HITsにおけるWord文書の採点プログラム2013年度版の開発

長谷部　勝　也
松　井　吉　光
谷　口　正　明

要　　約：本稿はHITsのWord部門における、文書解析・採点プログラムwt2013.pyの解説である。このプログラムは2013年度の情報リテラシー入門・応用で稼働しており、拡張子.pyから分かる様にPythonで記述されている。

キーワード：e-Learning、教材、自動採点、Word、Python

1 はじめに

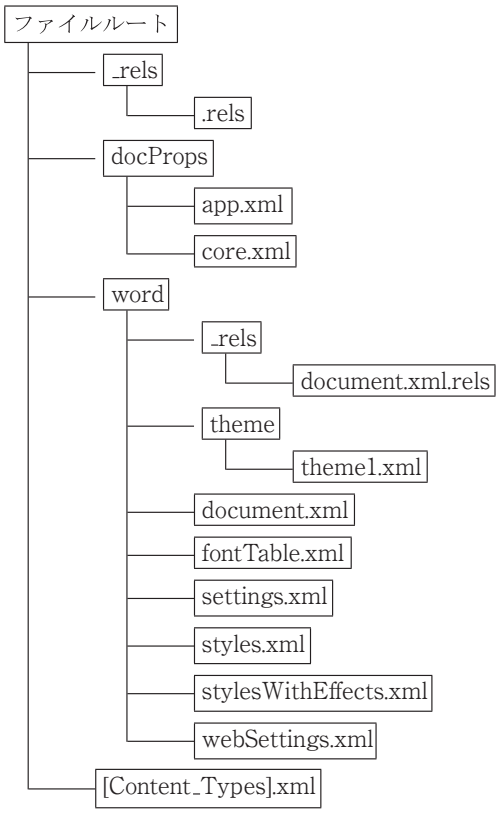
本学名古屋校舎では、2006年度から学生の情報リテラシーレベルの向上を目的として情報リテラシー入門・応用を開講している。教材は独自開発したHITs（Highly Interactive Training system）である。これはWebサーバ上で動作する学習・自動採点・記録（e-Learning）システムであり、Microsoft OfficeのWordとExcelをサポートする。即ちWebサーバが蓄積する課題ファイルを学生がダウンロードし指示に従って編集してアップロードすると正解/不正解（正しく指示に従ったか否か）を表示する。また、タイピング練習も自動的に行い、タイピングの正確さの採点と速度の計測を行う。そして全ての情報（学生のID、学習日時、判定結果、タイピング速度、アップロードされたファイル）を収録し、管理する。HITsのインターフェースとデータベースの処理にはPHP+PostgreSQLを使用している。本稿は、HITsのWord部門の自動判定プログラムwt2013.pyの解説である。

2 Microsoft Word文書ファイルの構造

wt2013.pyの採点対象は、Word2007、Word2010、Word2013で作成された.docxファイルである。.docxファイルにはXML形式が採用されており、ファイルフォーマットはOffice Open XMLFile Formatsとして ECMA[1]から公開されている。

.docxファイルはいくつかのXML形式のファイルをZIPを用いて圧縮したもので、新規で作成された.docxファイルの中身をみると図のような木構造になっている。

文書のプロパティ、すなわち文書の所有者、更新時間などの情報はdocProps/core.xmlに格納される。文書に様々な操作、例えば文字を入力したり、修飾を加えたりしていくと、追加された情報がword/document.xmlに格納されていく。但し、いくつかの操作では新たにファイルが追加され、その中に情報が格納される。例えば、文書にフッター、ヘッダーを加えるとfooter1.xml、footer2.xml、footer3.xml、header1.xml、header2.xml、header3.xml等のファイルのどれかが作成さ



れ、脚注を加えるとfootnote.xmlが、文末脚注を加えるとendnote.xmlが作成されてword/ディレクトリ内に追加される。また文書に画像ファイルを挿入すると、word/ディレクトリ内にmedia/ディレクトリが作成され、その中に画像ファイルが格納される。

3 SampleQ.docxとSampleA.docx

以下はWord文書の一例（SampleQ.docx）である。

平成25年○月○日
新工場開設のお知らせ

即ち第１段落に「平成25年○月○日」と入力し、第２段落に「新工場開設のお知らせ」と入力したもので文字修飾は全く加えていない。

そのdocument.xmlの木構造を一部省略して示す。

```
w:document
  w:body
    w:p[1]
      w:pPr
        w:rPr
          w:rFonts
            w:hint=eastAsia
      w:r[1]
        w:t
          平成25年○月○日
    w:p[2]
      w:r
        w:rPr
          w:rFonts
            w:hint=eastAsia
        w:t
          新工場開設のお知らせ
    w:sectPr
      ...
```

次（SampleA.docx）はこの文書に以下の修飾を加えた場合である。即ち第１段落「平成25年○月○日」全体を右揃えにし、第２段落「新工場開設のお知らせ」のフォントサイズを12ptにして太字にして下線を引き、第２段落全体を中央揃えにしている。

平成25 年○月○日

新工場開設のお知らせ

この時のdocument.xmlファイルの内容はおおよそ以下のようになっている。

```
<w:document>
  <w:body>
    <w:p>
      <w:pPr>
        <w:jc w:val="right" />
       <w:r>
         <w:rPr>
```

```
          <w:rFonts w:hint="eastAsia"/>
         </w:rPr>
         <w:t>
           平成25 年〇月〇日
         </w:t>
       </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:jc w:val="center" />
       <w:r>
         <w:rPr>
           <w:rFonts w:hint="eastAsia"/>
           <w:b />
           <w:sz w:val="24" />
           <w:u w:val="single" />
         </w:rPr>
         <w:t>
           新工場開設のお知らせ
         </w:t>
       </w:r>
    <w:p>
    </w:p>
    <w:sectPr>
     ...
    </w:sectPr>
  </w:body>
</w:document>
```

w:documentはdocument.xmlファイルのルートエレメントである。w:bodyは本文を表している。w:pは段落を表し、通常、w:pは一つのWord文章に段落の数だけ表れる。なお、Wordでは、改行記号又は文頭から次の改行記号までが一つの段落である。段落の体裁（中央揃えなど）はw:pPr（paragraph propertyエレメント）の属性値として指定される。その有効範囲は、その段落内である。また、装飾付き文字列を表すのはw:r（runエレメント）である。その中にはw:tというエレメントがあり、ここに文字列が格納される。文字列の装飾は、w:rPrというエレメントの属性値で指定される。当然ながら、w:rは一つの段落内に複数存在しうる。また、一つのファイルには一つ以上の節があり、例えばページ設定などの節の設定に関する情報はw:sectPrに格納される（上では内容を略している）。これらのエレメントには開始タグと終了タグがあり、例えば、<w:body>と</w:body>の間に<w:p> 〜 </w:p>などの子エレメントが記述される。SampleQ.docxとSampleA.docxの二つの木構造を比較するとWord文書の修飾がXMLにどの様に反映されるかが分かる。

HITsサーバにSampleQ.docxが保存されていて学生はこれをダウンロードし上に述べた幾つかの修飾を施してサーバにアップロードすることを求められているとする。採点プログラムwt2013.pyはその内容を採点する。

1．第１段落全体を右揃えにする
```
<w:jc w:val="right" />
```
2．第２段落について
（a）フォントサイズを12ptにする
```
<w:sz w:val="24" />
```
（b）全体を太字にする
```
<w:b />
```
（c）全体に下線を引く
```
<w:u w:val="single" />
```
（d）全体を中央揃えとする
```
<w:jc w:val="center" />
```

4 XMLファイルの解析

前節の例に示す様に採点プログラムとは大まかに言って提出されたXMLファイルが然るべきノードを持つことを判別するプログラムである。ここでノードには、エレメントノード、アトリビュートノード、テキストノードなどの様々な型がある。上掲の例で、エレメ

ントノードは「w:document」、「w:body」など、テキストノードは「平成25年○月○日」など、アトリビュートノードは「[@w:val="right"]」などである。

wt2013.pyはライブラリー、lxmlを利用してこの作業を行う。XML Path Languageと呼ばれる規格（文法規則）に適合した式（xpath式）をlxmlに与えるとXML文書の中の特定のノードオブジェクトを取得できる。例えばSampleA.docxの第１段落を右揃えに修飾するアトリビュートノードを指示するxpath式は

```
/w:document[1]/w:body[1]/w:p[1]/w:pPr[1]\\
  /w:jc[1][@w:val="right"]
```

である。（なお\\は、紙面の幅に収めるための見かけ上の改行を表し、実際には改行が存在しないものとする。以降も同様。）このxpath式は木の根w:documentを起点として次の枝w:bodyを指し更にw:p → w:pPr → w:jcと辿って最後にアトリビュートノードw:val="right"を指している。XML Path Languageは獲得したノードオブジェクトの上の階層、下の階層、同じ階層の中にある別のノードオブジェクトを取得する書式を定めるなど豊かな表現力を持っている。

上の例で番号の表記[1]は１番目（同じ階層に属する同じ名前のノードの１番）を表す。従って

```
/w:document[1]/w:body[1]/w:p[2]
```

は第２の段落を示す。番号を書かないと「全て」を意味する。例えば上の式でw:p[2]をw:pに変えると

```
/w:document[1]/w:body[1]/w:p
```

lxmlは全ての段落オブジェクト（のリスト）を取得する。

与えられた.xmlファイルが持つ（根を起点とする）全てのxpath式の集合を考えよう。上の例で最も簡単なxpath式は/w:documentでこれは一つしかない。次に簡単なのは/w:document/w:bodyでこれも一つである。次は段落を示すxpath式で二つある。/w:document/w:body/w:p[1]及び/w:document/w:body/w:p[2] である。この様にして.xmlファイルが持つ全てのxpath式の集合を考えるとこれはオリジナルの木構造と等価である（xpath 式の集合はオリジナルの木構造を再現する）。

前節の問題に立ち返ると、ここでは第１段落に１問、第２段落に４問、合計５問が提示されていた。wt2013.pyは提出された.docxファイルとxpath式その他の情報を記述したファイルsample.ctl（コントロールファイルと呼ぶ）を読み込む。ここでsample.ctlは以下の内容を持つテキストファイルである。

```
open_file = word/document.xml

/w:document[1]/w:body[1]/w:p[1]/w:pPr[1]\\
  /w:jc[1][@w:val="right"]
平成25年○月○日
文字揃え

/w:document[1]/w:body[1]/w:p[2]/w:pPr[1]\\
  /w:jc[1][@w:val="center"]
新工場開設のお知らせ
文字揃え

/w:document[1]/w:body[1]/w:p[2]/w:r[1]\\
  /w:rPr[1]/w:b[1]
新工場開設のお知らせ
太字

/w:document[1]/w:body[1]/w:p[2]/w:r[1]\\
  /w:rPr[1]/w:sz[1][@w:val="24"]
新工場開設のお知らせ
フォントサイズ
/w:document[1]/w:body[1]/w:p[2]/w:r[1]\\
  /w:rPr[1]/w:u[1][@w:val="single"]
```

新工場開設のお知らせ
一重下線

open_file = word/document.xmlは、提出された.docxのファイル群の内のword/document.xmlを検査することを指示している。それ以降の部分は、三行で一まとまりとなっていて、最初の行はxpath式、二行目は修飾を加えるべき文字列、三行目は判定結果を伝える時のコメントである。このコントロールファイルは上から順に右揃え、中央揃え、太字、12ptのフォント、一重下線の合計５問の採点をしようとしている。学生が課題を正確に遂行した場合の出力は以下の様になる。

正解 第１段落 平成25年○月○日 文字揃え
正解 第２段落 新工場開設のお知らせ 文字揃え
正解 第２段落 新工場開設のお知らせ 太字
正解 第２段落 新工場開設のお知らせ フォントサイズ
正解 第２段落 新工場開設のお知らせ 一重下線

何かのミスがあれば「正解」の代わりに「不正解」が表示される。この出力は１行が１問に対応し、各行は空白で区切られた４つの文字列から成る。左から順に

１．正解、不正解の別
２．対応する段落番号
３．修飾すべき文字列（コントローファイルの３行組みのうちの２行目）
４．コメント（３行目）

である。

5 問題作成の手順

上の節から分かる様にHITsのサーバーには

１．問題ファイル
２．コントロールファイル

の二つが保存されている。学生は問題ファイルをダウンロードし指示に従って編集してアップロードする。wt2013.pyはアップロードされたファイルをコントロールファイルの指示に従って解析し結果をWeb上に表示すると同時に関連情報をデータベースに収録する。

この節で問題ファイルとそれに対応するコントロールファイルをどの様に作成するかを述べる。

5.1 問題ファイルの作成

通常、問題ファイルの書き出しにこのファイルをどの様に編集すべきかを書き、その先に編集すべき文字列を並べる。問題ファイルを制限する特別な規則はないが一つ注意することがある。それは学生には問題ファイルへの加筆、訂正、削除を原則として許していないことである。（例外として、tab挿入、表作成、脚注挿入、文末注挿入等の問題があり、これらの場合は加筆を許す。）即ち編集は文字列の修飾に限定される。例えば先の例で第１段落の「平成25年○月○日」を勝手に「平成25年５月３日」等と訂正するとwt2013.pyは不正解を出力する

5.2 コントロールファイルの作成

コントロールファイルを作成する最初の手順は模範解答ファイル（sampleA.docx）を作ることである。このファイルのxpath式の集合には判定すべきxpath式が含まれる。従ってその中から第１段落の右揃えを反映した

```
/w:document[1]/w:body[1]/w:p[1]/w:pPr[1]\\
  /w:jc[1][@w:val="right"]
```

を見つけて１）照合する文字列「平成25年○

月〇日」と２）コメント「文字揃え」を付け加えればコントロールファイルができる。

.xmlファイルが与えられた時、それが含むxpath式を全て出力するプログラムを書くことは易しいのでそこから問題に対応するxpath式を選び出せば良い。しかし小さな.xmlファイルと言えども多数のxpath式を出力するのでこれは厄介な作業である。

だが仮にSampleQ.docxのほんの一部を修正したものがSampleA.docxだったとする。夫々のxpath式の集合をSetQ及びSetAとする。SetAはSetQと殆ど同じだが修正によって発生した“SetQには存在しない”少数のxpath式を含むだろう。これがコントロールファイルを形成するxpath式である。即ちSetAからSetQを引き去ることで候補となるxpath式の数をかなり減らすことができる。この発想に基づいたPythonプログラムfilter.pyは自動的にコントロールファイルを作成する。即ち

```
./filter.py sampleA.docx sampleQ.docx
```

は先に例示したsample.ctlを出力する。残念ながらfilter.pyはそれほど賢くないので取り上げる価値のない採点課題も同時に出力する。それらは問題作成者が見て削除しなければならない。

6 テキスト照合

6.1 段落問題のテキスト照合

先の例、「平成25年〇月〇日」を右揃えする問題に戻ろう。wt2013.pyは提出されたファイルのword/docunet.xmlに対してxpath式

```
/w:document[1]/w:body[1]/w:p[1]/w:pPr[1]\\
  /w:jc[1][@w:val="right"]
```

を調べる。だがこのノードの存在は直ちに正解を意味しない、と言うのはこれが「平成25年〇月〇日」を修飾していることを確認する作業（テキスト照合）が残っているからである。w:pPrから分かる様に右揃えは段落全体に加えられており、修飾を受ける文字列は、たとえw:p内が複数のrunエレメントに分離されて、それぞれにテキストノードがあったとしても、w:p内に存在する全てのテキストノードの文字列を順に取得し、連結すれば得られる。それが「平成25年〇月〇日」と一致することを調べれば良いので段落問題では何も難しいことは起きない。

6.2 run問題のテキスト照合

しかし段落内の文字列の一部を修飾する課題では問題が起きる。例として「いろはにほへとちりぬるを」を問題文としその前半「いろはにほへと」を太字にし、後半に一重下線を引けと言う課題を考えよう。次の様なコントロールファイルが想定されるだろう。

```
open_file = word/document.xml

/w:document[1]/w:body[1]/w:p[1]/w:r[1]\\
  /w:rPr[1]/w:b
いろはにほへと
太字にする

/w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:rPr[1]/w:u[1][@w:val="single"]
ちりぬるを
一重下線を引く
```

ここでは

1．正解ファイルに二つのrunエレメントが発生し
2．w:r[1]が太字で修飾された「いろはにほへと」に対応し、
3．w:r[2]が一重下線で修飾された「ちりぬるを」に対応する

ことを期待している。しかし実際は、テキストの内容（文章自動校正で誤りと判定される文章等）、編集履歴などによって、三つ以上のrunエレメントが発生することがある。

```
w:r[1]
```

```
  w:rsidRPr="00744A37"
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:b
  w:t
    いろはに
w:proofErr
  w:type="gramStart"
w:r[2]
  w:rsidRPr="00744A37"
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:b
  w:t
    ほへ
w:proofErr
  w:type="gramEnd"
w:r[3]
  w:rsidRPr="00744A37"
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:b
  w:t
    と
w:r[4]
  w:rsidRPr="00744A37"
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:u
      w:val=single
  w:t
    ちりぬる
w:proofErr
  w:type="gramStart"
w:r[5]
  w:rsidRPr="00744A37"
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:u
      w:val=single
  w:t
    を
w:bookmarkStart
  w:id="0"
  w:name="_GoBack"
w:bookmarkEnd
  w:id="0"
w:proofErr
  w:type="gramEnd"
```

第１問に対応して連続する３個のw:rが作られ文字列「いろはに」「ほへ」「と」に同じ修飾

```
  w:rPr
    w:rFonts
      w:hint=eastAsia
    w:b
```

を与える。第１問のxpath式

```
/w:document[1]/w:body[1]/w:p[1]/w:r[1]\\
  /w:rPr[1]/w:b
```

は文字列「いろはに」を修飾するので照合文字列「いろはにほへと」と一致せず不正解となるだろう。

w:uが初めて現れるのはw:r[4]であるから第２問のxpath式

```
/w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:rPr[1]/w:u[1][@w:val="single"]
```

は発見に失敗してこれまた不正解となる。

ここで見る様に、連続する複数のw:rができてそれらが同じ修飾を与える現象をテキスト分断と呼ぶ。Wordのオプションで「組み込みの精度を向上させるためランダムな番号を保存する（T)」が設定されている場合、「自動文章校正（M)」が設定されている場合などで起こる。同じ修飾というのは、必ずしも

w:r内のテキストノード以外がすべて同じということを意味しない。採点の都合上、下位ノードにいくつか相違があっても（たとえば、w:spacing、w:rsidRPrなど）、それらの違いを無視して同じ修飾と見做す。テキスト分断には今までのところ統一的な規則性を見いだせていない。同一修飾を受けるテキストが幾つに分断されるか、どこで分断されるかは、テキストの内容や加えた操作の種類、順番にも依存する可能性があるため、予測不能であるといってよい。

テキスト分断を受け入れてコントロールファイルを柔軟に適用し正確に採点するにはどうしたら良いであろうか。w:rの番号は信頼できないのであるからwt2013.pyの（runエレメントにかかわる）テキスト照合は次の手順を採用している。

1．w:rの番号は無視し番号なしの扱いとする。
2．一つの段落内で同一の修飾を与える連続したw:rの格納する文字列は繋ぐ。

今の例では

```
/w:document[1]/w:body[1]/w:p[1]/w:r[1]\\
  /w:rPr[1]/w:b
```

に対応して文字列「いろはにほへと」が得られ、

```
/w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:rPr[1]/w:u[1][@w:val="single"]
```

に対応して「ちりぬるを」が得られテキスト分断が克服される。

しかし完全に解決した訳ではない。「いろはにほへとちりぬるをわかよたれそ」を問題文としその前半「いろはにほへと」を太字にし、「ちりぬるを」に一重下線を引き（ここまでは前問と同じ）「わかよたれそ」を再び太字とせよと言う課題を考えよう。この時xpath式

```
/w:document[1]/w:body[1]/w:p[1]/w:r[1]\\
  /w:rPr[1]/w:b
```

は二つの文字列「いろはにほへと」と「わかよたれそ」を生成する。従って前の規則に加えて

- 繋いで得られる文字列のどれかと照合文字列が一致すればテキスト照合が成功したと見做す。

としなければならない。

6.3 修飾する文字列が無い場合のテキスト照合

ページサイズの設定、マージンの設定等の課題を考えて見よう。対応するxpath式は如何なる文字列をも修飾しない。即ち照合文字列は空文字列である。当然コントロールファイル中に空文字列を表現する方法が必要になるので便法として特殊文字列「テキストなし」を採用して空文字列を表すことにしている。wt2013.pyは照合文字列の「テキストなし」を読むと内部的にこれを空文字列に置き換えて処理を進める。

6.4 テキスト照合を無視したい場合

コントロールファイル中のxpath式が修飾する文字列が、編集操作の手順によって変化する場合がある。典型的なのはtab文字の挿入で、例えば「いろはにほへ」という文字列の「いろはに」と「ほへ」の間にtab文字を入れた場合、Wordの画面上では全く違わないが、ツリーとして

```
w:r[1]
  w:t
    いろはに
w:r[2]
```

```
    w:t
      ほへ
    w:tab
```

となる場合や

```
  w:r[1]
    w:t
      いろはに
  w:r[2]
    w:tab
  w:r[3]
    w:t
      ほへ
```

となる場合がある。こうしたときに採点項目を

```
/w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:tab[1]
ほへ
タブの挿入
```

と設定すると、上の場合では正解となるが、下の場合では誤って不正解となってしまう。それを避けるために、特殊文字列「テキスト無視」を用意している。この例では、採点項目を

```
/w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:tab[1]
テキスト無視
タブの挿入
```

と書く。こうすると文字列照合が省略されるため、いずれの場合も正しく正解と判定することができる。ただ、「テキスト無視」導入の悪しき副作用としてどこにtab文字の挿入したかを採点できなくなるが、この困難はグループ化オプション（次章）によって解決する。

7 課題のグループ化

ある段落の文字列全体を紫色にする課題を考えよう。以下はコントロールファイルである。

```
/w:document[1]/w:body[1]/w:p[5]/w:pPr[1]\\
  /w:rPr[1]/w:color[1][@w:val="7030A0"]
この行全体を紫色にしなさい。
フォントの色
```

w:colorエレメントのアトリビュートのw:valの値“7030A0”が紫色の指定である。実はこれはWord2003の場合であってWord2007以降では“800080”としなくてはならない。即ちこの場合のコントロールファイルは

```
/w:document[1]/w:body[1]/w:p[5]/w:pPr[1]\\
 /w:rPr[1]/w:color[1][@w:val="800080"]
この行全体を紫色にしなさい。
フォントの色
```

である。学生の使っているWordのヴァージョンに応じてコントロールファイルを変更するのは厄介な作業である。できれば二つの採点項目を一つの問題と見做しどれか一問が正解なら正解と判定したいものである。wt2013.pyはこのためにキーワードORを用意している。

```
OR /w:document[1]/w:body[1]/w:p[5]/w:pPr[1]\\
  /w:rPr[1]/w:color[1][@w:val="7030A0"]
この行全体を紫色にしなさい。
フォントの色

OR /w:document[1]/w:body[1]/w:p[5]/w:pPr[1]\\
  /w:rPr[1]/w:color[1][@w:val="800080"]
この行全体を紫色にしなさい。
フォントの色
```

wt2013.pyは最初の採点項目を採点して正解

なら次の採点項目を無視する。不正解なら最初の採点項目（が不正解であったこと）を無視して次の採点項目を採点する。要するにプログラム言語のORの取り扱いに倣う。
２以上の採点項目のグループ化もできる。次のコントロールファイルがあるとする。

```
OR 採点項目 1
OR 採点項目 2
OR 採点項目 3
OR 採点項目 4
```

４つが全て不正解の場合は採点項目4が不正解であると表示される。採点項目1と採点項目2が不正解で採点項目3が正解の場合は採点項目3が正解と表示されて採点項目4は無視される…以下同様。そしてキーワードANDも使用できる。

```
AND 採点項目 1
AND 採点項目 2
AND 採点項目 3
AND 採点項目 4
```

では４つの採点項目が全て正解の場合に採点項目4が正解であると表示される。採点項目1と採点項目2が正解で採点項目3が不正解の場合は採点項目3が不正解と表示されて採点項目4は無視される。

前の章で書き残した「テキスト無視」の困難の解決は

```
AND /w:document[1]/w:body[1]/w:p[1]/w:r[2]\\
  /w:tab[1]
```
テキスト無視
タブの挿入
```
AND /w:document[1]/w:body[1]/w:p[1]/w:r[2]
```
ほへ
タブの挿入

である。第１の項目でタブの挿入を確認し、第２で文字列「ほへ」を確認している。

キーワードSEPARATEはグループ化を分離する。

```
AND 採点項目 1
AND 採点項目 2
SEPARATE
AND 採点項目 3
AND 採点項目 4
```

では採点項目1と採点項目2が一つのグループ、採点項目3と採点項目4が今一つのグループである。

キーワードNOTは正解、不正解を反転させる。

```
NOT 採点項目 1
```

は採点項目1が正解の場合に不正解と判定され、不正解の場合に正解と判定される。

```
NOT NOT 採点項目 1
```

も許されて採点項目1が正解の場合に正解と判定され、不正解の場合に不正解と判定される。NOTは幾つ並べても良く、その都度正解・不正解を反転させる。NOTの使用例を一つ示す。

```
AND /w:document[1]/w:body[1]/w:p[15]\\
  /w:pPr[1]/w:tabs[1]/w:tab[1]
```
講座名文章技術速習法
タブ設定

```
AND NOT /w:document[1]/w:body[1]/w:p[15]\\
  /w:pPr[1]/w:tabs[1]/w:tab[2]
```
講座名文章技術速習法
タブストップ数(1)

これは「講座名文章技術速習法」の「講座名」の直後にタブを一つ入れて整形する問題である。ところが時に複数のタブを入れて見かけ上同じ整形を実現する解答例がある。２つ目のxpath式はそこを咎めて第２のタブw:tab[2]が存在しないことをチェックしている。

これらのキーワードは任意の深さの入れ子にすることができる。

```
OR 採点項目 1
OR 採点項目 2
OR 採点項目 3
OR AND 採点項目 4
OR AND 採点項目 5
OR AND 採点項目 6
```

では最初に採点項目4採点項目5採点項目6がキーワードANDによってグループ化されその最終評価が採点項目1採点項目2採点項目3と共にORによってグループ化される。実は今の例ではキーワードNONによって自動的に

```
OR NON 採点項目 1
OR NON 採点項目 2
OR NON 採点項目 3
OR AND 採点項目 4
OR AND 採点項目 5
OR AND 採点項目 6
```

の様に整形される。NONは実効を持たないplace holderである。上のコントロールファイルは最初に第２列が評価され次に第１列が評価される（深い入れ子も同様で常にNONで整形され後ろの列から順に前の列に評価が進む)。つまりまず

```
NON 採点項目 1
NON 採点項目 2
NON 採点項目 3
AND 採点項目 4
AND 採点項目 5
AND 採点項目 6
```

が評価される。この時NONが付いた採点項目1採点項目2採点項目3は未だ評価されない。しかし採点項目4採点項目5採点項目6はANDでグループ化されて評価される。例えば採点項目4は正解、採点項目5が不正解であったとすると上は

```
採点項目 1 未評価
採点項目 2 未評価
採点項目 3 未評価
採点項目 4 評価済（正解）で無用となる
採点項目 5 不正解
採点項目 6 未評価で無用となる
```

と変わり第１列の評価

```
OR 採点項目 1 未評価
OR 採点項目 2 未評価
OR 採点項目 3 未評価
OR 採点項目 5 不正解
```

を受けることになる。即ち採点項目1採点項目2採点項目3の全てが不正解なら採点項目5が不正解と表示され、そうでなければ採点項目1採点項目2採点項目3の順で最初に正解になった採点項目が正解と表示される。

上記の例ではNONは自動的に補われるが、次は採点項目1と採点項目2のNONを省略できない場合である。（採点項目3と採点項目4のNONは省略できる）

```
AND NON OR 採点項目 1
AND NON OR 採点項目 2
AND OR NON 採点項目 3
AND OR NON 採点項目 4
```

このグループ化は採点項目1採点項目2の少なくとも一方が正解で且つ採点項目3採点項目4の少なくとも一方が正解である場合に正解、それ以外は不正解と判定する。

ORとANDは採点項目のグループ化をしたがNOTにはその機能はない：単に現在の採点項目の正解・不正解を反転させるだけである。NONも同様でこれは単なるplace holderである。なをこれらのキーワードは大文字小文字を区別しない。AND And andは全て同じである。

8 エバリュエイトオプション(>>)

8.1 変数の範囲を指定する

例として左インデントを設定する課題を考える。

```
/w:document[1]/w:body[1]/w:p[11]/w:pPr[1]\\
  /w:ind[1][@w:left="850"]
```

このxpath式は左インデントのサイズを“850”に設定することを求めている。しかし、この問題では、左インデントの設定ができているかどうかを採点したいので、そのサイズは0より大きければ良いことにしたい。このためにwt2013.pyは次の書式を許している。

```
/w:document[1]/w:body[1]/w:p[11]/w:pPr[1]\\
  /w:ind[1][@w:left="850"] >> 0 < w:left
```

殆ど説明を要しないと思うが、アトリビュートノードの後ろに>>を書き、その後ろにアトリビュートノードの変数w:leftについての式を書く。wt2013.pyはアトリビュートノードを除いたxpath式

```
/w:document[1]/w:body[1]/w:p[11]/w:pPr[1]\\
  /w:ind[1]
```

を調べてアトリビュートノードの変数w:leftの値を取得し、次に式“0 < w:left”を評価してそれに従って正解･不正解の判定をする。又若し

```
/w:document[1]/w:body[1]/w:p[11]/w:pPr[1]\\
  /w:ind[1][@w:left="850"]\\
  >> 800 < w:left < 900
```

と書けば左インデントが800より大きく900より小さいことを判定する。

8.2 keepとcomp

エバリュエイトオプション（>>）は組み込み関数eval()を利用するのである意味で万能である。ここで二つの段落に同じインデントを与えて文書の見栄えを整えよという課題を考えて見よう。その場合には異なるxpath式の属性値が同じか否かを点検しなければならない。以下の例は第６段落と第７段落のアトリビュートノードの変数w:LeftCharsが同じかどうかを調べている。

```
AND /w:document[1]/w:body[1]/w:p[6]\\
  /w:pPr[1]/w:ind[1][@
w:leftChars="2497"]\\
  >> keep(w:leftChars)
AND /w:document[1]/w:body[1]/w:p[7]\\
  /w:pPr[1]/w:ind[1][@
w:leftChars="2497"]\\
  >> comp(w:leftChars)
```

keep()とcomp()はwt2013.pyが持つ関数である。keep(x)は変数xの値を記憶してTrueを返す。今の例ではxは第６段落のインデントの量である（必ずしも“2497”とは限らない）。comp(x)は変数x（第７段落のインデントの量）を先に記憶した量と比較して一致すればTrueを返し、不一致ならFalseを返す。

9 まとめ

本稿では、HITsシステムの一部で、Micosoft Wordファイルの自動採点プログラムwt.pyの2013年度版wt2013.pyについて説明した。wt.pyは毎年改良を加え、採点の精度が向上してきているが、このバージョンにおいてもまだ完全ではない。その不完全さの最も大きな要因は、run問題のテキスト照合である。段落単位の問題、表のセル単位の問題等であれば、採点箇所を特定することは難しくないが、run問題即ち文字列単位の問題の場合は、採点箇所の特定が難しい場合が存在する。これは、Microsoft Wordを採点対象とした場合の根本的な困難さと言えよう。

また、他の改善すべき点としては採点結果の表示の問題がある。wt.pyの採点結果の出力も徐々に改善を施しているが、現行のwt2013.pyでもまだ不十分で、学生からどこがどのように間違えているのかわかりにくいとの指摘を受けている。

今後も引き続き改良を加え、より精度の高い採点が可能なプログラム、より学習効果が期待できるプログラムを目指して開発していきたいと考えている。

参考文献

[1] Standard ECMA-376 Office Open XML File Formats, http://www.ecma-international.org/publications/standards/Ecma-376.htm
[2] XML Path Language バージョン1.0 W3C勧告, http://www.w3.org/TR/1999/REC-xpath-19991116
[3] Office 2003 XML リファレンススキーマ, http://www.microsoft.com/japan/office/previous/2003/xml/default.mspx