TOM の改良

03H094 藤川 慎也

目次

1 はじめに 1

2 現在の TOM について 2
- 2.1 禁則チェック . . . 2
- 2.2 ワードラップチェック . . . 2
- 2.3 文体チェック . . . 2
- 2.4 類似警告 . . . 2
- 2.5 変更差分 . . . 2

3 理論と仮定 3
- 3.1 理論 . . . 3
- 3.2 仮定 . . . 3

4 WEB 検索 3

5 システムの手順 3

6 結論 5

7 あとがき 6

付録 A プログラムのソース 7
- A.1 段落で文章を分割する . . . 7
- A.2 メイルの文章をわかち書きにする . . . 7
- A.3 メイル文章から名詞のものを取り出す . . . 7
- A.4 Yahoo!検索 WEB サービス . . . 8
- A.5 検索結果から一つずづソースを抽出する . . . 9
- A.6 類似度を算出する . . . 9

1　はじめに

大学に入ると、一般的にレポート課題が増える。他の大学の友人たちも皆レポートのたびに四苦八苦しているようだが、世の中の学生達は実際に自分で考えた文を書いているのだろうかと疑問に感じた。

大垣斉講師の講義では、講義中に出席確認をするのではなく、講義後にメイルで講義内容を要約して提出するという出席確認レポートがある。その出席確認レポートの提出時など、メイルの提出時に違反がある場合に警告を出すシステムが TOM である。TOM には、禁則文字の使用をしている場合や、ワードラップ処理を行っていない場合、文体を統一していない場合、他人のメイル内容をコピーして用いている場合、文章を変更して提出した際に変更部分が少ない場合に警告を出すものがある。

現在の TOM には他人のメイル内容をコピーしている場合に警告を出すものは存在するが、ウェブサイトの内容をコピーしている場合に警告を出すものがない。今回の研究ではウェブサイトの内容をコピーしてメイルを書いている場合にも警告を出すシステムの構築を行う。

文章をコピーしていると、学生が自分で本当に考えている内容を書かなくなり、さらに個性や研究に対する意欲が薄れ、新たな意匠が生まれなくなる。ウェブサイトの文章をコピーし、そのままメイルを書いている場合に警告文を送信することができれば、学生がウェブサイトの文章をコピーすることが減少するだろうと考えられる。

2　現在の TOM について

現在 TOM が警告を出す違反の種類は、5 種類ある。

2.1　禁則チェック

行頭に捨て仮名や句読点、閉括弧などが使用されていたり、行末に開け括弧などが使用されているような禁則文字を使用している場合に警告を出す。禁則違反行が含まれるレポートは評価の対象にはならない。

2.2　ワードラップチェック

行を折り返すときに行末の 1 つの単語を途中で改行し、2 行に分割しないで単語全体を次の行に送るワードラップ処理を行わずに単語の途中で改行した場合に警告を出す。

2.3　文体チェック

文章の文末には、「だ・である」のような敬語を含んでいない常体、「です・ます・した」のような敬語を含んでいる敬体があるが、その書いている文章の文体が常体か敬体に統一されていない可能性がある場合に警告を出す。

2.4　類似警告

自分のメイル内容と他人のメイル内容と 60 パーセント以上類似している場合は、文章をコピーしてる可能性があるとして警告を出す。

2.5　変更差分

前回自分が提出したメイルと今回自分が再提出したメイルの変更差分をパーセントで出し、75 パーセント以上酷似している場合に警告を出す。

3　理論と仮定

3.1　理論

ウェブサイトの文章をコピーしているメイルを送信するということは、類似度が高く、TOM に警告文を送信されるということである。つまり、類似度が高くなればなるほど、パーセンテージが高くなる。

3.2　仮定

学生がウェブサイトの文章をコピーし、コピーした文章を含むメイルを送信する。すると、TOM がウェブサイトの文章とメイルの文章の類似を検知し、警告文を送信するはずである。つまり、類似度が高いということになる。

4　WEB 検索

代表的な検索サービスに GoogleWebAPI と Yahoo! API がある。Yahoo! JAPAN が提供する検索 Web サービスには、ウェブ検索、画像検索、動画検索、関連キーワードがある。これらを用いることによって検索などの機能を外部から利用できる。使用するにはどちらもアプリケーション ID を登録しなければ使用できない。今回の研究では Yahoo! API を使用している。なぜなら、どちらにも利用回数に対する制限があり、GoogleWebAPI では一人一日 1000 回まで、Yahoo! API では一人一日 5000 回までと決められている。そのため、一日の利用回数が不確定なので、Yahoo! API を選択している。

5　システムの手順

様々なウェブサイトから文章をコピーしているのを防ぐために、メイルの文章を段落ごとに分割し、その一つ一つについて類似度を出す。段落ごとに分割した文章を chasen を用いてわかち書き[*1]にする。(図 1)

そのわかち書きされた文章から名詞のものを取り出し、使用回数の多い順番に並べ、使用回数の多いものをキーワードとして抽出する。(図 2)

抽出されたキーワードを検索語として Yahoo!検索 WEB サービスを用いて検索をする。(図 3)

これにより検索件数が絞られ、さらにキーワードを使用している文書を取り出すことができる。検索結果から URL を取り出し、ウェブサイトの文章とメイルの文章を比較する。メイルの文章とウェブサイトの文章をわかち書きにする。わかち書きされた文章の区切りの空白に改行を入れ、文章を一列にする。そして、メイルの文章とウェブサイトの文章を区切られた語ごとに比較する。diff コマンドでメイルの文章とウェブサイトの文章の差分を出す。これで、文章が変更されている箇所が抽出される。全体のメイルの行数から文章の差分を引き、それを全体のメイルの行数で割り、パーセンテージに直すと類似度が算出される。

[*1] 文章の区切りに空白を挟んで記述すること

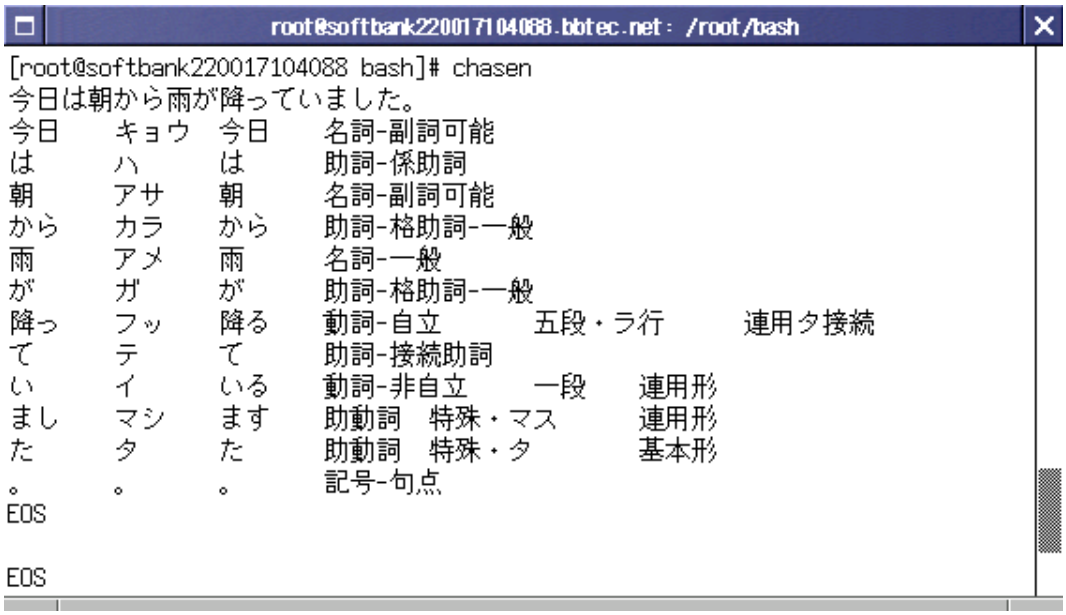

図 1 (例) わかち書きした結果

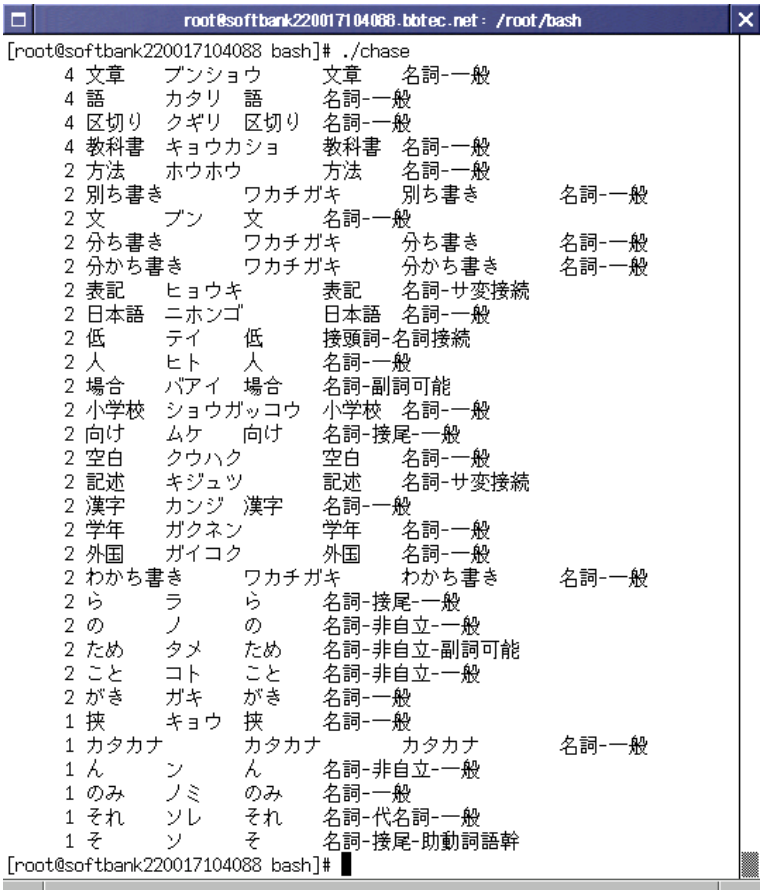

図 2 (例) 使用回数の多い順番

図 3　(例) 検索結果

6　結論

パーセンテージを出し、70 %以上と算出されると、類似度が高いことがわかる。つまりこれは、ウェブサイトの文章をコピーした可能性が高いということを示す。さらに段落の分割数毎にプログラムが実行されるので、他もウェブサイトの文章をコピーすると TOM に警告文が送信される。ゆえに仮定は立証された。

7　あとがき

研究をしている中で、メイルの文章とウェブサイトの文章が類似しているものをどのように取り出せるかという部分が困難に感じた。なぜならばウェブサイト全体とメイルの文章を一つ一つ比較するのは手間のかかる作業である。しかし、Yahoo! 検索 WEB サービスを利用することで比較する件数がある程度まで絞られた。メイルの文章とウェブサイトの文章の比較をしているところでは、現在の TOM の類似警告を参照にしている。

謝辞

本研究を行うにあたって、御指導および助言をいただいた藤井 信夫教授、大垣斉講師及び fken の方々に深く感謝の意を表します。

参考文献

[1] Yahoo!デベロッパーネットワーク, [http://developer.yahoo.co.jp/search/web/V1/webSearch.html]
[2] Perl で Yahoo!Web 検索サービス, [http://www.geekpage.jp/web/yahoo-developer/wss-perl-0.php]
[3] Perl CGI 最強講座, [http://www.rfs.jp/sb/perl/]

付録 A　プログラムのソース

A.1　段落で文章を分割する

```
#! /usr/bin/perl

open FILE,"qqq";
$no = 1;
while($line = <FILE>){

    if ( $line !~ /^$/ ){
$aaa[$zz] = "$line";

$zz++;

    }else{

open FH,">A$no";    作成されるファイル A1 ~
print FH @aaa;

$no++;
$zz=0;
close FH;
         }
 }
close (FILE);
```

A.2　メイルの文章をわかち書きにする

```
#! /bin/sh

cat BODY | \
sed -e '1,/^$/d' | sed -n -e '1,/^-- $/p' | \
grep -v "^>" | grep -v "^#" | nkf -Z1 | \
tr -s '\t' ' ' | tr -d '\n' | \
chasen -F"%m\n " >qqq
```

A.3　メイル文章から名詞のものを取り出す

```
chasen  BODY | grep ' 名詞-一般' | sort | uniq -c | sort -nr | \
tr -s '\t ' ' ' | cut -d " " -f 3 | head -n 4 | tr -s '\n' ' ' >ggg
```

A.4　Yahoo!検索 WEB サービス

```
#! /usr/bin/perl

use LWP::Simple;
use XML::Simple;
#binmode(STDOUT, ":utf8");

$appid = "simple-heartEAS9104";
$start = "1";
$results = "100";
$language = "ja";
$country = "jp";

open (FILE,"ggg");
$query = <FILE>;
close (FILE);

$url ="http://api.search.yahoo.co.jp/WebSearchService/V1/webSearch?appid=$appid&query=$query";


$url .= "&results=$results&start=$start";

 $response = get($url);

$xml = new XML::Simple;
$yahoo_xml = $xml->XMLin($response);

foreach $result (@{$yahoo_xml->{Result}}){

  $result_url = $result->{Url};

  print "$result_url\n";
}

exit;
```

A.5　検索結果から一つずづソースを抽出する

```
#! /bin/sh

count=1

DD=‘cat A.txt | wc -l‘
DO=‘expr $DD + 1‘

while [ $count -le $DO ];  A テキストの行数回ループ

do
tail +$count A.txt >B.txt
head -n 1 B.txt >C.txt
 cat ’C.txt’

  wget -O index.html -i C.txt     -O file    file という名前で保存する
                                  -i C.txt   C テキストの HTTP を GET する
  count=‘expr $count + 1‘
done
```

A.6　類似度を算出する

```
CO=‘cat a | wc -l‘
CC=‘diff -b -B a b | grep ">" | wc -l‘
Ruiji=‘expr 100 - $CC \* 100 / $CO - 1‘

echo $Ruiji
```