってしまうと、そうした誤認識・誤判定が積み重なります。最終的には、その後の機械学習処理やテキストマイニング処理の性能を大きく損なう危険性があるのです。

それでは、前項で紹介したサンプルプログラムを土台に、クリーニング処理の実装について考えていきましょう。サンプルプログラム code-3-1-1-1.php で得られた単語のうち、不純物を含む単語は次のとおりです。

（ 4 番目）(NLP)
（10 番目） science,
（12 番目） intelligence,

（23番目）(natural)

（24番目）languages.

単語の前後に付着している不要な文字は、半角の括弧 "(" と ")"、半角カンマ ","、半角ピリオド "." です。英文の構成要素としては必要ですが、自然言語処理をする際には不純物となる文字です。単語の先頭1文字と末尾1文字をチェックして、これら不純物の有無を判定していきましょう。

単語の先頭1文字、また末尾1文字を取り出すには、次のようにsubstr()関数を使います。

```
取り出した文字 = substr ( 対象文字列 , 開始位置 , 取り出す文字数 );
```

〈解 説〉

- 対象文字列において、開始位置から指定した文字数を取り出します。
- 開始位置は整数です。0が先頭を表します。
- 開始位置を負の値にすると、最後尾からの位置になります。
  例えば、開始位置を－1とすると、最後の文字が開始位置となります。
- 取り出す文字数を指定しない場合は、開始位置以降全ての部分を取り出します。

また、開始位置の調節に際して必要となるのが単語の文字列長です。単語の文字列長は、次のようにstrlen()関数で取得することができます。

```
文字列長　= strlen( 対象文字列 );
```

〈解 説〉

- 文字列長は整数です。

上記2つの関数、substr()関数とstrlen()関数を組み合わせて、単語に付着した不純物を取り除く処理をするには、以下のように考えるとよいでしょう。

(1) 単語の先頭に半角左括弧 "(" があった場合、先頭の文字を取り除く。
(2) 単語の末尾に半角右括弧 ")"、または半角カンマ ","、半角ピリオド "." があった場合、末尾1文字を取り除く。

英文からの単語の切り出し、及び単語に付着した不純物を取り除く処理を実装したのが、サンプルプログラムcode-3-1-1-2.phpです。このcode-3-1-1-2.phpを本書サポートサイトから入手して下さい。

WinSCPを使い、開発サーバのディレクトリ/home/nlp_ai/training_codeにcode-3-1-1-2.phpをコピーします。コピーしたら、Tera Termで開発サーバにログインして、コマンドラインからcode-3-1-1-2.phpを実行して下さい。

●英文を単語に分割し、単語に付着した不純物を除去するサンプルプログラム**code-3-1-1-2.php**の実行

```
[nlp_ai@localhost ~]$ cd training_code/
[nlp_ai@localhost training_code]$ php code-3-1-1-2.php
[ 元の英文 ]
Natural language processing (NLP) is a field of computer science, artificial intelligence, and
linguistics concerned with the interactions between computers and human (natural) languages.

[ 分割で得られた単語数 ] 24
( 1 番目 ) Natural
( 2 番目 ) language
( 3 番目 ) processing
( 4 番目 ) NLP
( 5 番目 ) is
( 6 番目 ) a
( 7 番目 ) field
( 8 番目 ) of
( 9 番目 ) computer
( 10 番目 ) science
( 11 番目 ) artificial
( 12 番目 ) intelligence
( 13 番目 ) and
( 14 番目 ) linguistics
( 15 番目 ) concerned
( 16 番目 ) with
( 17 番目 ) the
( 18 番目 ) interactions
( 19 番目 ) between
( 20 番目 ) computers
( 21 番目 ) and
( 22 番目 ) human
( 23 番目 ) natural
( 24 番目 ) languages
```

実行結果を見ると、単語の前後に付着していた半角括弧やカンマ、ピリオドといった不純物が除去されていることがわかります。それでは、サンプルプログラムcode-3-1-1-2.phpの中身を見ていきましょう。

❶英文を単語に分割するところまでは、前項のサンプルプログラムと同じです。新たにロジックを加えた個所は、単語のリストを表示するforeachループ処理の部分です。

```
/* 分割で得られた単語を表示する */
foreach( $chunkResult as $eachPos => $eachWord )
{
    /* 単語の先頭に不純物が付着している場合は除去する */
```

```
    if( substr( $eachWord , 0 , 1 ) === '(' ){
        /* 単語の2文字目以降を取り出し、単語とする */
        $eachWord  = substr( $eachWord , 1 );
    }

    /* 単語の末尾に不純物が付着している場合は除去する */
    $eachTail  = substr( $eachWord , -1 , 1 );
    if( $eachTail === ')' || $eachTail === ',' || $eachTail === '.' ){
        /* 末尾の文字の手前までを取り出して、単語とする */
        $eachLength = strlen( $eachWord );
        $eachWord  = substr( $eachWord , 0 , $eachLength - 1 );
    }

    printf( "( %d 番目 ) %s¥n" , $eachPos + 1 , $eachWord );
}
```

❷foreachループで取り出された各単語は$eachWordにセットされています。まず最初に、$eachWordの先頭1文字を取り出し、それが半角左括弧かどうか判定します。そして、半角左括弧だった場合は、単語の2文字目以降を取り出し、それを$eachWordにセットします。

```
/* 単語の先頭に不純物が付着している場合は除去する */
if( substr( $eachWord , 0 , 1 ) === '(' ){
    /* 単語の2文字目以降を取り出し、単語とする */
    $eachWord  = substr( $eachWord , 1 );
}
```

❸次に、$eachWordの末尾1文字を取り出し、それが半角右括弧、半角カンマ、半角ピリオドのいずれかに該当するか判定します。もしいずれかに該当する場合は、末尾の文字の手前までを取り出して、それを$eachWordにセットします。

❹substr()関数で取り出す文字数を（$eachWordの文字列長－1）とすることで、末尾の手前までを取り出しています。

```
/* 単語の末尾に不純物が付着している場合は除去する */
$eachTail  = substr( $eachWord , -1 , 1 );
if( $eachTail === ')' || $eachTail === ',' || $eachTail === '.' ){
    /* 末尾の文字の手前までを取り出して、単語とする */
    $eachLength = strlen( $eachWord );
    $eachWord  = substr( $eachWord , 0 , $eachLength - 1 );
}
```