コピーレポート検知システムの開発

06H051 竹中康朝

目次

1 はじめに 1

1.1 目標 1

1.2 先行研究 1

1.3 既存のシステム 1

2 開発したシステム 2

2.1 開発環境 2

2.2 システムの説明 2

2.3 システムの動作 6

3 結果と考察 8

3.1 利点と問題点 9

4 まとめ 10

4.1 今後の課題 10

付録 A ソースコード 13

A.1 付録 1.1 14

概要

今日、インターネットの普及により情報の取得は容易になった。しかし容易に情報を仕入れることが可能になり、web 上からコピーした文章をそのまま載せたコピーレポートを提出する学生が増えてしまった。しかし、コピーレポートは人の目で判別することは大変な労力を要する。しかし放置すれば学生の質が低下するばかりか、学生に対して正当な評価を下せなくなってしまう。そこで、コピーレポートかどうかを自動検知するシステムの開発を行った。本研究では、メールの本文に直接記述する形式のレポートを対象とする。プログラムにより自動的に類似度を調べあげて、web 上からコピーペーストしたレポートの提出を拒否する。また、メールを受信したときに自動的にプログラムが起動するので、教員の負担を大幅に軽減できる。メールを受信すると、それをトリガーにプログラムが起動する。類似したシステムとして、大阪産業大学工学部情報システム工学科で稼働している TOM が存在する。TOM は類似度の比較対象が他の学生が提出したレポートである。しかしながら、アルゴリズムは TOM と似通っているので、今回開発したシステムは TOM の web 版と言える。実験の結果、判定をシステムが行うので、教員側への負担は小さいと結論づけた。しかし、類似度判定の精度が高くないので、今後の課題としては、類似度判定の精度の向上を図る。

1　はじめに

今日、インターネットの普及により情報の取得は容易になった。しかし容易に情報を仕入れることが可能になり、web 上からコピーした文章をそのまま載せたレポートを提出する学生が増えてしまった。コピーレポートは人の目で判別することは大変な労力を要する。しかし放置すれば学生に対して正当な評価を下せない。そこでコピーレポートかどうかを自動で検知するシステムの開発を行った。

1.1　目標

本システムの目標は、コピーレポートを検知してコピーレポートを成績に反映させないことである。また、現在稼働している TOM[*1]と併用して類似度判定の範囲を web まで広げ、検知率を向上させることである。

1.2　先行研究

先行研究として現在情報システム工学科の一部で稼働している TOM が存在している。

1.3　既存のシステム

- TOM: 大阪産業大学の大垣斉講師が開発したシステム。現在大阪産業大学の情報システム工学科で稼働中。他人の提出したレポートを対象に類似度判定を行うシステムである。 今回開発したシステムの参考にしたシステムはこの TOM である。

- 名称不明: 金沢工業大学の杉光一成教授が開発したシステム。名称は不明。これは当研究と同じく web からの情報から類似度を求めるシステムであるが、相違点は MS-Word[*2] を用いて作成された添付ファイルを判別する対象とすること。類似しているかどうかをまず人間が判断し、手動でシステムに渡すことである。

[*1] たぶん 同じ メール の略称
[*2] Microsoft Office Word

2　開発したシステム

メールを受信するとプログラムが起動する。まず、メール本体を分解して単語ごとに分割する。出てきた単語を使って、検索エンジンを使って検索する。検索結果の HTML から比較対象となる web ページの URI の HTML を取得する。取得した HTML とメールとを比較して類似度を求める。結果のパーセンテージを記述したメールを教員側に送信する。

2.1　開発環境

- ハードウェア (AT 互換機)
  - CPU:Intel(R) Celeron(R) CPU 2.80GHz
  - memory:512MB
- OS:Vine Linux 3.2
- 開発言語:シェルスクリプト (bash)

2.2　システムの説明

提出されたメールが web 上からコピーされたものでないかを調べる。システムに通された結果は、教員に対してメールで知らされる。システムの概略を図 **1** に示す。

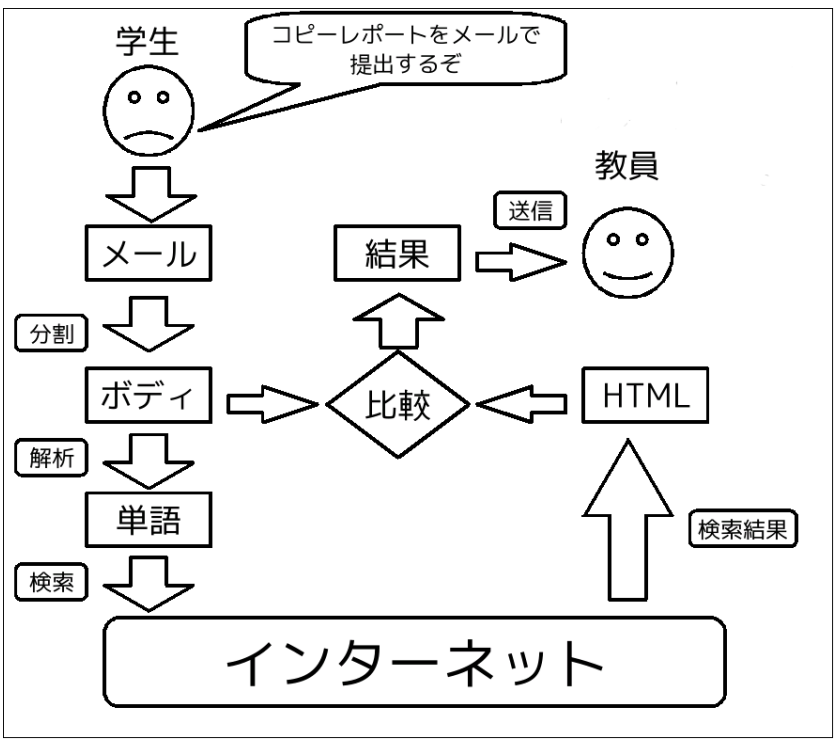


図 1　概略図

2.2.1 形態素解析

検索には、提出されたメールの本文を形態素解析[*3]にかけて抽出された単語を用いる。形態素解析には Kakasi を使用した。Kakasi とは、 日本語を平仮名やローマ字綴りの文に変換するプログラムである。単語ごとにわかち書きができる機能が存在する。Kakasi はフリーソフトウェアであり、GPL[*4] に基づいて配布されている。

今回使用した Kakasi の他にもいくつか形態素解析ソフトが存在する。

- Juman: 京都大学言語メディア研究室で開発された。隠れマルコフモデル[*5]を使った解析で高い精度を誇り、形態素解析ツール Chasen の元となったシステム。

- Chasen: 奈良先端科学技術大学院大学松本研究室で開発された。統計的な手法を用いており、解析速度と使い勝手の向上している。

- MeCab: 奈良先端科学技術大学院大学の工藤拓氏によって開発された。形態素解析ツール Chasen に比べて解析精度は同程度で、解析速度は平均 3-4 倍速い。

[*3] 自然言語で書かれた文を形態素の列に分割し品詞を見分ける作業

[*4] General Public License の略。コピーレフトのソフトウェアライセンスの一つ。許諾する内容は、 プログラムの実行、プログラムの動作を調べ、それを改変すること複製物の再頒布、プログラムを改良し、改良を公衆にリリースする権利

[*5] 確率モデルの一つである。「システムがパラメータ未知のマルコフ過程である」と仮定し、観測可能な情報からその未知のパラメータを推定する

2.2.2　形態素解析ソフトの比較

形態素解析ソフトの選定を行った。選定対象は Kakasi,Mecab,Chasen である。解析結果を図 **2** に示す。

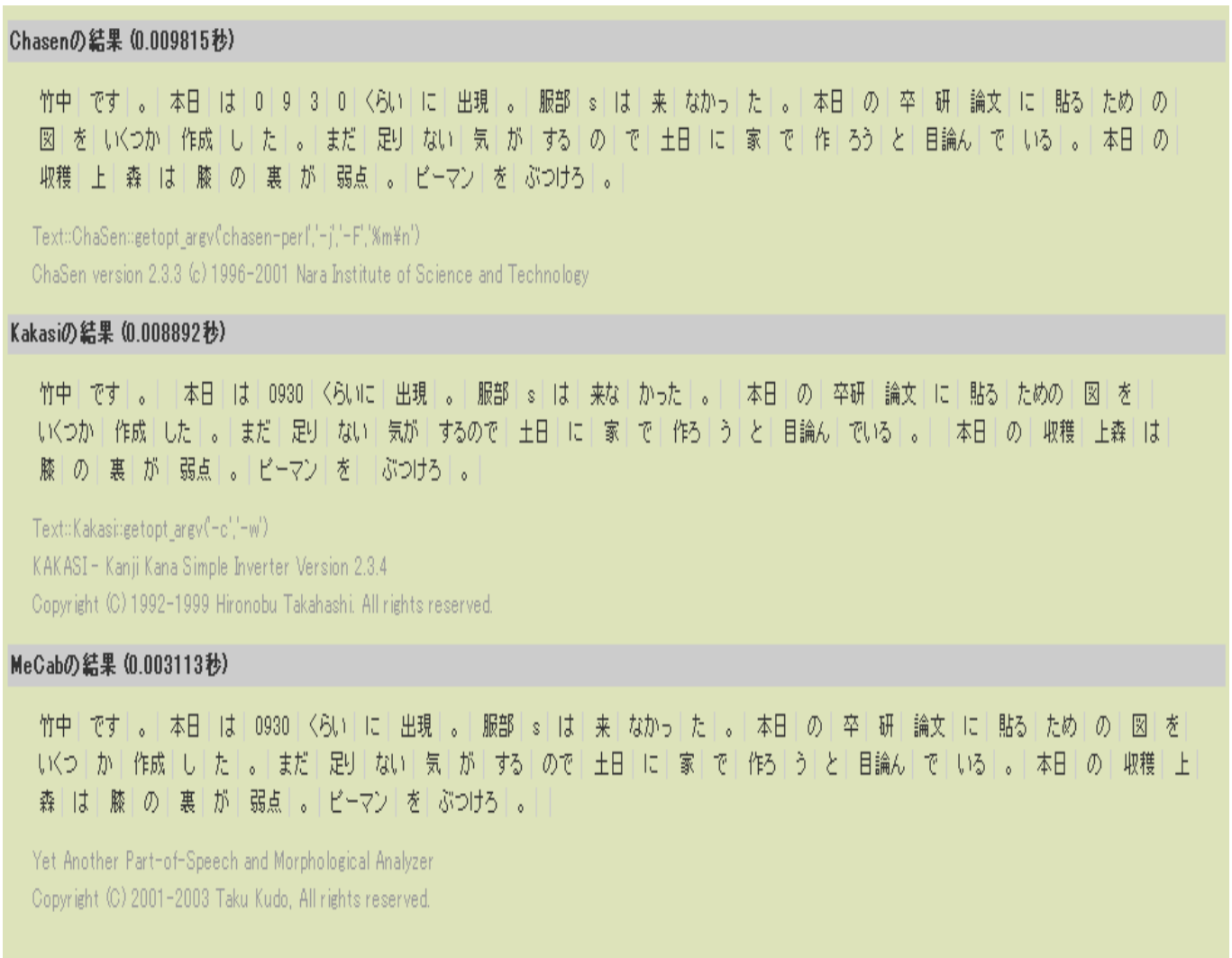


図 2　解析結果

解析結果より、MeCab と Chasen は必要以上に分割してしまうのでキーワード検索に用いる単語を抽出するには不向きである。よって本研究では Kakasi を採用した。

2.2.3　キーワードの抽出

形態素解析されて単語毎に分割された文章から接続詞や句読点を除外する。

2.2.4　テキストブラウザ

今回は使用するウェブブラウザとして w3m を使用した。w3m とは、現東北大学伊藤彰則教授によって開発されたページャ/テキストベースの WWW[*6] ブラウザ。kterm や rxvt などのターミナルエミュレータ[*7] を使って Web を閲覧できるほか、HTML をテキスト形式に整形するツールとして利用することもできる。今回 w3m を使

[*6] World Wide Web
[*7] 端末として動作するソフトウェア

用した理由は、w3m オプション指定で読み込んだ HTML 文書をプレーンテキスト[*8]に整形して出力する機能を備えるからである。

2.2.5 検索エンジン

本システムでは検索エンジンは yahoo を利用する。yahoo での検索結果の HTML から URI を取り出し、取り出した URI の HTML から提出されたメールと類似度を調べる。

他の検索エンジンには Google が存在するが、検索エンジンの選択肢には他に GoogleAPI[*9]の利用が挙げられたが、使用回数に制限があったため今回は回数制限のない yahoo を選択した。

2.2.6 URI の選別

判定を行っても効果が得られないと思われる web ページの URI は判定対象から除外する。除外対象は以下に示す。

- google
- amazon
- nicovideo
- youtube

これらを除外対象にした理由は、動画サイトやショッピングサイトは比較するために必要なテキストが少なく、また価格や広告などが多く存在するために比較対象としては不適格と判断したため。

[*8] レイアウト情報や修飾情報を持たない純粋に文字のみで構成されるデータ

[*9] GoogleAPI とは、様々なアプリケーションから Google のデータベースを利用できるインターフェースである。Google が公開しており無料で利用できるが、1 日に 1000 クエリーという制限がある。

2.3　システムの動作

開発したシステムの動作を順を追って説明する。動作の流れを図 **3** に示す。

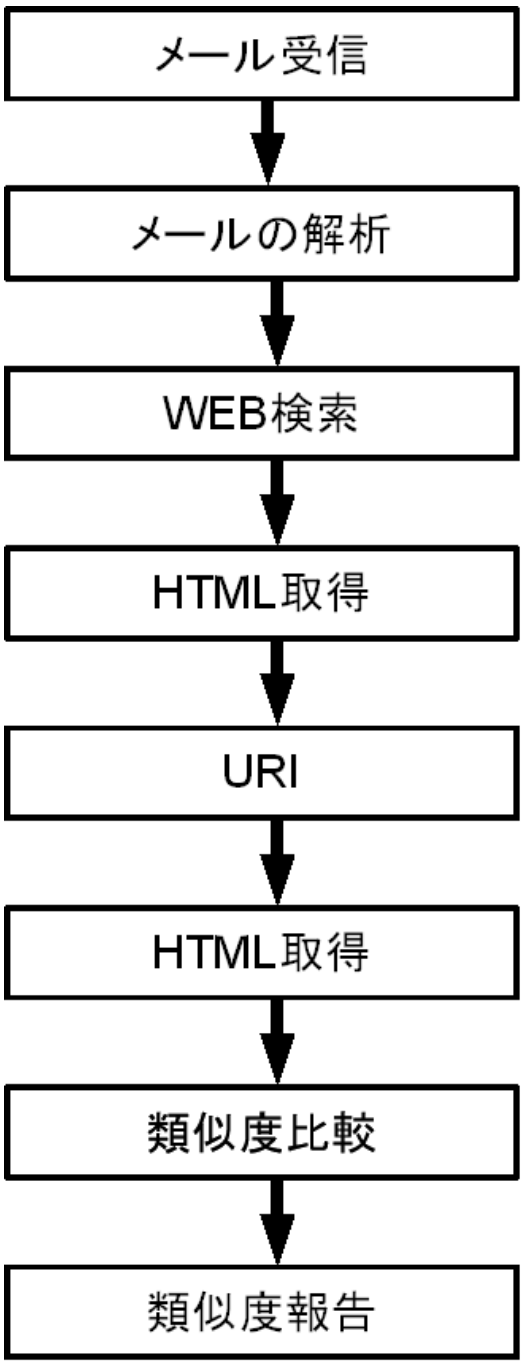


図 3　システムの流れ

2.3.1　ヘッダとボディの分割

メールを受信するとプログラムが起動する。起動するとメールのヘッダとボディを分割する。分割には sed[*10]を用いる。ヘッダとボディの間に存在する空行を境界に分割を行う。イメージを図 4 に示す。

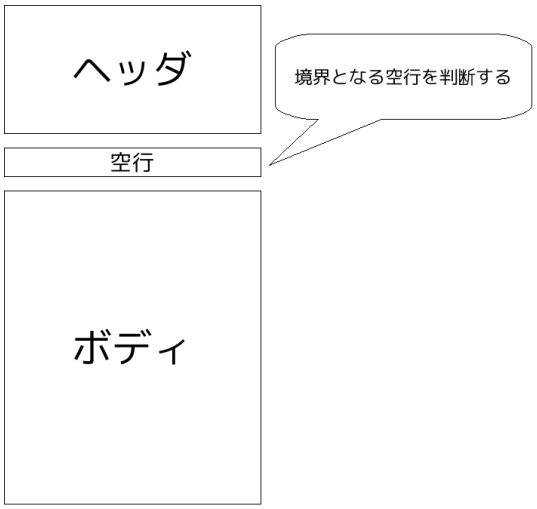


図 4　ヘッダとボディの分割

2.3.2　キーワード抽出

分割されたメールのボディを形態素解析ソフト kakasi を使って単語ごとにわかち書き[*11]の状態で抽出にする。

2.3.3　web 検索

抽出した単語を用いて web 検索を行う。検索に用いるブラウザはテキストブラウザ w3m を使用する。検索エンジンには yahoo を用いる。検索結果は HTML の状態でローカルに保存される。

2.3.4　URI の抽出

ローカルに保存された検索結果の HTML から URI のみを抽出する。

2.3.5　web ページの情報を取得

抽出した URI を使って、web ページの HTML を取得する。

2.3.6　判定

取得した HTML と提出されたメールのボディとを比較し、類似度を調べる。

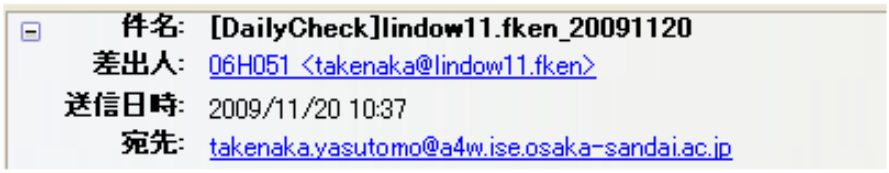
2.3.7　報告

類似度比較が終了すると、類似度を本文に記入したメールを教員に向けて発信する。メールはプログラムが生成して自動で送信される。

[*10] 文字列の置換や行の削除を行うコマンド
[*11] 文章において語の区切りに空白を挟んで記述することである

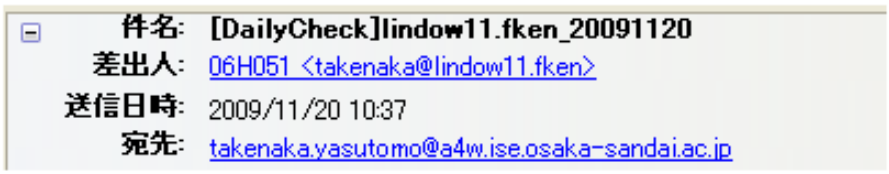
3　結果と考察

ここに実行結果を示す。送信した元のメールを図 6 に示し、返ってきた結果を図 6 に示す。



Diese E-Mail aus dem Internet konnten 75% dupliziert haben

図 5　出力結果



Diese E-Mail aus dem Internet konnten 75% dupliziert haben

図 6　出力結果

3.1 利点と問題点

不正に作成されたレポートを判別させるシステムを開発し、実際に稼働させた。以下に利点と問題点が挙げられる。

3.1.1 利点

類似度判定はシステムが自動で行うので負担が少ない。文字コード ISO-2022.JP[*12]でメールが送信できる環境が整っていればどこからでもシステムを通すことができる。

3.1.2 問題点

yahoo を用いての検索結果の HTML から URI を取り出す際に、全ての URI を抽出できない場合がある。HTML に記述された URI の途中にタブが付いており、それをプログラムが判断できないことが原因である。また、比較対象が web からのみなので、web 上からコピーされたレポート以外には対処できない。

[*12] インターネット上 (特に電子メール) などで使われる日本の文字用の文字符号化方式。ISO/IEC 2022 のエスケープシーケンスを利用して文字集合を切り替える 7 ビットのコードであることを特徴とする。JIS コードと呼ばれる

4　まとめ

本システムを用いて、テストを行った。wikipedia からコピーした文章を貼り付けたメールを送信したところ、75% 類似しているとの結果が出た。十分な精度を持っていると言えるが、既存の TOM との連携を図ることで他の学生が提出したレポートと web 両方から類似度比較が可能となり、コピーレポートの検知率が向上する。受信されたメールの内容が web からのコピーでないかの判断をプログラムが行うので、教員への負担はかなり低いと言える。

4.1　今後の課題

- TOM との連携現在稼働している TOM との連携を図ることで、他の学生が書いたレポートと web 両方から類似度比較を行う。

- URI 抽出機能強化検索結果の HTML から URI を取り出すとき、HTML に記述されている URI が一部抜けてしまう場合があるので、全ての URI が表示されるように改良を加える。検索結果の HTML が比較的単純な GoogleAPI を用いれば URI が抜けることは無いが、GoogleAPI はクエリ制限があるために実現は困難である。よって、現在の URI 抽出に改良を加える。yahoo の検索結果の HTML では URI の途中でタブが入っている場合がある。そのタブの存在をプログラムに判断させて除去することで URI の漏れを防ぐ。

- 精度の向上類似度の判定の精度を向上させて、コピーレポートを確実に見分けさせる。現在使用している判定式にも改良を加えることで精度が向上させる。

謝辞

本研究を進めていく上で、ご教授頂いた大垣斉講師、藤井信夫教授、情報安全工学研究室並びに情報システム工学科卒業生の方々に深く感謝の意を表します。

参考文献

[1] Shell Script, [http://www.k4.dion.ne.jp/ mms/unix/shellscript/index.html]

[2] Vine Linux, [http://www.vinelinux.org/]

[3] Proc Mail, [http://www.akatsuki.ne.jp/daemon/procmail/setup.html]

[4] w3m, [http://w3m.sourceforge.net/index.ja.html]

[5] 形態素解析ソフト, [http://www.lanches.co.jp/search/cache/1219626256.html]

(注)URI は全て 2009 年 11 月 25 日時点のもの

付録 A　ソースコード

```
#! /bin/sh
TMP1=/var/tmp/TMP1.$$
TMP2=/var/tmp/TMP2.$$

#MAIL=/home/2009/takenaka/sotuken/test/mail.txt

ENEMY=/home/2009/takenaka/sotuken/sign2.txt
OMAIL=a
MAIL=a
URI2=a
URI3=a

trap "rm -f $URI2 $URI3 $OBDY $BODY $TMP1 $TMP2; exit 0" 0 1 2 3 15

cat > $OMAIL

#解析
cat $OMAIL | \
sed -e '1,/^$/d' | sed -n -e '1,/^-- $/p' | \
grep -v "^>" | grep -v "^#" | nkf -Z1 | \
tr -s '\t' ' ' |  tr -d '\n' | \
kakasi -w | kakasi -JH | tr -s ' ' '\n' > $MAIL
cat $MAIL > /home/2009/takenaka/sotuken/body.txt

#文字コード変更
nkf -e /home/2009/takenaka/sotuken/body.txt > /home/2009/takenaka/sotuken/ebody.txt

#w3m で検索
for WORD in `cat /home/2009/takenaka/sotuken/ebody.txt | sed 's/,/ /g' | awk '{print $2,$3,$4}'`
do

#検索パターン
w3m -dump_source http://search.yahoo.co.jp/search?p=$WORD >> /home/2009/takenaka/sotuken/search.txt
#w3m  -dump 'http://search.yahoo.co.jp/search?p='$WORD' ie=euc-jp' >> /home/2009/takenaka/sotuken/search.t
#w3m -dump 'http://www.google.co.jp/search?q='$WORD' ie=euc-jp'
#w3m -dump  http://search.yahoo.co.jp/search?p=$WORD > /home/2009/takenaka/sotuken/search.txt

#cat  /home/2009/takenaka/sotuken/search.txt |  sed 's/<em class="yschurl">/\n/g' | sed 's/<wbr>/\n/g' > /


# for URI in `cat /home/2009/takenaka/sotuken/search.txt | grep 'div class="abs"' \
#              | sed 's/.*"yschurl">//g' | sed 's/<[^>]*>//g' | head -n 3`
#  do
    #echo -n ,$URI >> /home/2009/takenaka/sotuken/URI.txt
#done
done


URI=/home/2009/takenaka/sotuken/search.txt

#URI 抽出
cat $URI | grep 'div class="abs"' \
 | sed 's/.*"yschurl">//g' | sed 's/<[^>]*>//g' | head -n 3 >> $URI2


cat $URI2 | grep -v google | grep -v amazon | grep -v nicovideo | grep -v youtube  > $URI3
cat $URI3 > /home/2009/takenaka/sotuken/test/test2.txt


wget -np  --restrict-file-names=nocontrol --directory-prefix=/home/2009/takenaka/sotuken/html/ -i  /home/2

cd /home/2009/takenaka/sotuken/html/

i=1
#拡張子変更
for name in *.html ; do
   mv $name $name.txt
done

#ファイル名強制変更
for name2    in ?* ; do
   mv $name2 $i.txt
let i=${i}+1
done
# find . -type f | xargs -n 10 nkf -e --overwrite
```

```
#格納
OBDY=/home/2009/takenaka/sotuken/ebody.txt
for rname in *.txt ; do
    nkf -e $rname >>  /home/2009/takenaka/sotuken/test/test3.txt

    WGET=/home/2009/takenaka/sotuken/test/test3.txt
#echo $WGET

    tail +3 $OBDY | grep -v "^>" >$TMP1
    tail +3 $WGET | grep -v "^>" >$TMP2

    CO=‘cat $TMP1 | wc -l‘
   echo $CO
    CC=‘/usr/bin/diff -b -B $TMP1 $TMP2 | wc -l‘
    HAN=‘expr 100 - $CC \* 100 / $CO - 1‘
#    cat $HAN
if [ $HAN -gt 75 ]
then
echo  "${HAN}" | cat >  /home/2009/takenaka/sotuken/p3.txt
fi


#cat  $HAN >> /home/2009/takenaka/sotuken/p3.txt
done

adad=takenaka.yasutomo@a4w.ise.osaka-sandai.ac.jp

cat <<END | /usr/lib/sendmail -t
To: takenaka.yasutomo@a4w.ise.osaka-sandai.ac.jp
Subject: [DailyCheck]‘hostname‘_20‘date +%y%m%d‘
Mime-Version: 1.0
Content-Type: text/plain; charset=ISO-2022-JP

Diese E-Mail aus dem Internet konnten {$HAN}% dupliziert haben
END

exit 0
```

A.1　付録 1.1