HITsにおけるWord文書の採点プログラム2018年度版の開発

松井吉光
谷口正明

キーワード：e-Learning，教材，自動採点，Word，Python

要　　旨：本稿では，本学で独自開発したe-learningシステムであるHITsのWord部門における，文書解析・採点プログラムwt2018.pyの現行バージョンであるwt2016.pyからの変更点について述べる。

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。「情報リテラシー・入門，応用」では，主に本学で独自開発したe-learningシステムである，HITs (Highly Interactive Training system) を教材として利用している。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，Typing及びMicrosoft Ⓡ Office Ⓡ のWord Ⓡ とExcel Ⓡを対象としている。

wt.pyはHITsのWord部門の文書解析・採点エンジンでword_test.pyの意味である。拡張子の.pyで分かる通りpythonで記述されている。wt.pyはxml形式で記述されzipで集約・圧縮されたWord文書 (.docxファイル) と，その文書の採点について記述されたコントロールファイルを読み込んで採点結果を出力する。wt.pyはフロントエンドのwt＊＊＊.pyとwtmain＊＊＊＊.py，tool＊＊＊＊.pyの三つのファイルからなるプログラム群であるが，以下では総称してwt＊＊＊＊.pyと表記する。なお，＊＊＊＊には開発年度を表す，バージョン情報が入る。

本稿では現在運用しているwt2016.py [7] における採点の問題点と，それに対するwt2018.pyの変更点について述べる。

2　wt2016.pyの採点の問題点

まずはじめに，docxファイルの段落の文字列がどのように装飾されているかを述べる。Wordは段落別に管理し，さらにその段落の文字列をいくつかの部分 (run要素) に切り分ける。

$$paraText = subText_1 + subText_2 + \cdots + subText_n$$

m番目の部分文字列$subText_m$中の全ての文字はrun要素に共通の装飾$Deco_m$を受ける。

即ち docx ファイルの内部表現において各段落 paraDocx は順序和

$$\mathrm{paraDocx} = \sum_{m=1}^{n} \mathrm{subText}_m \times \mathrm{Deco}_m$$

となっている。ここで $\mathrm{subText}_m$ は具体的には /w:r[m]/w:t/ に格納された文字列であり Deco_m は w:r[m]/w:rPr 以下の樹構造である。右辺の各項を run 要素或いは番号を付けて第 m run 要素と呼ぶことにする。HITs の問題でこうした構造を考慮しなければならなくなるのは，段落のある部分を装飾する型の問題である。以降，この型の問題を run 型の問題と呼ぶ。

wt2016.py では run 型の問題の場合，Deco_m を調べ，隣り合う run 要素が共通する場合，$\mathrm{subText}_m$ を結合して新たに $\mathrm{logText}_i$ を作る。最終的に，paraDocx は

$$\mathrm{paraDocx} = \sum_{i=1}^{k} \mathrm{logText}_i \times \mathrm{Deco}_i$$

となる。ここで i を論理番号とする。右辺の各項を論理セグメントあるいは番号をつけて第 i 論理セグメントと呼ぶことにする。

wt2016.py はこの処理の後，コントロールファイルにある採点対象となる修飾を Deco_i から探し，存在する場合に，$\mathrm{logText}_i$ がコントロールファイルにある照合用の文字列と一致するかを判定している。

この採点方法では，正解であっても過って不正解と誤判定してしまう場合がある。具体例を示そう。段落

> この問題は文字飾りの練習です。
> ワードでは様々な文字飾りができます。例えば，これは太字です。
> これは斜体です。これは下線付きです。

において，

(1) 全体のフォントの種類をＭＳ 明朝に変更する。

(2)「これは太字」の部分を太字にする。

という問題があったとする。両方が正解・不正解の場合，(1) のみが不正解の場合は正しく採点される。しかし，(2) のみ不正解の場合は，(1) が正解であっても過って不正解と判定される。誤って不正解となる事情は以下の通りである。

この問題のコントロールファイルは，

```
<prob prob_num="1" group_size="3">
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[1]/w:rPr[1]
      /w:rFonts[1][@w:eastAsia
                          =" ＭＳ 明朝 "]
    </xpath>
    <ctext> この問題は文字飾りの練習です。ワードでは様々な文字飾りができます。
    例えば、</ctext>
    <comment> フォントの種類
    </comment>
  </prob_unit>
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[2]/w:rPr[1]
      /w:rFonts[1][@w:eastAsia
                          =" ＭＳ 明朝 "]
    </xpath>
    <ctext> これは太字 </ctext>
    <comment> フォントの種類
    </comment>
  </prob_unit>
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1" >
      /w:document[1]/w:body[1]/w:p[1]
```

```
        /w:r[3]/w:rPr[1]
        /w:rFonts[1][@w:eastAsia
                              =" ＭＳ 明朝 "]
    </xpath>
    <ctext> です。これは斜体です。
    これは下線付きです。</ctext>
    <comment> フォントの種類
    </comment>
  </prob_unit>
</prob>
<prob prob_num="2" group_size="1">
  <prob_unit>
    <xpath xtype="run" para_num="1" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[2]/w:rPr[1]/w:b[1]
    </xpath>
    <ctext> これは太字 </ctext>
    <comment> 太字 </comment>
  </prob_unit>
</prob>
```

である。正解ファイルでは，

$logText_1$ = この問題は文字飾りの練習です。
　　ワードでは様々な文字飾りができます。例えば，
$logText_2$ = これは太字
$logText_3$ = です。これは斜体です。
　　これは下線付きです。

となるため，正しく採点される。しかし (2) が不正解の場合は，上記のように論理セグメントに分かれる可能性は極めて低い。例えば，

$logText_1$ = この問題は文字飾りの練習です。
　　ワードでは様々な文字飾りが
　　できます。例えば，
　　これは太字です。
　　これは斜体です。
　　これは下線付きです。

のように，第 1 論理セグメントのみになることが想定される。この場合，フォントが正しく「ＭＳ 明朝」に変更されていたとしても，論理セグメントの文字列と照合用の文字列が一致しないため，不正解と判定されてしまう。

そこで，wt2018.py では，この誤った判定を防ぐ改善を行った。次節では，その仕様について述べる。

3　wt2018.py の仕様

wt2018.py では run 型の問題の場合，採点対象となる修飾を $Deco_m$ 内に存在するかを調べ，存在する場合を Flag=1，存在しない場合を Flag=0 と設定する。そして m=1 から順番に調べ，Flag=1 または Flag =0 が連続する場合は，$subText_m$ を結合し，新たに $logText_i$ を作る。最終的に，paraDocx は

$$paraDocx = \sum_{i=1}^{k} logText_i \times Deco_i \times Flag_i$$

となる。前節同様，右辺の各項を第 i 論理セグメントと呼ぶことにする。

wt2018.py はこの処理の後，コントロールファイルにある論理番号 (log_run_num) に相当する論理セグメントが Flag=1 となっていること及び，照合用の文字列と $logText_i$ が一致するかを判定する。

具体的な採点例を示す。段落

　この問題は文字飾りの練習です。
　ワードでは様々な文字飾りができます。例えば，これは太字です。これは斜体です。これは下線付きです。また，これらを組み合わせて，太字で斜体で下線とすることもできます。

において，

(1) 全体のフォントの種類とサイズを変更する。
(2)「これは太字」の部分を太字に変更する。
(3)「太字で斜体で下線」の部分を太字かつ斜体字かつ下線付きに変更する。

という問題を採点する場合を考える。

まず，フォントの種類とサイズを変える問

題では，段落全体を変更するため，run がいくつかに分かれていたとしても正解の場合には必ず論理セグメントは 1 つになっている。したがって，この場合は

$$paraDocx = logText_1 \times Deco_1 \times Flag_1$$

となる。後は，$Flag_1$ が 1 であること，$logText_1$ が照合用の文字列が一致することを確認すれば採点ができたことになる。

次に「これは太字」の部分を太字に設定する問題を考える。正解の場合は，論理セグメントは，

$$paraDocx = \sum_{i=1}^{5} logText_i \times Deco_i \times Flag_i$$

となるはずである。ここで，

$logText_1$ = この問題は文字飾りの練習です。
　　ワードでは様々な文字飾りが
　　できます。例えば，
$logText_2$ = これは太字
$logText_3$ = です。これは斜体です。
　　これは下線付きです。
　　また，これらを組み合わせて，
$logText_4$ = 太字で斜体で下線
$logText_5$ = とすることもできます。

である。したがって採点は，第 2 論理セグメントの $Flag_2$ が 1 であることおよび，$logText_2$ が照合用の文字列「これは太字」と一致することを確かめれば良い。

では，「太字で斜体で下線」の部分を太字にする場合はどうであろうか。この場合も正解であれば，論理セグメントは上のようになる。ただし，この場合の採点は第 4 論理セグメントの $Flag_4$ が 1 であること，$logText_4$ が照合用の文字列「太字で斜体で下線」と一致することを確認することになる。

ただし wt2018.py の場合，「これが太字」が不正解で「太字で斜体で下線」が正解の場合では，「太字で斜体で下線」も誤って不正解と判定される可能性が高い。それは，太字というアトリビュートで論理セグメントに分けたとき「太字で斜体で下線」よりも前の部分が 3 つに分割されていない状態になっている可能性が高いことによる。このような場合には，「この箇所は正解の可能性があるが，同じ段落の別の箇所に同じ修飾に関する間違いがあります」というメッセージを付加することで対応することにしている。

以上が，wt2018.py の段落内の一部の文字列についての修飾の採点の基本的な流れであり，ほとんどの run 型の採点に支障なく適用できる。しかし，一部例外処理を要する場合があることが分かっている。それは，タブ (w:tab) やソフトブレーク (w:br) を追加する問題である。これらの要素は，独立してテキストを持たない run 要素の子要素になる場合と，タブまたはソフトブレークを挿入した位置の後のテキストを持つ run 要素の子要素になる場合がある。（Word2013, Word2016 ではほとんど前者になり，後者が出現する可能性は低い。ただし，可能性がゼロではないことが分かっている。）前者ではタブやソフトブレークの挿入箇所を特定できなくなるという問題が発生する。そのため，wt2018.py はこれらの要素を持つ場合，論理セグメントを構成するときに $Flag_n$ の値をそれぞれ 2 と 3 とし，他とは区別して扱う方法を採用した。

なぜ，特別扱いが必要になるのか具体例を示そう。段落

　　講座名文章技術速習法

において，「講座名」と「文章技術速習法」の間にタブを挿入する問題を考える。この場合の正解ファイルの run 要素のでき方は

$subText_1$ = 講座名
$subText_2$ =
$subText_3$ = 文章技術速習法

と

$subText_1$ = 講座名
$subText_2$ = 文章技術速習法

の 2 通りの可能性がある。後者の場合は，第 2run 要素の $subText_2$ に文字列があり，$Deco_2$ に w:tab 要素が含まれることになるため，wt2018.py で採点の際，w:tab 要素が含まれるかどうかで，論理セグメントを作成し，第 2 論理セグメントで $logText_2$ が「文章技術速習法」と一致するかと $Deco_2$ に w:tab 要素に含まれているかで採点ができる。しかし，前者の場合第 2 run 要素の $Deco_2$ に w:tab 要素はあるが，$subText_2$ に文字列がないため，採点箇所を特定する即ち，タブを挿入した箇所を特定することが単純な方法では不可能である。

そのため，コントロールファイルでは，

```
<prob prob_num="1" group_size="3">
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1"
    log_run_num="1" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[1]/w:rPr[1]
      /w:rFonts[1][@w:eastAsia
                    =" ＭＳ Ｐ 明朝 "]
    </xpath>
    <ctext> 講座名 </ctext>
    <comment> タブの挿入位置
    </comment>
  </prob_unit>
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1"
    log_run_num="2" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[2]/w:tab[1]
    </xpath>
    <ctext> テキストなし </ctext>
    <comment> タブの挿入位置
    </comment>
  </prob_unit>
  <prob_unit>
    <group_test>and</group_test>
    <xpath xtype="run" para_num="1"
    log_run_num="3" >
      /w:document[1]/w:body[1]/w:p[1]
      /w:r[3]/w:rPr[1]
      /w:rFonts[1][@w:eastAsia
                    =" ＭＳ Ｐ 明朝 "]
    </xpath>
    <ctext regu="1">^ 文章 </ctext>
    <mtext> 文章技術速習法 </mtext>
    <comment> タブの挿入位置
    </comment>
  </prob_unit>
</prob>
```

のように，タブの前後の文字列を別の要素（この場合はフォントの種類）を使って特定する必要が生じる。ただ，この場合第 2 run 要素の $Deco_2$ にも /w:rFonts[1][@w:eastAsia=" ＭＳ Ｐ 明朝 "] が含まれるため，タブの $Flag_n$ を他と区別しない方法では，最初の採点項目において論理セグメントは段落全体が 1 つにつながってしまい，不正解になるという問題が発生する。フォントの種類以外の要素を採点に利用としても結果は同じある可能性が高い。そのため，タブが含まれる run 要素と他と区別して取り扱う必要が生じるのである。wt2018.py で採用した，タブが含まれる論理セグメントの $Flag_n$ を 2 にする方法では，上記コントロールファイルの全ての採点項目 (prob_unit) において

$logText_1$ = 講座名
$logText_2$ =
$logText_3$ = 文章技術速習法

となるため，結果正しく採点できるようになる。

4　python3.x への移行

wt2018.py では，採点方法の変更ではないが，重要な変更を行った。これまでの wt.py は python2.x 系列で動作するよう記述してきた。しかし，そのサポートが 2020 年に打ち切られることが発表されているため [8]，余裕を持って python3.x 系列への移行を行うためには，早期に移行を実行する必要があった。そのため，今回 wt2018.py を開発するにあたり，python3.x 系列への移植作業と動作確認を行った。

python2.x から python3.x への変更点は多々あるが，文法の関する変更点は簡単な移植ツール (2to3) があり，それを実行するだけで簡単に変更できた。日本語の処理に関しても，python2 では文字列型としてバイト列型とユニコード文字列型が共存していたため，絶えず encode，decode の処理に悩まされたが，python は文字列はユニコード文字列型に統一されたため扱いが非常に楽になった。

ただ，python3 はコンピュータの LANG 環境変数を参照するため，コマンドラインでは問題なく日本語処理できる場合でも，HITs のシステム即ち，httpd から php を通して呼び出す場合には不具合が生じた。それはシステムの LANG 環境変数が標準では C(ascii) であるためで，日本語を処理を仕様とすると UnicodeEncodeError が出現して正常に動作しなかった。

そのため，ファイル内に

```
import io,sys
sys.stdout
  = io.TextIOWrapper(sys.stdout.buffer,
                encoding='utf-8')
sys.stderr
  = io.TextIOWrapper(sys.stderr.buffer,
                encoding='utf-8')
```

と標準出力と標準エラーへの出力を明示的に encode するとともファイルの呼び出しで

```
f=open("hoge.txt","a",encoding='utf-8')
```

のように，ファイルの文字コードを指定する必要があった。

以上のような変更を加えることで，wt2018.py が問題なく HITs でも動作することを確認した。

5　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 2018 年度版 wt2018.py の現行バージョンである wt2016.py からの変更点について述べた。

wt.py は毎年改良を加え，採点の精度の向上に努めるとともに，問題作成の簡単化を模索している。ただ，現時点でも問題作成，特に採点コントロールファイルの作成は Word の XML に精通し，Word の操作による変化についての知識が必要であり，とても誰でも作成できるレベルではないことは否めない。次のバージョンの wt.py では，より簡単にコントロールファイルを作成できるよう，現在開発・検討を進めている。

謝辞

HITs システムの開発者の一人である長谷部勝也氏には，本論文の研究につき，議論に加わっていただき貴重な助言をいただいた。深く感謝申し上げたい。

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
[4] "Excel, Word 自動採点システム HITs の構築と運用 "
岩田員典，他，
愛知大学情報メディアセンター (20)No.1(2010)
[5] "HITs における Word 文書の採点プログラムの開発 "
長谷部勝也，松井吉光，谷口正明
愛知大学一般教育論集 (40)25-40(2011)
[6] "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
愛知大学一般教育論集，(45)41-53(2013)
[7] "HITs における Word 文書の採点プログラム 2016 年度版の開発 "
松井吉光，長谷部勝也，谷口正明
愛知大学一般教育論集，(52)27-36(2017)
[8] pep-0373
https://hg.python.org/peps
/le/76d43e52d978/pep-0373.txt