2011年度 卒業論文

コピーレポート検知システムの実装と評価

大阪産業大学 工学部 情報システム工学科
情報教育システム研究室

08H067 高橋徹

目次

1　はじめに

今日、インターネットの普及によって欲しい情報を容易に取得できるようになった。そのため、大学でレポート課題が出されると検索エンジンを使って検索し、検索結果の Web ページをコピー＆ペーストしただけのレポート (以下コピーレポート) を提出する学生がいる。当然このようなレポートは成績に反映するべきではない。しかし、教員側で１つ１つレポートを見てチェックしようとすると大変な労力がかかってしまう。かといってコピーレポートを黙認してしまうと自身で考えてレポートを書く学生が減り、学生の質の低下につながってしまう。その対策として、コピーレポートを提出する学生に対して何らかの形で警告を出すシステムが必要である。コピーレポートを検知するシステムはこれまでにも開発されており、いくつかの大学では実際に稼働している。しかし、既存のシステムはレポートの収集やシステムの起動を人の手で行うため、教員にとっては負担となっている。教員の負担を増やすことなく、自動でレポートを解析しコピーレポートを検知するシステムが必要である。

2 本研究の目的

本研究の目的は、学生が提出したコピーレポートを防ぎ、学生自身が考えて書いたレポートの提出を促すことでコピーレポートをなくすシステムを構築することである。また、構築したシステムを実際に稼働させシステムの有効性を検証することである。これまでにもコピーレポートの提出を防ぐシステムは開発されており [1]、いくつかの大学では実際に使用されている [2]。しかし、既存のシステムはレポートの収集やシステムの起動に人の手がかかるという問題点がある。人の手がかかるため、当然教員の負担も大きい。この問題を解消するためにレポートを自動で処理し、コピーレポートの提出を防ぐシステムを構築した。本研究で構築するシステムはメイルで提出されたレポートを自動で処理し、コピーレポートであった場合は警告メイルを送るものである。警告メイルにはコピー元と考えられる文章や URI を記述することによって、どの Web ページからどの部分をコピーしたかがすぐにわかる。この方法であれば、警告メイルをチェックするだけでコピーレポートの有無を調べることが出来るので、教員への負担も少ないと考えられる。

3 先行研究

レポートが Web ページからコピーされたものかどうかを判定するシステムはこれまでにも開発されている [1, 2]。また、大阪産業大学の大垣斉講師の講義では提出されたレポートを解析し、他人のレポートをコピーしているかどうかを検知し警告を出す TOM というシステムが存在する。

3.1 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。現在大阪産業大学工学部の情報システム工学科で稼働している。メイルの受信をトリガーに起動し、他人のメイルをコピーしているかどうかを判定する。TOM には Web ページからのコピーを検知する機能はないので、本研究ではコピーレポート検知システムを開発し、TOM との連携を試みた。

- コピーレポート検知システム [3]: 大阪産業大学工学部情報システム工学科の竹中康朝氏が開発したシステム。TOM と同じくメイルの受信をトリガーに起動する。本研究と同じく Web ページからのコピーを検知するシステムであるが、コピーレポートの検知率が低く実用段階には達していない。本研究ではこのコピーレポート検知システムを改良し、システムを構築した。

- コピペルナー [1]: 金沢工業大学の杉光一成教授が考案し、株式会社アンクが開発したシステム。
  本研究と同じくコピーレポートを検出するシステムであるが、レポートの収集やシステムの起動など一部の作業は手動で行われる。

- 名称不明: 阪南大学花川研究室で開発されたシステム [2]。本研究と同じく学生の出したコピーレポートを検知するシステムである。コピペルナーと同じく起動さえすれば自動で判定を行ってくれるが、システムの起動自体は毎回手動で行われる。

4 仕様と用語説明

4.1 システムの概要

メイルで提出されたレポートが Web ページからのコピーされたものかどうかを判定する。受け取ったメイルを解析しメイルの本文を検索にかけ、検索結果からコピー疑いのある Web ページを取得する。メイル本文と取得した Web ページを圧縮プログラムを用いて比較する。本文の 5 割以上にコピー判定が認められた場合は、講義の ML[*1]へコピー疑いのある文やコピー元の URI を記述した警告メイルを送信する。

4.2 形態素解析

開発したシステムで、比較時に使用する文章には形態素解析[*2]にかけた文章を用いる。形態素解析ソフトウェアはいくつか種類がある。

- Juman [4]: 京都大学言語メディア研究室で開発された形態素解析ソフトウェア。隠れマルコフモデル[*3]を使った解析で高い精度を誇っており、Chasen [5] の元となったシステムである。

- Chasen [5]: 奈良先端科学技術大学院大学で開発された形態素解析ソフトウェア。Juman [4] を元に開発され、現在ではテキストマイニング[*4]や音声合成用の振り仮名付けなどに利用されている。

- MeCab [6]: 京都大学情報学研究で開発された形態素解析ソフトウェア。言語、辞書に依存しない汎用的な設計を基本方針として開発されており、品詞情報を利用した解析・推定を行うことができる。

[*1] メイリング リスト
[*2] 自然言語で書かれた文章を分割し、それぞれの品詞を判別する
[*3] 観測可能な情報から未知のパラメータを推定する確率モデルの一つ
[*4] 文章からなるデータを解析し、出現傾向や時系列を解析することで有用な情報を取り出す分析手法

4.2.1 形態素解析エンジンの比較

Juman、Chasen、MeCab を対象に形態素解析の比較を行う。**図 1** に示す解析元の文章を Juman、Chasen、Mecab で解析した。解析結果を**図 2**、**図 3**、**図 4** にそれぞれ示す。

ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。

図 1　解析元の文章の例 学生のレポートから一文を抽出した。

ALOHA ALOHA ALOHA 未定義語 15 その他 1 * 0 * 0 NIL
と と と 助詞 9 格助詞 1 * 0 * 0 NIL
@ と と と 助詞 9 接続助詞 3 * 0 * 0 NIL
イーサネット イーサネット イーサネット 未定義語 15 カタカナ 2 * 0 * 0 NIL
の の の 助詞 9 格助詞 1 * 0 * 0 NIL
違い ちがい 違う 動詞 2 * 0 子音動詞ワ行 12 基本連用形 7 ”付属動詞候補（基本） 代表表記:違う”
は は は 助詞 9 副助詞 2 * 0 * 0 NIL
、 、 、 特殊 1 読点 2 * 0 * 0 NIL
イーサネット イーサネット イーサネット 未定義語 15 カタカナ 2 * 0 * 0 NIL
で で で 助詞 9 格助詞 1 * 0 * 0 NIL
は は は 助詞 9 副助詞 2 * 0 * 0 NIL
CSMA/CD CSMA/CD CSMA/CD 未定義語 15 その他 1 * 0 * 0 NIL
を を を 助詞 9 格助詞 1 * 0 * 0 NIL
採用 さいよう 採用 名詞 6 サ変名詞 2 * 0 * 0 ”代表表記:採用”
して して する 動詞 2 * 0 サ変動詞 16 タ系連用テ形 11 ”付属動詞候補（基本） 代表表記:する”
いる いる いる 接尾辞 14 動詞性接尾辞 7 母音動詞 1 基本形 2 NIL
と と と 助詞 9 格助詞 1 * 0 * 0 NIL
いう いう いう 動詞 2 * 0 子音動詞ワ行 12 基本形 2 NIL
点 てん 点 名詞 6 普通名詞 1 * 0 * 0 ”漢字読み:音 代表表記:点”
である である だ 判定詞 4 * 0 判定詞 25 デアル列基本形 14 NIL
。 。 。 特殊 1 句点 1 * 0 * 0 NIL

図 2　Juman を用いた解析結果 動詞や助詞の分類は優れているが、名詞の分類が甘く、未定義語に分類されるものが多い。

```
A エイ A 記号-アルファベット
L エル L 記号-アルファベット
O オー O 記号-アルファベット
H エッチ H 記号-アルファベット
A エイ A 記号-アルファベット
と ト と 助詞-並立助詞
イーサネット 未知語
の ノ の 助詞-連体化
違い チガイ 違い 名詞-ナイ形容詞語幹
は ハ は 助詞-係助詞
、 、 、 記号-読点
イーサネット 未知語
で デ で 助詞-格助詞-一般
は ハ は 助詞-係助詞
C シー C 記号-アルファベット
S エス S 記号-アルファベット
M エム M 記号-アルファベット
A エイ A 記号-アルファベット
/ / / 記号-一般
C シー C 記号-アルファベット
D ディー D 記号-アルファベット
を ヲ を 助詞-格助詞-一般
採用 サイヨウ 採用 名詞-サ変接続
し シ する 動詞-自立 サ変・スル 連用形
て テ て 助詞-接続助詞
いる イル いる 動詞-非自立 一段 基本形
と ト と 助詞-格助詞-引用
いう イウ いう 動詞-自立 五段・ワ行促音便 基本形
点 テン 点 名詞-非自立-一般
で デ だ 助動詞 特殊・ダ 連用形
ある アル ある 助動詞 五段・ラ行アル 基本形
。 。 。 記号-句点
```

図 3　Chasen を用いた解析結果 アルファベットや記号の分類も正確で、記号は他の形態素解析ソフトウェアに比べてさらに細かく分類される。

```
ALOHA 名詞, 固有名詞, 組織,*,*,*,*
と 助詞, 並立助詞,*,*,*,*, と, ト, ト
イーサネット 名詞, 一般,*,*,*,*,*
の 助詞, 連体化,*,*,*,*, の, ノ, ノ
違い 名詞, ナイ形容詞語幹,*,*,*,*, 違い, チガイ, チガイ
は 助詞, 係助詞,*,*,*,*, は, ハ, ワ
、 記号, 読点,*,*,*,*,、, 、, 、
イーサネット 名詞, 一般,*,*,*,*,*
で 助詞, 格助詞, 一般,*,*,*, で, デ, デ
は 助詞, 係助詞,*,*,*,*, は, ハ, ワ
CSMA 名詞, 固有名詞, 組織,*,*,*,*
/ 名詞, サ変接続,*,*,*,*,*
CD 名詞, 一般,*,*,*,*,*
を 助詞, 格助詞, 一般,*,*,*, を, ヲ, ヲ
採用 名詞, サ変接続,*,*,*,*, 採用, サイヨウ, サイヨー
し 動詞, 自立,*,*, サ変・スル, 連用形, する, シ, シ
て 助詞, 接続助詞,*,*,*,*, て, テ, テ
いる 動詞, 非自立,*,*, 一段, 基本形, いる, イル, イル
という 助詞, 格助詞, 連語,*,*,*, という, トイウ, トユウ
点 名詞, 非自立, 一般,*,*,*, 点, テン, テン
で 助動詞,*,*,*, 特殊・ダ, 連用形, だ, デ, デ
ある 助動詞,*,*,*, 五段・ラ行アル, 基本形, ある, アル, アル
。 記号, 句点,*,*,*,*,。, 。, 。
```

図 4　MeCab を用いた解析結果 Juman に比べて単語の分類精度が上がっているが、一部の記号や未定義語が名詞に分類されてしまう。

解析結果より、Juman や MeCab に比べて Chasen のほうがアルファベットの分類や記号の分類が正確であった。品詞の分類では MeCab も優れているが、未定義語や一部の記号まで名詞に分類されてしまうため、解析には向かないと判断した。他にもいくつかのサンプルを学生のレポートから抽出し実験を行ったが、同様の結果が得ることが出来た。よって本研究では Chasen を採用する。

4.3 テキストブラウザ

Web ページの取得には w3m [7] を用いる。w3m とはテキストブラウザ[*5]の一種で CUI[*6]を用いて、コマンドラインから利用するフリーのページャ/テキストベース WWW [*7]ブラウザである。ターミナルエミュレータ[*8]を使って Web を閲覧出来るほか、HTML をテキスト形式に整形するツールとしても利用できる。本研究で w3m を使用した理由は Web ページをプレーンテキストに整形して取得出来るからである。

4.4 検索エンジン

Web ページからのコピーを調べるためにメイル本文を一文ずつ検索にかけ、検索結果を取得する。検索結果を取得するための検索エンジンには Yahoo!JAPAN を用いる。他にも有名な検索エンジンとして GoogleAPI [8] が挙げられるが、一日に検索できる回数に制限があるため Yahoo!JAPAN を採用した。

4.5 文章の比較

メイル本文と Web ページとの比較には圧縮プログラムを用いた類似度判別 [9] を行う。圧縮プログラムは本来圧縮元のデータ中から冗長な部分を識別し、より短いデータに置き換えることで容量を節約するといった目的で使われているが、近年ではこれを類似度判別に用いる研究が行われている [10–12]。圧縮プログラムを用いた類似度判別の手法は非常にシンプルなもので、例えば比較したい二つのテキストデータがあった場合に、その二つが類似していればいるほど共通する冗長な部分は多くなると考えられる。そのため二つのデータを連結した場合に、そのデータの圧縮率が高ければ高いほど類似度は高いと言える。圧縮プログラムを用いた著者推定の論文では、多数の文献と著者不明の文献を比較し圧縮したテキストデータサイズの差を取ることで著者を推定していた。しかし、本研究では比較するテキストデータはメイル本文と Web ページの本文の二つのみである。そこで、比較式に改良を加え二つのデータのみで文章の類似度を判別出来るようにした。なお、比較はメイル本文と Web ページから一文ずつ抽出し文単位で行う。

[*5] Web ページを text/html のみで表示する Web ブラウザ
[*6] 情報の表示を文字によって行い、すべての操作をキーボードで行うユーザインタフェース
[*7] World Wide Web の略称
[*8] 端末として動作するソフトウェアのこと

4.5.1 比較に用いるテキストデータ

改良した比較式ではメイル本文と Web ページの文章を用いて 3 つのテキストデータを用意する。テキストデータの内容を以下に記述する。

- メイルの一文を 2 回繰り返し繋げたもの (以下 MailZip と呼ぶ)
- Web ページの一文を 2 回繰り返し繋げたもの (以下 WebZip と呼ぶ)
- メイルの一文と Web ページの一文を繋げたもの (以下 MailWebZip と呼ぶ)

3 つのテキストデータの例を**図 5**、**図 6**、**図 7** にそれぞれ示す。

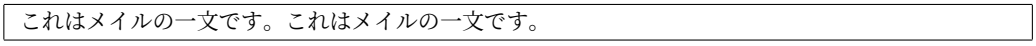

図 5　MailZip を用いた例 メイル本文の一文を二回繰り返し繋げる。

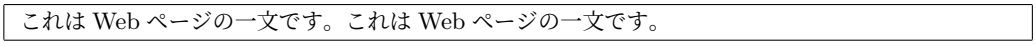

図 6　WebZip を用いた例 Web ページ文の一文を二回繰り返し繋げる。

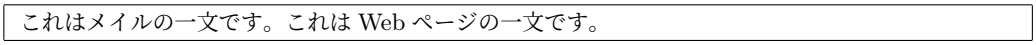

図 7　MailWebZip を用いた例 メイル文の一文と、Web ページ文の一文を繋げる。

4.6 文章の解析

3 つのテキストデータを Chasen で解析し、記号や未知語、接続詞の除去を行う。記号や未知語、接続詞に分類される単語は比較には不要と判断したからである。

4.6.1 圧縮率の取得

Chasen で解析したテキストデータを圧縮し、出力される deflated の値から圧縮率を取得する。圧縮には zip[*9] を使用する。zip を採用した理由は、標準的な圧縮形式でメジャーな OS であれば実装が容易に出来る点や、テキストデータによる圧縮率が高いとされるからである。圧縮の例を**図 8** に示す。

```
$ zip Test Mail-Web-Zip.txt
  adding: Mail-Web-Zip.txt (deflated 22%)
$
```

図 8　MailWebZip の圧縮例 この例では圧縮元のデータサイズを 22% 圧縮したことを表す。deflated 値が高いほど、圧縮率は高いといえる。

4.6.2 比較

類似度を算出するために比較を行う。類似度の算出を以下の式で求める。

$$\text{類似度} = 1 - \frac{(MailZip\text{の圧縮率} + WebZip\text{の圧縮率})/2 - MailWebZip\text{の圧縮率}}{(MailZip\text{の圧縮率} + WebZip\text{の圧縮率})/2} \tag{1}$$

まず MailZip と WebZip の圧縮率の平均値を算出する。MailZip と WebZip は両方が同じ文章を 2 回繰り返したテキストデータを圧縮しているので、当然圧縮率は高くなる。算出した平均値から MailWebZip の圧縮率を引き、その値をさらに平均値で割る。仮に MailZip の文章と WebZip の文章が酷似していれば、MailWebZip の圧縮率は MailZip と WebZip の圧縮率の平均値に近い値を出すはずなので、類似度が高い場合は算出された値は小さくなるはずである。1 からその値を引いた数値が類似度となる。数値が 1 に近いほど類似度が高く、遠いほど類似度が低いといえる。

[*9] ファイルを zip 形式で圧縮するコマンド

どこまでの類似度の値をコピー判定とするかについては、あらかじめ手動で Web ページからのコピーと考えられるレポート、Web ページからのコピーではないと考えられるレポートの文章を抽出し上記の比較方法を用いて実験を行った。Web ページからのコピーと考えられるレポートに関してはコピー元の文章をあらかじめ手動で抽出して値を算出した。コピーをしていないレポートに関しては、本研究で構築するシステムと同じ手法で取得した Web ページとの比較を行い値を算出した。

サンプルとして用いたレポートのうち、明らかに Web からのコピーと考えられるレポートを対象として**式 (1)** による類似度を求めた。その分布を**図 9** に示す。また、同様に Web ページからのコピーではないと考えられるレポートを対象として**式 (1)** による類似度を求めた。その分布を**図 10** に示す。

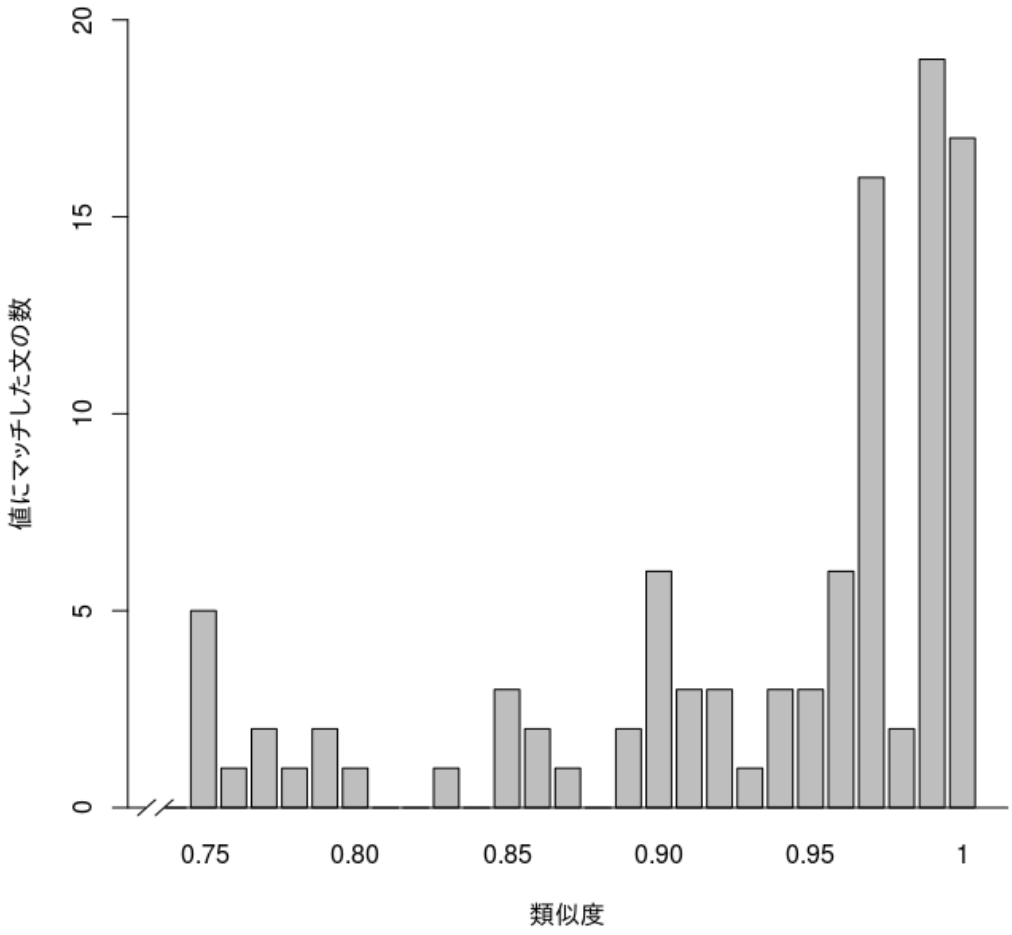

図 9　コピーレポート文の類似度の分布 Web ページとの文章の違いが少ないほど数値は高くなり、値が 1 の場合はそのままコピーしたことを表す。

全部で 100 文のコピーレポート文を用いて値を算出した。類似度が最も高かったのは類似度が 1 の文章であった。類似度が 1 の場合は、Web ページの文章を改変せずにコピーしている場合である。また、類似度が最も低かったのは 0.75 であった。よって以後の実験では類似度が 0.75 以上の場合はコピー文と仮定する。

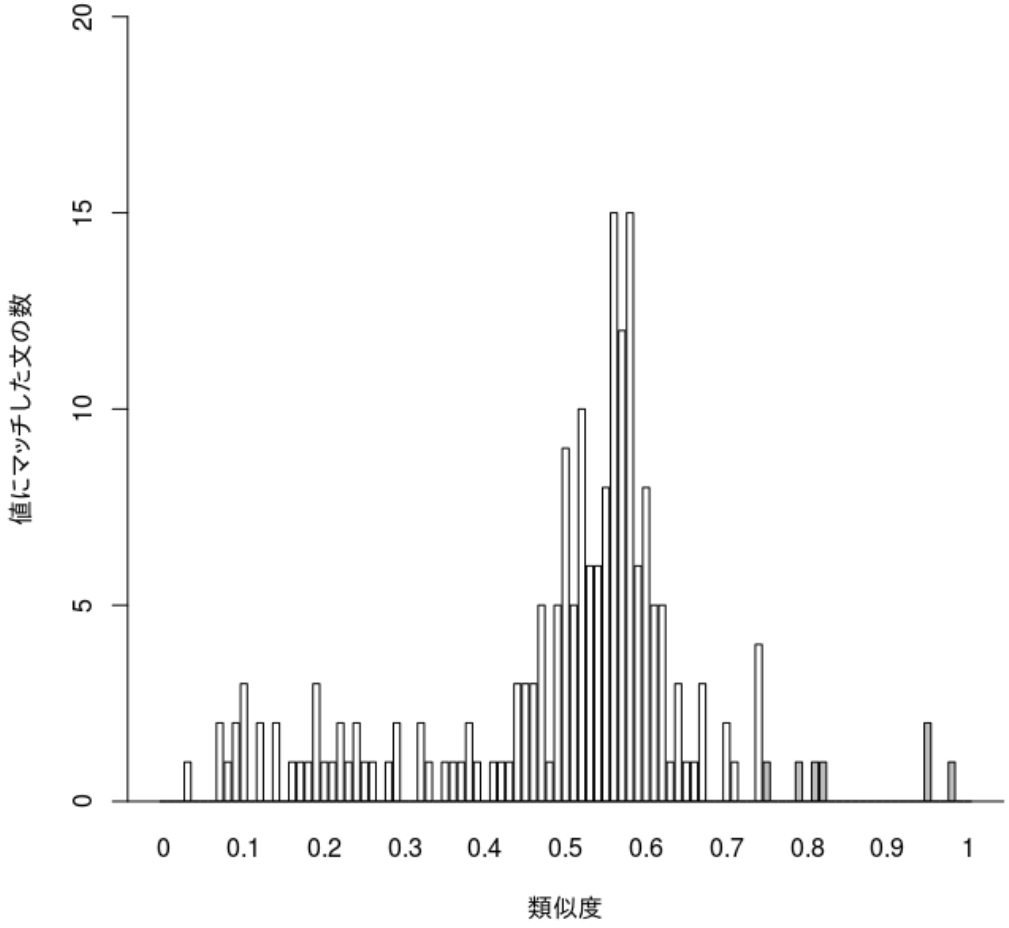

図 10　コピーをしていない文の圧縮率の分布 数値が小さくなるほど Web ページからコピーしている可能性が低くなる。

コピーではないと考えられる文を元に Web ページを取得し、比較を行った。全部で 196 回の比較を行い、類似度を算出した。グラフより、最も類似度が高かったの 0.98 であった。逆に最も類似度が低かったのは 0.03 であった。

明らかにコピーではないと考えられる文を用いて**式 (1)** による類似度を求めたにも関わらず、類似度が 0.75 以上の場合はコピーとみなすという前提条件を考えると 7 文が誤検知を起こしている。原因として比較に使用するメイル文の文字数と Web ページ文の文字数の差があまりにも大きい場合に比較がうまくいかなかったことが挙げられる。文字数に差がありすぎると圧縮率による比較がうまくいかず、誤検知を起こしてしまう。

比較がうまくいかないことへの対策として比較する Web ページ文とメイル文の文字数を比べ、文字数の差が大きい場合には比較を行わないことで誤検知を防ぐ。概略を**図 11** に示す。

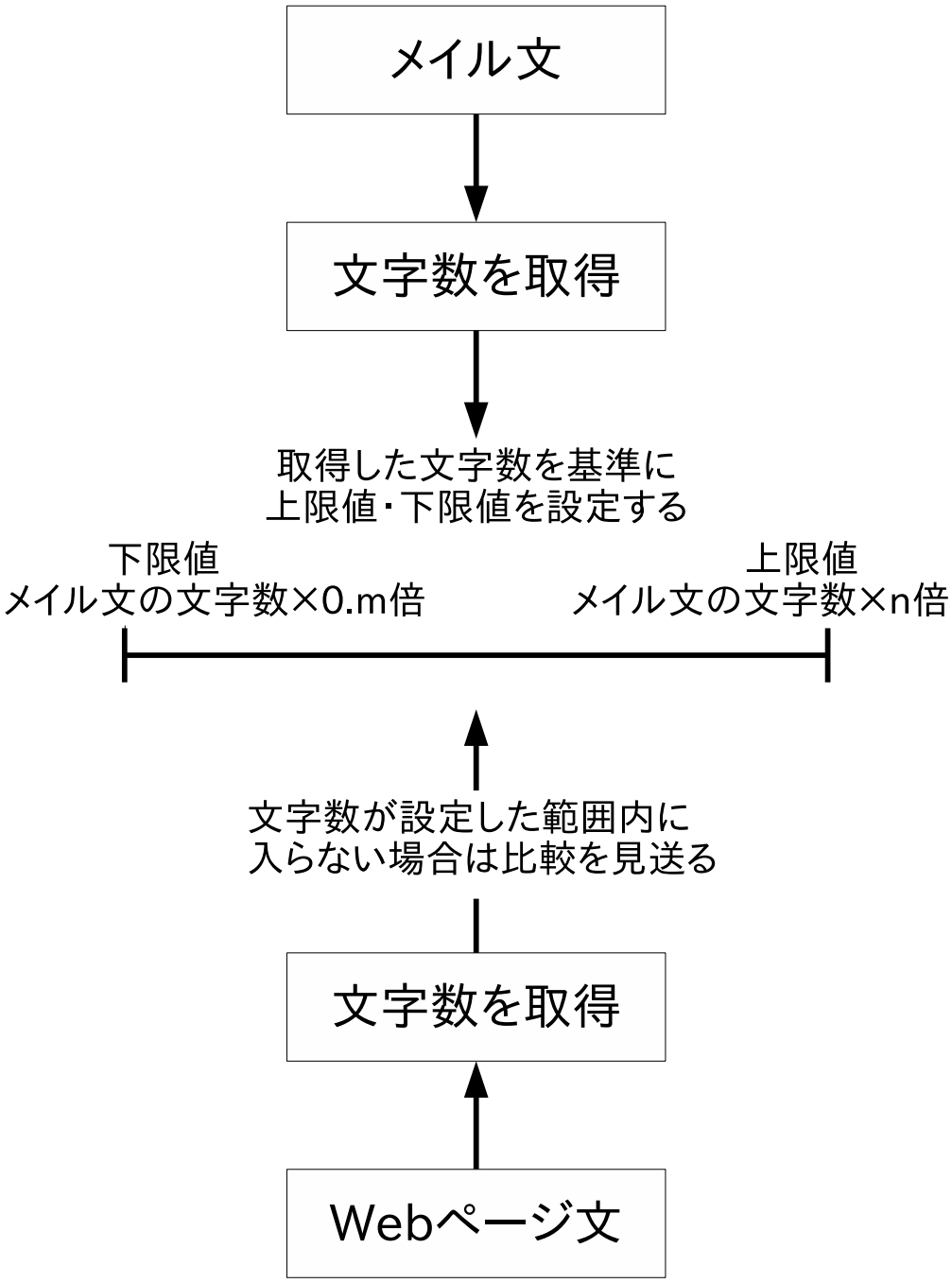

図 11　メイル文の文字数を取得し、取得した文字数から上限値と下限値を設定する。Web ページ文の文字数が設定した文字数の範囲に入らなかった場合は比較を見送る。

上限値と下限値の決定には、**図 10** で用いたコピーをしていないと考えられる文を用いて実験を行った。実験毎に文字数の上限値と下限値の設定を変え、類似度が 0.75 以上の場合はコピーとみなすという前提条件を元に誤検知の数を調査した。上限値と下限値の設定を以下に示す。

- 下限値=設定なし 上限値=文字数の 2 倍
- 下限値=文字数の 0.5 倍 上限値=文字数の 2 倍
- 下限値=文字数の 0.5 倍 上限値=文字数の 3 倍
- 下限値=文字数の 0.5 倍 上限値=文字数の 4 倍

実験結果を**表 1** に示す。

表 1　コピーではないと考えられる文を用いた実験 メイル文の文字数を元に下限値と上限値を設定する。設定範囲を変化させ誤検知数の調査を行った。

下限値-上限値の設定	設定なし	設定なし-2 倍	0.5 倍-2 倍	0.5 倍-3 倍	0.5 倍-4 倍
誤検知の数	7 文	3 文	1 文	1 文	4 文

実験結果より下限値と上限値の値をメイル文の文字数の 0.5 倍から 2 倍、0.5 倍から 3 倍までに設定した場合には、誤検知は 1 文に抑えられ、比較が正常に出来ていると考えられる。

次に下限値と上限値を設定した場合にコピーレポートの検知率がどの程度変化するかを調査した。**図 9** よりコピーレポート文を用いた場合の類似度は 0.75 から 1 の間を推移することがわかったので、**図 9** で使用したコピーレポート文を用いて、**式 (1)** により圧縮率を求め、類似度が 0.75 以上であった場合にはコピーと判定する。調査結果を**表 2** に示す。

表 2　コピーレポート文を用いた実験 **図 9** で用いた 100 文の Web からのコピーと考えられる文を用いた。あらかじめメイル文の文字数を取得し、下限値と上限値を設定する。設定範囲を変化させ調査を行った。

下限値-上限値の設定	設定なし	設定なし-2 倍	0.5 倍-2 倍	0.5 倍-3 倍	0.5 倍-4 倍
検出率	87%	90%	92%	94%	91%
誤検出率	12%	8%	5%	5%	8%
検出漏れ率	1%	2%	3%	1%	1%

本実験では、あらかじめコピー元の文章を抽出し比較を行った**図 9** の実験時とは違い、本研究で用いるシステムの動作と同じくコピーレポート文をもとに取得した Web ページを整形し、1 文目から順に**式 (1)** による比較を行った。検出率はコピー元の文章を正しく検出出来た割合、誤検出率はコピー元の文章と比較を行う前にコピー元でない文章との比較で類似度 0.75 以上の値が出た場合の割合、検出漏れ率はコピー元の文が検出出来なかった割合である。表より、上限値と下限値を設定しなかった場合の検出率は 87% であった。文字数の範囲を限定し、下限値と上限値をメイル文の文字数の 0.5 倍、3 倍までに設定した場合は検出率が 94% まで上昇した。よって本研究では類似度が 0.75 以上であればコピーと判定し、Web ページ文の文字数がメイル文の文字数の 0.5 倍から 3 倍の範囲内であれば比較を行い、そうでなければ比較を行わないこととする。

5　システムの動作

メイルを受信するとシステムが起動する。メイル本文を解析し一文ずつに整形する。整形したメイル本文を一文ずつ検索エンジンにかけ、検索結果から URI を抽出し格納する。格納した URI から Web ページ本文を取得し、整形する。整形した Web ページ文と検索元のメイル文を一文ずつ比較する。メイル本文全体の 5 割以上にコピー判定が出た場合は、コピー判定の出たメイルと Web ページの文章、Web ページの URI を記述した警告メイルを講義の ML へ送信する。システムの動作を**図 12** に示す。

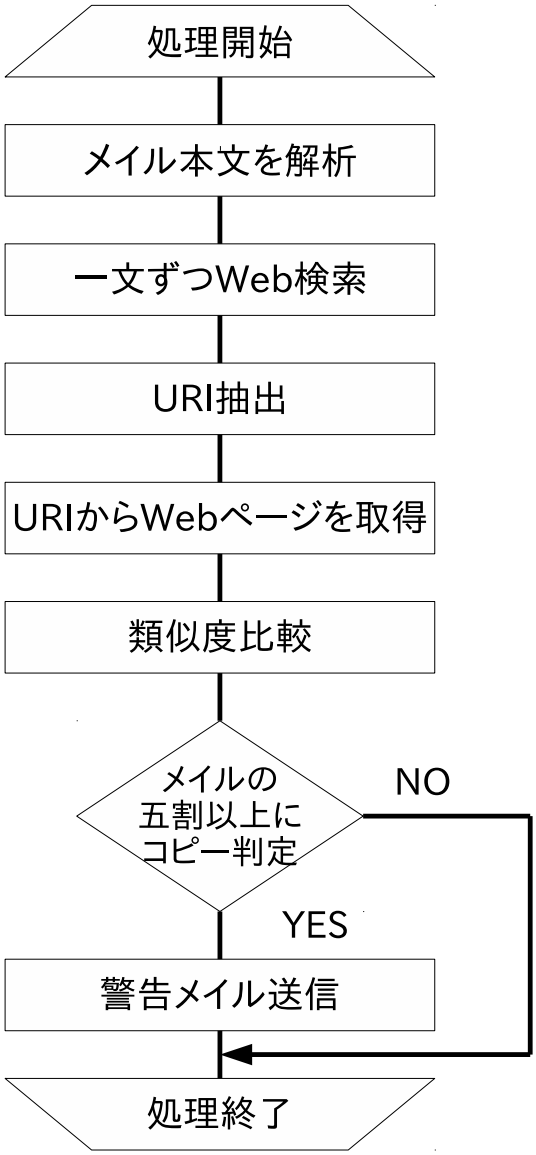

図 12　システムの動作 メイルを受信するとシステムが起動し、メイル本文が Web ページからのコピーかどうかを調べる。本文の 5 割以上にコピー判定が出た場合は警告メイルを送信する。

5.1 メイルの解析

メイルの受信をトリガーにシステムが起動する。受け取ったメイルをヘッダとボディに分割する。シグネチャは比較には必要ないのでこの段階で除去する。分割、除去には grep[*10]、sed[*11]を用いる。

5.2 本文の整形

メイル本文を解析し、一文ずつに整形する。整形には sed を用いる。整形前の文章の例を**図 13** に、整形後の文章の例を**図 14** にそれぞれ示す。

ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。これは衝突発生時に接続している全コンピュータにジャミング信号をブロードキャストし、各コンピュータに現在のパケットやフレームを捨てさせる。伝送遅延が伝播遅延に比べて支配的な場合、ジャミング信号を使うと伝送媒体を素早く解放でき、多くのイーサネットに適している。ALOHA は無線システムなので、小規模の LAN ではうまく機能するプロトコルでも常にうまく機能するとは限らないという問題もある（隠れ端末問題など）。ハワイ諸島のネットワークの通信範囲は直径 400km だったが、伝播遅延はほぼ確実に伝送遅延に比べて小さかったため、プロトコルは十分な頑健性を備えている必要があった。

図 13　整形前の文章の例

ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。

これは衝突発生時に接続している全コンピュータにジャミング信号をブロードキャストし、各コンピュータに現在のパケットやフレームを捨てさせる。

伝送遅延が伝播遅延に比べて支配的な場合、ジャミング信号を使うと伝送媒体を素早く解放でき、多くのイーサネットに適している。

ALOHA は無線システムなので、小規模の LAN ではうまく機能するプロトコルでも常にうまく機能するとは限らないという問題もある（隠れ端末問題など）。

ハワイ諸島のネットワークの通信範囲は直径 400km だったが、伝播遅延はほぼ確実に伝送遅延に比べて小さかったため、プロトコルは十分な頑健性を備えている必要があった。

図 14　整形後の文章の例

5.3 検索

整形したメイル本文を一文ずつ読み込み、句読点を除去する。除去した文章を検索にかける。
検索には Yahoo!JAPAN の検索エンジンを用いる。

[*10] ファイルに対してマッチングを行い、行の抽出や削除を行うコマンド
[*11] 文字列の置換や、行の削除を行うコマンド

5.4 URI 抽出

検索結果を取得し検索結果の第一位の URI を取得する。検索結果の取得には wget[*12]を使用し、URI の抽出には grep、sed を用いる。URI 抽出の動作、例を**図 15**、**図 16** にそれぞれ示す。

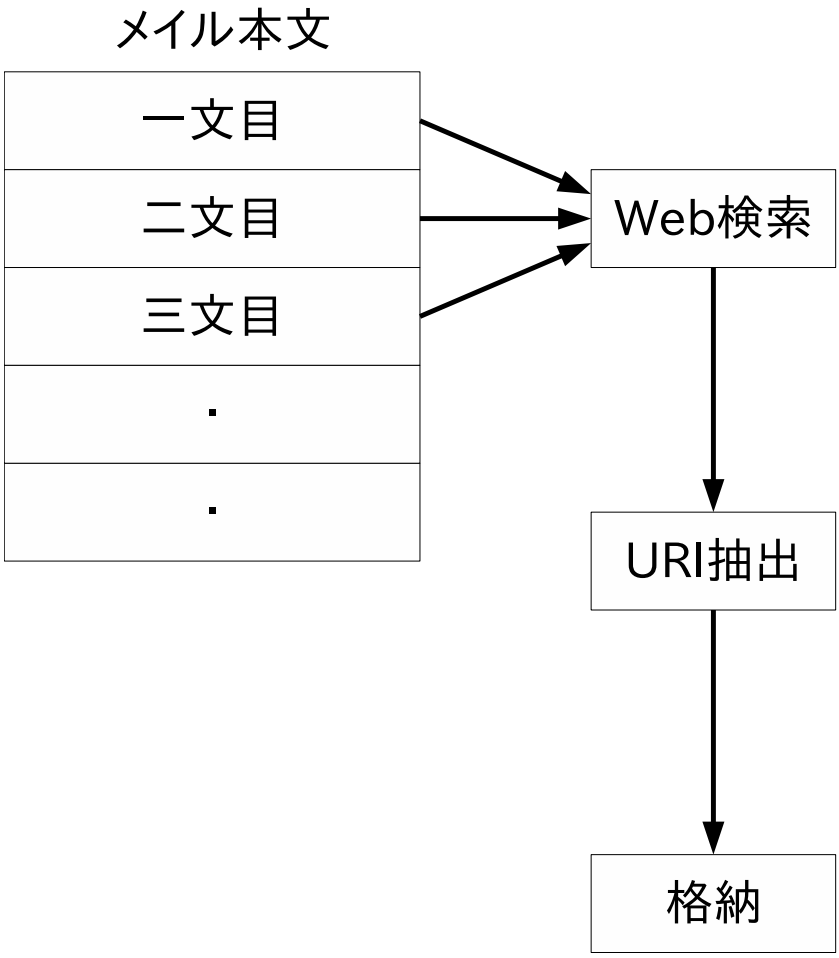

図 15　URI 抽出方法 整形したメイル本文を一文ずつ読み込み検索エンジンにかける。検索結果から URI を抽出し、格納する。

```
http://ja.wikipedia.org/wiki/ALOHAnet
http://ja.wikipedia.org/wiki/ALOHAnet
http://ja.wikipedia.org/wiki/ALOHAnet
http://ja.wikipedia.org/wiki/ALOHAnet
http://www.atmarkit.co.jp/fwin2k/network/tcpip005/tcpip02.html
http://www.atmarkit.co.jp/fwin2k/network/tcpip005/tcpip02.html
http://www.atmarkit.co.jp/fwin2k/network/tcpip005/tcpip02.html
```

図 16　格納した URI の例

[*12] コマンドライン上でファイル取得を行うコマンド

5.5 Web ページの取得・整形

抽出した URI から Web ページを取得する。Web ページの取得には w3m を用いる。取得した Web ページをメイル本文と同じように一文ずつに整形する。整形には sed を用いる。

5.6 類似度比較

メイル本文と整形した Web ページの本文を一文ずつ比較する。比較には圧縮プログラムを使用し、文章の圧縮率の差から類似度を算出する。**4.6.2** で求めた通り 2 つの文章の類似度が 0.75 以上であった場合はコピー文とみなす。

5.7 警告メイル送信

本文の 50% 以上の文にコピー判定が認められると講義の ML に向けて警告メイルを送信する。警告メイルには以下の内容をメイルのボディ部分に記述する。

- 送信者のメイルアドレス
- 検知したメイルの Message-ID
- コピー判定の出たメイルの文章
- コピー元と思われる Web ページの URI と文章

コピーレポートの例を**図 17** に、返信された警告メイルを**図 18** に示す。

ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。
これは衝突発生時に接続している全コンピュータにジャミング信号をブロードキャストし、
各コンピュータに現在のパケットやフレームを捨てさせる。
伝送遅延が伝播遅延に比べて支配的な場合、
ジャミング信号を使うと伝送媒体を素早く解放でき、
多くのイーサネットに適している。
ALOHA は無線システムなので、小規模の LAN ではうまく機能するプロトコルでも
常にうまく機能するとは限らないという問題もある（隠れ端末問題など）。
ハワイ諸島のネットワークの通信範囲は直径 400km だったが、
伝播遅延はほぼ確実に伝送遅延に比べて小さかったため、
プロトコルは十分な頑健性を備えている必要があった。

–
Takahashi Tohru

図 17　コピーレポートの例 本文に wikipedia の文章をコピーした。

警告
Takahashi.Toru@a4w.ise.osaka-sandai.ac.jp の書いたメイル
> <CAO_3qQWgJVJf_S4EBUobKE6OdT5P5K2R–0w9HJu6D_EEmUToA@mail.gmail.com>
は本文の 5 割以上を Web からコピーした可能性があります。
コピー疑いが認められた箇所は以下の通りです。

メイル本文
> ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。
コピー元 <http://ja.wikipedia.org/wiki/ALOHAnet>
> ALOHA とイーサネットの違いは、イーサネットでは CSMA/CD を採用しているという点である。

メイル本文
> これは衝突発生時に接続している全コンピュータにジャミング信号をブロードキャストし、各コンピュータに現在のパケットやフレームを捨てさせる。
コピー元 <http://ja.wikipedia.org/wiki/ALOHAnet>
> これは衝突発生時に接続している全コンピュータにジャミング信号をブロードキャストし、各コンピュータに現在のパケットやフレームを捨てさせる。

メイル本文
> 伝送遅延が伝播遅延に比べて支配的な場合、ジャミング信号を使うと伝送媒体を素早く解放でき、多くのイーサネットに適している。
コピー元 <http://ja.wikipedia.org/wiki/ALOHAnet>
> 伝送遅延が伝播遅延に比べて支配的な場合、ジャミング信号を使うと伝送媒体を素早く解放でき、多くのイーサネットに適している。

メイル本文
> ALOHA は無線システムなので、小規模の LAN ではうまく機能するプロトコルでも常にうまく機能するとは限らないという問題もある (隠れ端末問題など)。
コピー元 <http://ja.wikipedia.org/wiki/ALOHAnet>
> ALOHA は無線システムなので、小規模の LAN ではうまく機能するプロトコルでも常にうまく機能するとは限らないという問題もある（隠れ端末問題など)。

メイル本文
> ハワイ諸島のネットワークの通信範囲は直径 400km だったが、伝播遅延はほぼ確実に伝送遅延に比べて小さかったため、プロトコルは十分な頑健性を備えている必要があった。
コピー元 <http://ja.wikipedia.org/wiki/ALOHAnet>
> ハワイ諸島のネットワークの通信範囲は直径 400km だったが、伝播遅延はほぼ確実に伝送遅延に比べて小さかったため、プロトコルは十分な頑健性を備えている必要があった。

図 18　警告メイル ボディにコピー疑いの箇所と疑いのある URI を記述する

6　結果と考察

開発したコピーレポート検知システムを用いて実際に一回生向けの講義で稼働テストを行った。テストを行った回数は全部で 3 回である。本システムの稼働を開始させたのは第 1 回の講義終了後である。また、第 3 回の講義の終了時にメイルによるアンケート調査を行った。

6.1　開発したシステムの検知率

実験では全部で 58 通のレポートメイルを処理した。メイルの提出数、コピーレポート数、システムによる検知数、検知率を**表 3** に示す。

表 3　開発したシステムの精度

講義の回	提出レポート数	コピーレポート数	システムによる検知数	システムによる検知率	誤検知数
第 1 回	20 通	12 通	9 通	75%	0 通
第 2 回	25 通	5 通	3 通	60%	1 通
第 3 回	13 通	3 通	3 通	100%	1 通

本研究で開発したコピーレポート検知システムの平均検知率は 78% であった。検知出来なかった原因を以下に記述する。

- 検索にかけるメイル文にコピー文以外の単語が多く混ざっており、コピー元の URI が抽出出来なかった。
- 検索にかけるメイル文が短すぎるため、コピー元の URI が抽出出来なかった。

6.2 コピーレポート数の推移

以下に第 1 回から第 3 回までのコピーレポートメイルの推移を**表 4** に示す。

表 4　コピーレポート数の推移

講義の回	提出レポート数	コピーレポート数	コピーレポートの割合
第 1 回	20 通	12 通	60%
第 2 回	25 通	5 通	20%
第 3 回	13 通	3 通	23%

システムを稼働させる前の第 1 回の講義に比べて、システムを稼働させた第 2 回以降の講義ではコピーレポート数の割合が 4 割下がっている。第 3 回の講義ではコピーレポートの割合に若干の上昇が見られたが、これはレポート提出数自体が少なかったためであり、実際に提出されたコピーレポートの数は下がっている。

6.3 アンケートによる調査

第 3 回の講義終了時に課題レポートの内容と併せてアンケートを行った。アンケートではコピーレポート検知システムに感じた印象と、学生の意識の変化を調査するために行った。

以下にアンケート内容を示し、アンケートの集計結果を**図 19**、**図 20** にそれぞれ示す。

- 質問 1, 課題のレポートで何通かのコピーレポートが検知され警告メイルが流れたが、それについてどう感じたか。
  - 1-1, コピーレポートは撲滅するべき、もっとやってほしい。
  - 1-2, うざい、やめてほしい。
  - 1-3, こんな機能に意味があるとは思えない。
  - 1-4, 特に何も感じない。
  - 1-5, そもそも警告メイルを見ていない。

- 質問 2, コピーレポートが検知されるようになって、自身がレポートを書く際に何か変わったことはあったか。
  - 2-1, Web からのコピーはしていないので、変わらない。
  - 2-2, Web からのコピーをやめ、自分の言葉でレポートを書くようになった。
  - 2-3, Web からのコピーはしているが、文章を改変したり、付け足したりと工夫するようになった。
  - 2-4, 特に気にすることなく Web からのコピーを続けている。
  - 2-5, そもそもレポートを出していない。

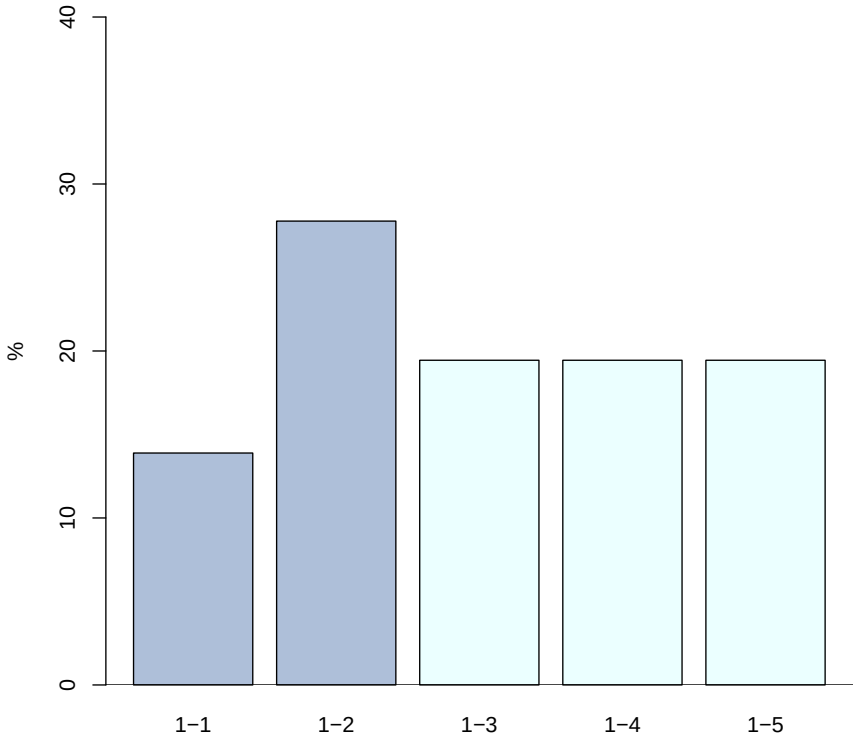

図 19　質問 1 コピーレポート検知システムに対する印象の質問への解答

集計結果を見ると、1-1、1-2 を選択した学生が合わせて 4 割を超えている。つまり、4 割以上の学生が開発したコピーレポート検知システムに何らかの関心を示したと言える。

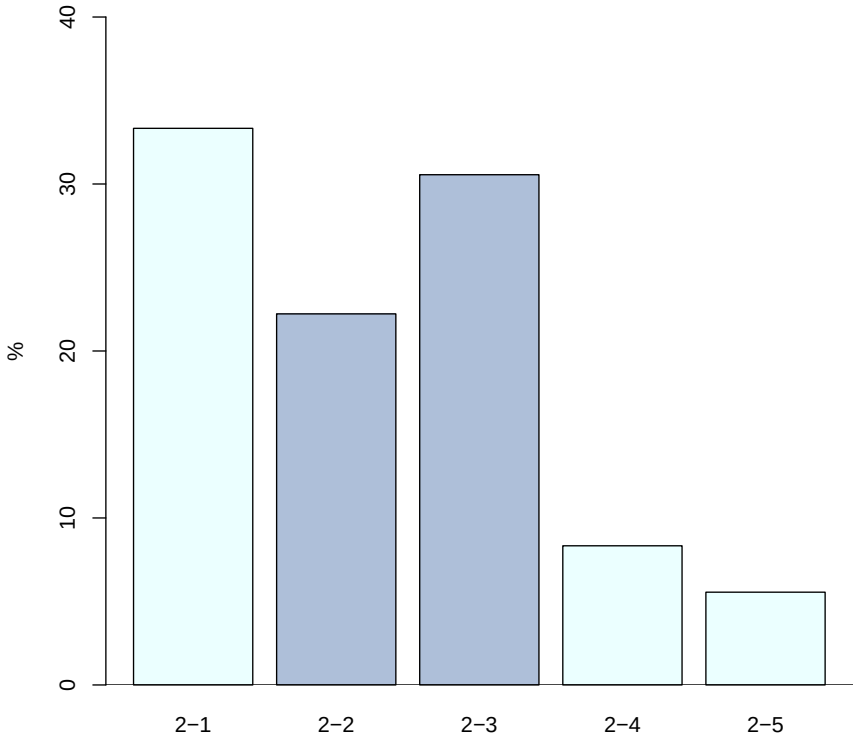

図 20　質問 2 システム稼働後の学生のレポートに対する意識の変化に関する質問への解答

集計結果を見ると質問 2-2、2-3 を選択した学生が合わせて 5 割を超えている。この結果より、コピーレポート検知システムを稼働させてから 5 割以上の学生がレポートの書き方を変えたと言える。

7　結論

本研究ではコピーレポート検知システムを開発し、一回生向けの講義で稼働させることにより、学生のレポートに対する意識の変化を検証した。システム稼働前は提出されたレポート数のうち 6 割がコピーレポートであったが、システム稼働後は、コピーレポートの割合が 2 割付近まで下がった。教員の負担という点でも、メイルの受信をトリガーに自動で起動し処理を行うため負担が少ないと言える。また、第 3 回の講義終了後にシステムの有効性を検証するため学生にアンケートを取った結果、4 割以上の学生がコピーレポート検知システムに何らかの関心を示し、5 割以上の学生がレポートの書き方を変えたという結果を示した。以上の結果により、本研究は有効であると言える。

8　今後の課題

8.1　比較速度の向上

本研究で開発したシステムはメイル本文と Web ページ本文を一文ずつ比較しコピー元の文章を割り出しているため、比較に時間がかかってしまう。比較速度の向上が必要である。

8.2　URI の抽出精度向上

検索に使用する文章が短すぎるとコピー元の URI が検索結果に出ないことがあった。検索に使用する文章部分の解析に改良の余地がある。

謝辞

本研究を行う上で細部にわたりご指導頂いた大垣 斉講師に深く感謝します。また、研究を進める上で助言を下さった東川 諒央氏、津川 翔丞氏、情報教育システム研究室のメンバーならびに卒業生の皆様に謝意を表します。

参考文献

[1] コピペルナー. http://www.ank.co.jp/works/products/copypelna/.

[2] *hanakawa.lab - 阪南大学. http://www2.hannan-u.ac.jp/~hanakawa/.

[3] 竹中康朝. コピーレポート検知システムの開発. 大阪産業大学, 2009.

[4] 河原 大輔黒橋禎夫. JUMAN version 5.1 - 思考と言語研究室, Sep 2005.

[5] 松本裕治, 北内啓, 山下達雄, 平野善隆, 浅原 正幸松田寛. 日本語形態素解析システム『茶筌』 version 2.2.1 使用説明書, Dec 2000.

[6] Mecab: Yet another part-of-speech and morphological analyzer. http://mecab.sourceforge.net/.

[7] w3m manual. http://homepage2.nifty.com/aito/w3m/MANUAL.html.

[8] Google soap search api. http://code.google.com/intl/ja/apis/soapsearch/api_faq.html#gen12.

[9] 圧縮プログラムでテキスト間の類似性を測る. http://djlab.sakura.ne.jp/mydiary/?p=2303.

[10] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. Language trees and zipping. *Phys. Rev. Lett.*, Vol. 88, p. 048702, Jan 2002.

[11] 安形輝. 異なるデータ長に対応したデータ圧縮による類似データ同定—近代日本文学データの著者推定による検証—. Technical Report 42(2005-FI-079), 亜細亜大学国際関係学部, may 2005.

[12] 安形輝. 圧縮プログラムを応用した著者推定. *Library and information science*, No. 54, pp. 1–18, 2005.

付録 A　ソースコード

```
#! /bin/sh
export LANG=ja_JP.UTF-8

OMAIL=/var/tmp/OMAIL.$$
HEAD=/var/tmp/HEAD.$$
HONBUN=/var/tmp/HONBUN.$$
SEARCH=/var/tmp/SEARCH.$$
URI=/var/tmp/URI.$$
W3M=/var/tmp/W3M.$$
OWEB=/var/tmp/OWEB.$$
OWEB2=/var/tmp/OWEB2.$$
WEBTXT=/var/tmp/WEBTXT.$$
ZIP=/var/tmp/ZIP.$$
MOZIHIKAKU=/var/tmp/MOZIHIKAKU.$$
HIKAKUBUN=/var/tmp/HIKAKUBUN.$$
HIKAKUKAISEKI=/var/tmp/HIKAKUKAISEKI.$$
MESSAGE=/var/tmp/MESSAGE.$$

trap "rm -f $OMAIL $HEAD $HONBUN $SEARCH $URI $W3M $OWEB $OWEB2 $WEBTXT $ZIP $MOZIHIKAKU $HIKAKUBUN $HI

#ヘッダ除去、文ごとに
func_HonbunGet ()
{
  nkf -e $1|sed '1,/^$/d'|sed -n -e '1,/^-- $/p'|\
  grep -v -- '-- '|tr -d '\n'|sed -e "s/。/。\n/g"|tr -d ' '
}


#URI 抽出、検索結果がない場合は知恵袋の URL しか抽出しないので省く
func_Uri ()
{
  wget -T 20 -q -O $SEARCH http://search.yahoo.co.jp/search?p=`echo $1`
  Uri=`cat $SEARCH|sed -e 's/>/>\n/g'|grep "^<a href="|sed -e "1d"|\
  sed -e 's/<a href=.//'|tr -d '">'|grep -v 'search.yahoo.co.jp'|\
  head -n 1`

  if [ $Uri != "http://chiebukuro.yahoo.co.jp/" ]; then
    echo $Uri
```

```
  else
    echo NoUri
  fi
}
```

```
#Web ページを取得
func_Text ()
{
  w3m -dump $1 > $W3M
  Zyogai=‘nkf -g $W3M‘
  nkf -e $W3M
  #sed ’/^\s*$/d’|tr -d ’ ’|tr -d ’\n’|sed -e "s/。/。\n/g"
}

#Chasen で解析
func_ChasenKaiseki ()
{
  chasen $1|egrep ’記号-句点|記号-アルファベット|名詞|助詞|動詞|形容詞|副詞’|\
  awk ’{print $1}’|tr -d ’\n’
}

#比較するテキストファイルを作成
func_Hikaku ()
{
  echo $1 > $HIKAKUBUN
  echo $2 >> $HIKAKUBUN
  func_Zip $HIKAKUBUN
}

#圧縮率取得
func_Zip ()
{
  zip -9 $ZIP "$1"|awk ’{print $4}’|tr -d ’%)’
}

#圧縮率から類似度を算出
func_Sabun ()
{
  MzWz=‘expr \( $1 + $2 \) / 2‘
  Ruizi=‘echo "scale=2; (1 - (($MzWz - $3) / $MzWz)) * 100"|bc‘
```

```
  echo $Ruizi|awk -F . '{print $1}'
}

#コピー判定
func_Hantei ()
{
  if [ $1 -ge 75 ]; then
    echo "メイル本文" >> $MESSAGE
    echo $Honbun2|sed -e "s/^/> /g" >> $MESSAGE
    echo "コピー元<‘echo $url|nkf -e‘>" >> $MESSAGE
    sed -n "$WebNum"p $OWEB2|sed -e "s/^/> /g" >> $MESSAGE
    echo "\n" >> $MESSAGE
    WebCopy=‘expr $WebCopy + 1‘
    break
  fi
}

func_Mail ()
{
  Line=‘wc -l $HONBUN|awk '{print $1}'‘
  Line2=‘expr $Line / 2‘
  cat $MESSAGE
  if [ $1 -gt 0 -a $1 -ge $Line2 ]; then

/usr/sbin/sendmail -t << END
From: ${MlName}-request@h-ps006.ise.osaka-sandai.ac.jp
To: ${MlName}@h-ps006.ise.osaka-sandai.ac.jp
Subject: WarningMail
In-Reply-To: ${Mid}
References: ${Mid}
Mime-Version: 1.0
Content-Type: text/plain; charaset=ISO-2022-JP
‘sed -e "1i 警告\n${Address}の書いたメイル\
\n> ${Mid}\n は本文の 5 割以上を Web からコピーした可能性があります。\
\n コピー疑いが認められた箇所は以下の通りです。\n" $MESSAGE|nkf -j‘
END

   fi
}

cat > $OMAIL

nkf -Z1 --overwrite $OMAIL
```

```
cat $OMAIL|sed -n -e '1,/^$/p' > $HEAD
```

#ヘッダからアドレス抽出

```
Address=`grep ^From: $HEAD|sed -e "s/.* //g"|tr -d "<"|tr -d ">"`
Mid=`cat $HEAD|grep -i "^Message-Id:"|head -1|awk '{print $2}'`
MlName=`grep ^To $HEAD|awk '{print $2 $3}'|sed  -e "s/@.*//g"|sed -e "s/.*<//g"`

func_HonbunGet $OMAIL > $HONBUN
```

#メイル本文から Web **ページ取得**

```
for ichibun in `cat $HONBUN`
do
  Kensaku=`echo $ichibun|sed "s/。//g"|sed "s/、//g"`
  func_Uri $Kensaku >> $URI
done

WebCopy=0
MailNum=1
for url in `cat $URI`
do
  func_Text $url > $OWEB

  if [ $Zyogai = "BINARY" ]; then
    MailNum=`expr $MailNum + 1`
  else
    func_ChasenKaiseki $OWEB|sed -e "s/。/。\n/g" > $WEBTXT
    sed '/^\s*$/d' $OWEB|tr -d ' '|tr -d '\n'|sed -e "s/。/。\n/g" > $OWEB2


    Honbun2=`sed -n "$MailNum"p $HONBUN`
    MailNum=`expr $MailNum + 1`

    echo $Honbun2 > $MOZIHIKAKU
    Mozisu=`func_ChasenKaiseki $MOZIHIKAKU|wc -c`
    MoziHigh=`expr $Mozisu \* 3`
    MoziLow=`expr $Mozisu / 2`

    MailBun=`func_ChasenKaiseki $MOZIHIKAKU`
    MailZip=`func_Hikaku $MailBun $MailBun`

    WebNum=1
    for web in `cat $WEBTXT`
```

```
    do
      WebMozi=‘echo $web|wc -c‘

      if [ $WebMozi -lt $MoziHigh -a $WebMozi -gt $MoziLow ]; then

        WebZip=‘func_Hikaku $web $web‘
        MailWebZip=‘func_Hikaku $web $MailBun‘

        Sabun=‘func_Sabun $MailZip $WebZip $MailWebZip‘
        func_Hantei $Sabun
      fi
      WebNum=‘expr $WebNum + 1‘
    done
  fi
done

func_Mail $WebCopy
```