情報処理学会論文誌　データベース　Vol. 2　No. 1　1–11 (Mar. 2009)

複製文字列検知に基づいたSplogフィルタリング手法

竹田 隆治[†1]　高須 淳宏[†2]

ブログなどのCGM（Consumer Generated Media）のデータは，消費者の実体験や生の声を含んでおり，顧客のニーズを分析したり，プロモーションの効果を検証したりするための情報源として，その重要性が増してきている．しかし，ブログには，商品の販売促進や，特定のwebサイトのランクをあげることなどを目的としたsplogと呼ばれるスパムコンテンツが含まれており，ブログの検索や分析に悪影響を及ぼしている．本稿では特に日本語におけるsplogの特徴であるコピーコンテンツの検出に注目し，そのフィルタリング手法を提案する．日本語のsplogは，さまざまな文書に含まれる文字列をコピーしつなぎ合わせることによって機械的に生成されることが多い．そこで，本稿では，動的計画法とsuffix arrayを用いて，各ブログに含まれる文字列で，他の文書にも現れる文字列を効率良く検出するアルゴリズムを提案し，そのような文字列がブログに占める割合に基づいたsplogのフィルタリング法を提案する．また，フィルタリング性能を評価するためのコーパスを構築し，提案手法が高いフィルタリング性能を実現できることを示すとともに，その特性を分析する．

Splog Filtering Method Based on Copy String Detection

Takeda Takaharu[†1] and Atsuhiro Takasu[†2]

CGM (Consumer Generated Media) data such as blog contains valuable information about customers reputation and it becomes important information source for detecting customers' needs and analyzing effects of various product promotion. However, CGM data contains spam content such as so called "splogs" that are generated for promoting products or improving rank of search results. They are harmful for CGM content retrieval and analysis. This paper proposes a splog filtering method based on the feature of Japanese splogs. The Japanese splogs are often generated by combining words and phrases appearing in various documents. This paper proposes an efficient copy string detection algorithm using the dynamic programming technique and suffix array and apply the proposed algorithm to calculate the ratio of copied strings in a blog. We construct an evaluation corpus for splog filters and show that the proposed method achieves high filtering performance using the corpus.

1. はじめに

ブログは個人の日記としての情報が記されており，商品やサービスに関する消費者の体験談や感想，生の声を収集するうえで有益なメディアとして注目を集めている[10)]．しかし一方でブログコンテンツの作成は容易で，商用利用を目的としたsplogと呼ばれるスパムコンテンツが増加している．splogは情報検索品質を低下させ，webアーカイブ資源を浪費させるという問題を引き起こしている．splogとは，スパム（spam）とブログ（blog）を合成した造語で，インターネットにおいて，リンク誘導や広告収入などの目的で生成される．本稿では，このような自動的に生成されるsplogのフィルタリング手法について論じる．

splogは他のコンテンツから語やフレーズをコピーし，それらをつなぎ合わせて生成されることが多く，splogの内容はその時々のニュースや流行によって変わるため，スパムメールフィルタで使われるベイズフィルタのような一般的な統計的手法で対処することが難しい．また，検索エンジンで上位の検索結果コンテンツ（webページ）からコピーしてくるなどの方法によって，splog自身も検索ランクの上位に上がりやすくしている．複数のコンテンツを部分的にコピーしてつなぎ合わせることで検索エンジンに「類似したページ」と認識されないようにしている．

本稿は，splogのこのような性質に着目し，ブログ中に現れる複数の文書に現れる文字列を検出し，そのような文字列の比率に基づいてsplogの判定を行う方法を提案する．

なお本稿では，単一URLが指し示すwebページ，単一ブログコンテンツのことを**ブログエントリ**と表記し，ブログエントリの集合を**ブログ**と表記する．また，あるブログエントリがsplogであることを**ブログエントリ（splog）**，splogではないことを**ブログエントリ（blog）**と記述する．

2. 関連研究

2.1 フィルタリング手法

splogのフィルタリング手法は，リンク解析による手法とコンテンツ解析に手法に大別できる．

†1 総合研究大学院大学
The Graduate University for Advanced Studies
†2 国立情報学研究所
National Institute of Informatics

Kolari らは，英語のブログを対象に splog フィルタリングの先駆的研究を行った[1)–3)]．特定種類の単語の割合を特徴ベクトルとして表現し，SVM（Support Vector Machines）などの機械学習の手法を用いて判別を行うコンテンツ解析手法を提案している．

日本語のブログを対象とした研究も始められているが，英語の splog フィルタリングのように文書を特徴ベクトルとして表現した分類問題とは異なるアプローチがとられている．成澤ら[6),12)]はコンテンツ中に頻出する文字列に着目した．ブログエントリ中の部分文字列の出現頻度と異なり数には一般的に Zipf の法則が成り立つが，splog 中には，検索クエリとしてよく使われる単語，最近話題の言葉，注目を集める言葉など，ジップの法則からはずれる語やフレーズが多く含まれる．そこで，これらの splog に特有な語やフレーズを抽出する方法を提案した．

一方，リンク解析を用いた手法として，石田[13)]は，いわゆるリンクファームと呼ばれる大量の不自然な被リンク群に注目した splog（群）の特定手法を提案した．Lin ら[4)]も基本的に同じアイデアでリンク構造からの splog 検出を試みている．

コンテンツ解析とリンク解析を併用した手法として，Lin ら[5)]は，ブログエントリの自己相関から splog を検出する手法を提案した．splog は，何回も繰り返し同じコンテンツを作り出すことが多いため，リンク先，タグ，タイトルなどの相関行列を構成し，これを特徴量として splog 特定を行っている．

URL の文字列に着目した splog フィルタリング法も提案されている．たとえば，www.dietthatworks.com というドメイン名は，diet that works の 3 つの単語を結合して作られている．Slabetti[7)]は，ドメイン名を構成する各単語の splog 率を定義し，ベイジアンフィルタを用いて分類する方法を提案した．筆者らの調査では，日本語の splog にも，“youtubemovie”，“affiliateinfo” といった単語の組合せによるドメイン名を使った splog が見受けられたが，“skdsjdljdks” や “g3n0x9h2ja9y1” といったランダムな文字列を用いたドメイン名の方が多かった．

一般に，英語ブログの研究では，正解データを用いた supervised な手法を用いることが多いのに対し，日本語ブログの研究では，unsupervised な手法を用いることが多い．

本稿で提案するフィルタリング手法は，コンテンツのコピー検出に基づいてフィルタリングを行うものであり，着眼点は成澤ら[6),12)]の手法に近い．成澤らの研究はスパムテンプレートの発見に焦点が当てられているのに対して，本研究ではコピー文字列の検知に焦点が当てられている．両者の違いについては，6 章で述べる．

2.2　評価用データ

筆者らが知る限り，splog フィルタリングの評価用データには，まだ，標準的なものが整備されておらず，splog フィルタリング研究にはさまざまなデータが用いられている．Lin らは[5)] TREC Blog Track 2006 を対象とし，ラベル付けした 9,200 件のブログエントリ中から，さらに 800 件ずつ blog と splog をサンプリングしている．Kolari ら[1)–3)]はテクノラティ[⋆1]の検索結果からデータセットを構築した．このデータセットも，blog と splog を同数ずつ作製している．データセット中の blog と splog の比率は，必ずしも実データの比率とは一致していない．

Salvetti ら[7)]は web crawler によって blog と splog それぞれ 10,000 件ずつ合計 20,000 件のデータセットを作成し評価を行った．石田[13)]は主要 CSP（Contents Service Provider）の RSS などから取得した 691,674 件中の，60 件を対象に実験を行った．

成澤らは，文献 12) で，人工的に作成したデータを用いて実験を行っている．その後，ブログではないが，yahoo 掲示板のコメントを用いた研究も行っている[6)]．本研究では，5 章で述べるように，日本の主要な CSP から一定期間ブログを収集し，人手によって splog の判定を行ったデータを用いて評価を行った．

3.　splog の分類

本稿では，佐藤ら[11)]と同様に，splog を

> 1 つ以上の何らかの他のコンテンツを部分的あるいはすべてコピーし，それらを連結して生成するブログエントリ

と考えてフィルタリング方法を構築する．特に，本研究では，splog 生成ツールが内部的に保持しているテンプレート，定型文，単語辞書など，明示的にその存在が明らかではない非公開のコンテンツを用いて作られた語やフレーズもコピーと考える．「今日は ... について調べてみました！」や「... の最新ニュースです！」といったフレーズがテンプレートの例にあたる．自動生成ツールは，その文章があたかも人間が書いたオリジナルコンテンツであるかのように見せかけるためにこのようなテンプレートを活用している．顔文字や慣用句など，一見しただけでは分からない巧妙なテンプレートも数多く存在する．

著者が分類した splog の種類を**表 1** にまとめる．各タイプの splog には，いずれも数多くの亜種が存在しており，加えて，**表 2** のような処理を施して生成されている場合が多い．

⋆1 http://technorati.com/

表 1　splog の分類
Table 1　Categories of splog.

news update	web ニュースの記事本文を すべてコピーして生成する． 記事本文に編集はまったく加えない
mail magazine	メールマガジンの記事本文を すべてコピーして生成する． メール本文に編集はまったく加えない
dictionary	wikipedia，などの百科事典コンテンツを そのままコピーして生成する．
QA	yahoo 知恵袋，はてな人力検索，などの 質問文（場合によっては回答文も） をそのままコピーする．
product induction	EC サイトの商品販売ページと 「機能レベルで」ほとんど同じであり そこから商品を直接購入もできる．
RSS	特定の RSS（Rich/RDF Site Summary， Really Simple Syndication） の内容をそのままコピーする 誰でもアクセス可能な RSS や splogger が設定した RSS の場合もある
search result	ある単語 (X) の検索結果コンテンツの 中身をコピーする． 単語 X は，自動投稿ツールが 持っている辞書の単語を使う場合と， 最近話題のキーワードを API で取得する場合とがある． 単語 X は複数の単語の場合もある
word salad	単語を無数に並べる． 自動投稿ツールが内部的に持っている 辞書の単語を使う場合と， 最近話題のキーワードを API で 取得する場合と， その両方を使う場合とがある．
full template	html ソースレベルで完全に同じ コンテンツを生成し続ける． FX，出会い系サイト， ロト 6，競馬，など 商材のセールステンプレートである．

表 2　複合的な splog 生成
Table 2　Composite splog generation.

combine	表 1 の生成方法を 2 種類以上組み合わせたもの
template decorator	表 1 の方法で生成したコンテンツに テンプレート文字列を付加する． たとえば「お早うございます」 「今日は……でした」「……をしています」など

今回のコーパス中にも表 2 や表 1 の full template に該当するようなテンプレートと思われる不自然に長い文字列の完全一致が頻繁にあり，このようなテンプレートの検出からも splog の検出を行うことができると期待できる[12)]．

4. splog フィルタリング手法

4.1 記号の定義

まず，本稿で用いる記号を定義する．文字列 $\boldsymbol{s}$ の i 番目の文字を s_i と表す．文字列 $\boldsymbol{s}$ の長さ i の接頭辞を $\boldsymbol{s}_{:i}$ と，また，$\boldsymbol{s}$ の i 番目の文字から始まる接尾辞を $\boldsymbol{s}_{i:}$ と表す．$\boldsymbol{s}_{i:j}$ は，$\boldsymbol{s}$ の i 番目から j 番目の部分文字列を表す．$|\boldsymbol{s}|$ は，$\boldsymbol{s}$ の長さを表す．

3 章に述べたように，ブログエントリ（splog）中の文字列は他のブログエントリにも出現していると思われる．この前提をもとに，ブログエントリの copy rate を定義する．copy rate は，そのブログエントリの部分文字列がどの程度他のコンテンツに存在しているかを示す値であり，提案手法はこの値に基づいて splog 判定を行う．

4.2 コピー検出法

単語レベルの短い文字列はブログエントリ（splog）にもブログエントリ（blog）にも頻繁に表れるため，そのような短い領域もコピーと見なすことは本稿の目的には合致しない．一方，1 文以上にわたる長い文字列が一致する場合は，コピーされている可能性が高いと思われる．そこで，提案手法では，一定長以上の文字列が複数ブログエントリに重複して出現する場合，その文字列はコピーされたと見なすことにする．この文字列長の下限を l とする．コピー検出のために用意されたブログを B とする．splog 判定を行うブログエントリ $\boldsymbol{b}$ の任意の部分文字列 $\boldsymbol{s}$ に対して，文字列 $\boldsymbol{s}$ を含む B 中のブログエントリ数を $df(\boldsymbol{s})$ で表すことにする．このとき，文字列 $\boldsymbol{s}$ のコピー長を以下のように定義する．

$$cpl(\boldsymbol{s}) = \begin{cases} |\boldsymbol{s}| \log \frac{|B|}{df(\boldsymbol{s})} & |\boldsymbol{s}| \geq l, df(\boldsymbol{s}) \geq 2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

提案手法では，上式に示されるように，情報検索などで用いられる IDF（Inverse DocumentFrequency）で重みづけられたコピー文字列長を用いる．

次にブログエントリのコピー長を定義する．以下では，ブログエントリを文字列として扱う．まず，ブログエントリ $\boldsymbol{b} = c_1 c_2 \cdots c_l$ を，以下に示すように n 個の部分文字列に分割することを考える．

$$\underbrace{c_1 \cdots c_{p_1}}_{\boldsymbol{b}_1} \underbrace{c_{p_1+1} \cdots c_{p_2}}_{\boldsymbol{b}_2} \cdots \underbrace{c_{p_{n-1}+1} \cdots c_l}_{\boldsymbol{b}_n} \quad (2)$$

以下では，このような分割を，分割によって得られる部分文字列のリスト

$$\boldsymbol{p} \equiv (\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots \boldsymbol{b}_n) \quad (3)$$

で表すことにする．

ブログエントリ $\boldsymbol{b}$ の分割 $\boldsymbol{p}$ によるコピー文字列長を以下に示すように，$\boldsymbol{p}$ によって得られる部分文字列のコピー文字列長の和と定義する．

$$bl(\boldsymbol{b}, \boldsymbol{p}) \equiv \sum_{i=1}^{n} cpl(\boldsymbol{b}_i) \quad (4)$$

最後に，ブログエントリのコピー文字列長を，分割によって得られるコピー文字列長の最大値と定義する．

$$bl^*(\boldsymbol{b}) \equiv \max_{\boldsymbol{p} \in P(b)} bl(\boldsymbol{b}, \boldsymbol{p}) \quad (5)$$

ここで，$P(\boldsymbol{b})$ は，ブログエントリ $\boldsymbol{b}$ の可能な分割の集合を表している．

4.3　コピー文字列長計算アルゴリズム

文字列 $\boldsymbol{s}$ の可能な分割は，$O(2^{|\boldsymbol{s}|})$ の数だけあるため，すべての分割を列挙して，式 (5) を解くことは計算時間の観点から現実的でない．文字列 $\boldsymbol{s}$ の任意の接頭辞 $\boldsymbol{s}_p$ に対して残りの接尾辞を $\boldsymbol{s}_s$ と表すことにする．すると，式 (5) は以下のように再帰的に表すことができる．

$$bl^*(\boldsymbol{s}) = \begin{cases} 0 & |\boldsymbol{s}| < l \\ \max_{\boldsymbol{s}_s} \{bl^*(\boldsymbol{s}_p) + cpl(\boldsymbol{s}_s)\} & \text{otherwise} \end{cases} \quad (6)$$

ここで，最大値は，ブログエントリ $\boldsymbol{s}$ の任意の接頭辞に対して求めるものとする．

input: ブログエントリ $\boldsymbol{b}$，最小コピー文字列長 l
　suffix array A
output: ブログエントリ $\boldsymbol{b}$ のコピー文字列長

```
begin
  set 0 to all components of C
  for i = l + 1 to |b| do
    for j = 1 to i − l
      s ← s(b_{j:i}, A), e ← e(b_{j:i}, A)
      C[i] ← max(C[j], cpl(s, f(s, e)) + C[j])
    end
  end
  return C[|b|]
end
```

図 1　コピー長計算アルゴリズム
Fig. 1　Algorithm for copy length calcuation.

式 (6) に基づいて，コピー文字列長を計算するためには，ブログエントリの任意の部分文字列 $\boldsymbol{s}$ に対して，その文書出現頻度 $df(\boldsymbol{s})$ を求める必要がある．$df(\boldsymbol{s})$ は，suffix array を用いることで効率的に求めることができる．suffix array は，コピー文字列の検出を行うための文書集合 B 中のすべての接尾辞を辞書順にソートし，接尾辞へのポインタを保持している．そのため，文字列 $\boldsymbol{s}$ を接頭辞として含む B 中の接尾辞は，array 上で連続した領域に表れる．suffix array 上のこの領域の先頭および末尾の位置をそれぞれ $s(\boldsymbol{s})$ および $e(\boldsymbol{s})$ と表すことにする．この領域の中の接尾辞を含むブログエントリ数を数えることによって，$df(\boldsymbol{s})$ を求めることができる．

図 1 にブログエントリのコピー文字列長を求めるアルゴリズムを示す．$s(\boldsymbol{b}_{j:i}, A)$ および $e(\boldsymbol{b}_{j:i}, A)$ は，それぞれ，suffix array A を 2 分探索し，部分文字列 $\boldsymbol{b}_{j:i}$ の A 上の開始位置と終了位置を求める関数を，また，$f(s, e)$ は，A 上の位置 s および e の間に含まれるブログエントリ数を求める関数を表している．

ブログエントリの任意の部分文字列 $\boldsymbol{b}_{j:i}$ に対して suffix array 中の開始位置 $s(\boldsymbol{b}_{j:i}, A)$ と終了位置 $e(\boldsymbol{b}_{j:i}, A)$ を求めるためには，

(1)　suffix array を 2 分探索し，array 上の該当する位置を 1 つ求める

(2)　その位置の前後に suffix array を走査し，開始位置と終了位置を求める

必要がある．2 分探索に $O(log|A|)$ 回，suffix array の走査に $e(\boldsymbol{b}_{j:i}, A) - s(\boldsymbol{b}_{j:i}, A) + 3$ 回の文字列比較が必要になる．

以下では，この回数を $g(\boldsymbol{b}_{j:i}, A)$ で表すことにする．ブログエントリ $\boldsymbol{b}$ の部分文字列は，$O(|\boldsymbol{b}|^2)$ 個あるため，2 分探索に $O(|\boldsymbol{b}|^2 \cdot log|A|)$ の文字列比較が必要になる．一方，suffix array の走査について，$O(\sum_{i,j} g(\boldsymbol{b}_{j:i}, A))$ 回の文字列比較を要する．以上より，ブログエントリ $\boldsymbol{b}$ のコピー長の計算に

$$O(|\boldsymbol{b}|^2 \cdot log|A| + \sum_{i,j} g(\boldsymbol{b}_{j:i}, A))$$

回の文字列比較を要する．比較する文字列長は $O(|\boldsymbol{b}|)$ となりうるが，ある程度の長さ以上の部分文字列を含むブログエントリ数は 1 となるため，それよりも長い部分文字列の処理は不要となる．そのため，実用上は文字列比較のコストは定数と考えることができる．なお，出現頻度の高い部分文字列については，毎回，suffix array 上を走査し，その部分文字列を含むブログエントリ数を計数するコストが高いため，現在の実装では，関係データベースに部分文字列とその頻度を登録して用いている．

5. 評価用データ

5.1 作 成 方 法

本研究で提案するフィルタリング法を評価するために，splog ベンチマークデータを作成した．データは，2008 年 1 月 31 日 18:00 から 2008 年 2 月 5 日 06:00 までの間，日本国内の大手ブログ CSP（Content Service Provider）からサンプリングした．各社が配信している新着ブログエントリの RSS フィードを常時監視し，サンプリング率 0.01 でランダムサンプリングを行った．

5.2 Labeled entries

次に，収集したブログ（web ページを）を人手で目視し splog/blog の判定を行い，計 21,668 件のラベル付きリストを作成した．まず，表 1 に示すように splog を分類し，それぞれの具体例を準備した．次に，3 名の作業者に分類と例を提示し，splog 判定を行わせた．判断がつかない場合は，同一 blogger の過去のブログエントリまで確認し，不自然な点がないかを確認したうえでラベリングを行った．**表 3** に，得られたラベル付きリストの CSP 別の blog，splog 数を示す．

splog は一見しただけでは自動生成コンテンツであるとは気づかれないように作られている．しかし，その特徴的傾向に注目すれば明らかに不自然な点が分かるようになる．その結果，過去のブログエントリを確認する必要なく，単一ブログエントリのテキストだけから判断がつくようになった．

表 3　CSP 別 splog 含有率
Table 3　Ratio of splogs for each CSP.

CSP	blog	splog
livedoor Blog	2,208	936
goo ブログ	1,297	119
LOVELOG	68	8
Yahoo!ブログ	1,935	59
アメーバブログ	4,029	306
JUGEM	1,144	523
ココログ	713	341
FC2 ブログ	762	1,155
ヤプログ！	1,616	40
Seesaa ブログ	148	1,065
はてなダイアリー	273	14
ウェブリブログ	325	19
teacup. ブログ	484	48
So-net blog	244	93
忍者ブログ	447	41
ドリコムブログ	45	0
楽天広場	950	98
AOL ダイアリー	44	1
Iza!	61	5
CURURU	3	0
Cnet	1	0
計	16,797	4,871

5.3 Unlabel entries

この 21,668 件以外に，splog/blog ラベルが付いていない同期間のブログを 50,000 件収集し，実験に用いた．

5.4 Search API

データ収集期間に話題となった語をサーチエンジンより収集した．単語の収集を行った期間は 2007 年 10 月 1 日から 2008 年 2 月 6 日までである．使用したサーチエンジンは以下のとおりである．

(1)　http://kizasi.jp/

(2)　Yahoo!検索ランキング

http://searchranking.yahoo.co.jp/

(3)　テクノラティジャパン：人気のブログ検索キーワード
http://feeds.technorati.jp/trjcf/keyword_ranking/

(4)　goo キーワードランキング
http://ranking.goo.ne.jp/keyword/

(5)　注目キーワード はてなダイアリー
http://d.hatena.ne.jp/hotkeyword

こうして収集した単語（13,733 語）を検索クエリとして検索 API で検索結果を収集した．検索を行った期間は 2008 年 2 月 6 日から 2008 年 2 月 8 日である．使用した検索 API は以下のとおりである．

(1)　livedoor ブログ検索
(2)　goo ブログ検索
(3)　Nifty @search
(4)　Namaan
(5)　Yahoo ブログ検索
(6)　Technorati ブログ検索
(7)　Google ブログ検索
(8)　エキサイトブログ
(9)　So-net ブログ検索

検索結果として最初のページに表示されるページのスニペットを収集した．

このような方法でコーパスを構築する理由は，splog 生成ツールが，このような方法で splog を生成していると予想されるからである．表 1 の splog の種類のうち，word salad, search result などはこのような手法で生成を行っていると思われるため，元となる文書を収集することでコピー検知を行う．

5.5　ブログエントリの本文抽出

ブログエントリ中の広告などを除いて本文のみを抽出するために，html のタグを用いた．ブログエントリ本文を表すタグは CSP ごとに異なるため，文献 8) と同様に，それぞれの CSP の本文タグを調べ，その中のテキストだけを取得することにより本文を抽出した．

6. 実 験 結 果

6.1　実験の概要

5 章で述べたラベル付きブログエントリすべてに対して，4 章で提案したアルゴリズムを用いて各ブログエントリのコピー文字列長を計算し，これに基づいて splog 判定を行った．つまり，コピー文字列長が *threshold* 以上のブログエントリを splog と見なし，*threshold* 未満の場合は blog と見なした．*threshold* を変化させることで precision，recall の値が変化し，precision-recall 曲線を描くことができるが，今回は実験の結果が多いため，各コーパス別に結果の F 値が最大となるときの precision，recall，F 値の比較を行った．コピー文字列を検知するためのコーパスとして，5 章で述べたコーパスそれぞれを用いた場合の結果を調べた．

- Labeled entries
- Unlabel entries
- Search API
- Unlabel entries+Search API

提案手法では，最小コピー文字列長 l を決める必要がある．実験では，この l の値として 1，3，5，7，10，12，15，20，25 を用いた．

6.2　評 価 指 標

splog フィルタリングの性能をはかる評価指標を以下に示す．

$$F\text{ 値} \equiv \frac{2\,r\,p}{r+p} \tag{7}$$

ここで，r および p は，以下に示す recall と precision を表している．

$$r \equiv \frac{\text{システム出力と正解ラベルが splog であると一致する数}}{\text{データ中のブログエントリ（splog）数}} \tag{8}$$

$$p \equiv \frac{\text{システム出力と正解ラベルが splog であると一致する数}}{\text{システムがブログエントリ（splog）であると出力した数}} \tag{9}$$

6.3　フィルタリング性能

図 2 に，最小コピー文字列長 l とフィルタリング性能の関係を示す．コピー文字列を検知するためのデータベースとして，6.1 節に述べた 4 種類のコーパスを用いた．グラフ中の F 値は，21,668 件のラベル付きブログエントリの F 値の平均値を表している．

図 2 に示されているように，最小コピー文字列長 l が短い場合，フィルタリングの性能は

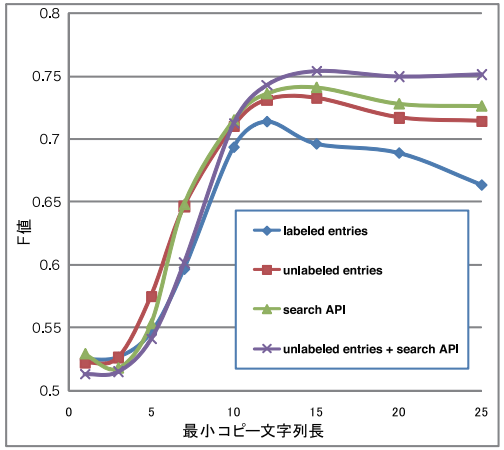

図 2　最小コピー文字列長とフィルタリング性能
Fig. 2　Filtering performance w.r.t. mimimum copy string length.

大きく劣化する．

この理由として，短い単語やフレーズはブログエントリ（blog）でも出現頻度が高いが，最小コピー文字列長が短い場合は，このような頻出単語やフレーズもコピー文字列と見なしてしまうことがあげられる．そのため，recall の上昇を超える precision の減少が生じ，F 値が下がってしまうものと思われる．

逆に l を 15 以上に設定した場合も，わずかではあるがフィルタリング性能が劣化している．これは，最小コピー文字列長以下のコピー文字列の検知ができなくなってしまうことに起因する．最小コピー文字列長を長くすることによって，precision を向上させる効果を得られるが，recall がそれ以上に減少してしまい，結果として F 値を下げてしまうものと思われる．

図 2 に示されるように，本実験で用いたすべてのコピー検出用コーパスにおいて，最小コピー文字列長が 14 文字から 16 文字のときに，提案手法は最も良いフィルタリング性能を示した．また，評価に用いたブログエントリがランダムにサンプリングされたものであることより，提案手法で用いる最小コピー文字列長は 15 文字程度が適しているものと思われる．図 6 に示すように，提案手法では precision が 0.9 まで 0.5 を超える recall を実現しており，高い precision に対して比較的高い recall を実現できるフィルタリング手法であるといえる．

表 4　閾値と precision，recall 値
Table 4　Precision and recall w.r.t. copy rate.

threshold	precision	recall	F
900	0.993	0.033	0.064
300	0.952	0.177	0.299
100	0.928	0.506	0.655
50	0.880	0.648	0.746
30	0.813	0.722	0.765
21.2	0.757	0.751	0.754
15	0.697	0.791	0.741
10	0.560	0.860	0.679
1	0.541	0.871	0.668

コーパスによる影響であるが，図 2 に示されるように，“Labeled entries” よりも “Unlabel entries” や “Search API” の方が平均的に少し高い精度を示している．また “Unlabel entries” よりも “Search API” の方がわずかに精度が上である．しかし，今回の実験では，コーパスによるフィルタリング性能の変化はわずかであった．

表 4 に *threshold* を変化させたときの precision と recall の値を示す．ここでは，コーパスに Unlabel entries+Search API を用い，最小コピー文字列長 l が 15 のときの値を示した．一般に，コピー率が大きい場合に precision が高くなる．顕著なブログエントリ（splog）にはそれだけコピー文字列が多いということを意味している．しかし同時に recall が低下することから，実験で用いた規模のコーパスではコピー検出できな文字列を含んだブログエントリ（splog）も多く存在するものと思われる．

表 5 に図 2 の各測定点における precision と recall の値を示す．最小コピー文字列長が長い場合，コピー率の大きさは，precision および recall の値にほとんど影響を及ぼさず，最小コピー文字列長以上の長さの文字列が他のブログエントリにも存在するかどうかが重要な指標となる．表 5 に示されるように，各コーパスにおいて，最小コピー文字列長が長いときに，最も高い precision が得られる．これは，十分に長い文字列が他のブログエントリにも存在する場合は，その文字列を含むブログエントリは，splog の可能性が高いということを意味している．

図 3 には，コピー検知に用いるデータの大きさとフィルタリング性能の関係を示す．図は，最小コピー文字列長が 12，15，20，25 の場合の性能の変化を示している．コーパスを大規模にすることが splog フィルタリングに有効であるが[9]，今回の実験では，コーパスの

表 5　最小コピー文字列長に対する最適な F 値と recall，precision
Table 5　F-value, recall and precision w.r.t. minimum copy string length.

Labeled entries				
l	threshold	precision	recall	F
1	1,600	0.525	0.524	0.524
3	1,150	0.524	0.529	0.526
5	540	0.546	0.545	0.546
7	200	0.601	0.592	0.596
10	40	0.688	0.699	0.693
12	19	0.748	0.682	0.713
15	9.3	0.752	0.648	0.696
20	0.01	0.735	0.648	0.689
25	0.01	0.816	0.558	0.663

Search API				
l	threshold	precision	recall	F
1	1,480	0.529	0.529	0.529
3	1,265	0.520	0.515	0.517
5	650	0.549	0.556	0.553
7	180	0.650	0.645	0.648
10	69	0.718	0.711	0.715
12	29	0.731	0.740	0.736
15	13	0.735	0.746	0.740
20	7	0.724	0.731	0.727
25	0.01	0.809	0.658	0.726

Unlabel entries				
l	threshold	precision	recall	F
1	1,630	0.521	0.523	0.522
3	1,250	0.524	0.529	0.526
5	410	0.572	0.578	0.575
7	165	0.645	0.648	0.646
10	65	0.718	0.703	0.710
12	29	0.733	0.729	0.731
15	10.5	0.743	0.722	0.732
20	8.5	0.730	0.704	0.717
25	0.01	0.809	0.639	0.714

Unlabel entries+Search API				
l	threshold	precision	recall	F
1	1,400	0.511	0.514	0.513
3	1,115	0.516	0.514	0.515
5	800	0.534	0.547	0.541
7	385	0.601	0.602	0.601
10	107	0.711	0.713	0.712
12	50	0.745	0.740	0.743
15	21.2	0.757	0.751	0.754
20	10.8	0.792	0.711	0.750
25	0.01	0.772	0.731	0.751

規模が，60 MB を超えたあたりから性能向上の鈍化がみられた．頻度の高いコピー文字列は比較的小規模のコーパスでも検知することができる，頻度の低いコピー文字列を検出するためには，コーパスサイズを増加させるとともに，コピー文字列検出に効果的な情報源を見つける必要がある．

6.4　処 理 性 能

提案手法の処理性能を調べるために，さまざまな最小コピー文字列長およびコピー検知用のデータベースサイズに対する，21,668 件のラベル付きブログエントリの処理時間を計測した．

図 4 は，最小コピー文字列長を変化させたときの 1 ブログエントリあたりの平均実行時間を示している．このグラフに示されるように，最小コピー文字列長が短いほど処理時間が長くなる．短い文字列を接頭辞とする接尾辞は多数あるため，suffix array 上で文字列を接頭辞として持つ接尾辞の範囲を検索した後に，その範囲の接尾辞を走査して出現頻度を計数する時間が増加するためである．

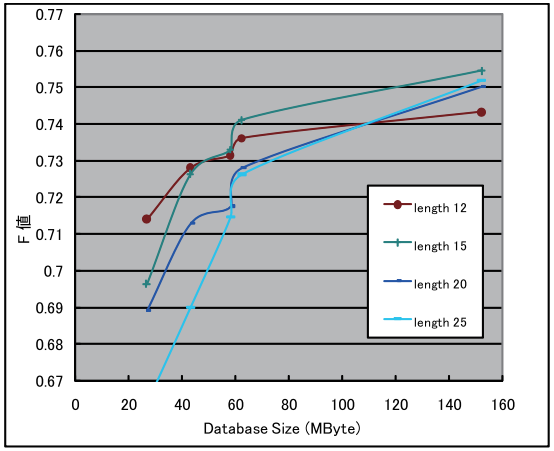

図 3　データベースサイズとフィルタリング性能
Fig. 3　Filtering performance w.r.t. database size.

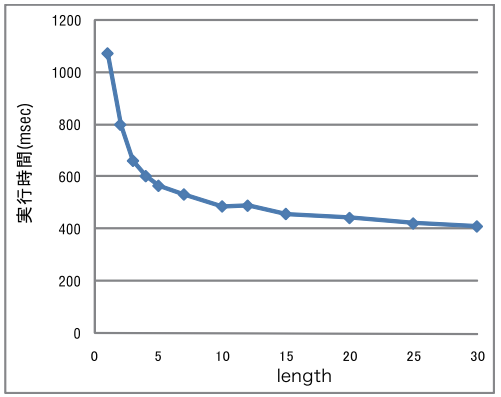

図 4　最小コピー文字列長と処理時間
Fig. 4　Processing time w.r.t. minimum copy string length.

図 5 は，最小コピー文字列長が 1 で処理時間が最もかかる場合の各ブログエントリの処理時間を示している．4 章で提案したアルゴリズムは，ブログエントリ長 $|b|$ に対して，suffix array を $O(|b|^2)$ 回検索する必要があるが，図 5 は処理時間が $O(|b|)$ に近いことを示してい

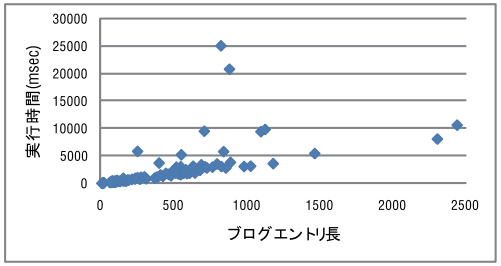

図 5　ブログエントリの長さと処理時間
Fig. 5　Processing time w.r.t. blog entrh length.

る．すでに述べたとおり，短い文字列の出現頻度を求めるためには時間がかかるが，一方，長い文字列を接頭辞とする接尾辞は少なく，高速に出現頻度を求めることができる．

6.3 節に述べたように，提案手法では，最適と思われる最小コピー文字列長 l は 15 程度となる．この場合，図 4 に示すように，$l = 1$ のときに比べてかなり高速に splog を検出できる．本実験では，25 Mbyte 程度のコピー検知コーパスを用いて，21,668 件分のブログエントリの splog 判定を行うのに，suffix array の構築から splog 検知まで一連の処理に 3 時間程度で実行できた．

6.5　スパムテンプレート検出法との比較

splog フィルタリングには，splog のさまざまな特徴を用いることが考えられる．たとえば，平均的にブログエントリ（blog）よりもブログエントリ（splog）の方がコンテンツ長が長く，本研究で作成したコーパスでは，ブログエントリ（blog）の平均の長さは 488.3 文字に対して，ブログエントリ（splog）の平均の長さは 2,263.3 文字であった．そこで，ブログエントリのコンテンツ長を特徴量とするフィルタリング法との比較を試みた．しかし，この方法のフィルタリング性能は，提案手法よりもかなり劣っていた．

そこで，本稿では，成澤ら[12] が提案したスパムテンプレート検出法を用いたフィルタリング法との比較を行う．まず，文献 12) の方法に基づいて，下記の手順でスパムテンプレートを検出する．

(1)　コーパスの Suffix Array を構築し部分文字列の数え上げによってジップ則に反する最も出現頻度の高い文字列を検出する．この出現頻度を f_s とする．
(2)　出現頻度 f_s の部分文字列の中で最長の文字列を検出する．この文字列を $\boldsymbol{s}$ とする．
(3)　$\boldsymbol{s}$ をコーパスから除去するとともに，スパムテンプレートとして保存する．

表 6　スパムテンプレート検出によるフィルタリング
Table 6　Filtering perfomance of spam template detection method.

Template 数	precision	recall	F
100	0.997	0.0823	0.152
110	0.993	0.0907	0.166
120	0.993	0.0907	0.166
130	0.993	0.0963	0.175
140	0.921	0.121	0.214
150	0.563	0.211	0.306
160	0.543	0.313	0.398
170	0.508	0.353	0.417
180	0.468	0.387	0.424
190	0.422	0.415	0.418
200	0.397	0.452	0.423
220	0.366	0.463	0.409
240	0.337	0.498	0.402
260	0.306	0.545	0.391
280	0.285	0.596	0.386
300	0.267	0.637	0.376
320	0.258	0.700	0.377
350	0.250	0.792	0.380
400	0.244	0.901	0.391

(4)　(1) に戻る．

この操作を N 回繰り返し，N 種類のスパムテンプレートを検出した．このようにして得られたスパムテンプレートを 1 つでも含むブログエントリを splog と判定した．検出するスパムテンプレートの数とフィルタリングの性能の関係を**表 6** に示す．表に示されているように，スパムテンプレート数の増加にともなって，フィルタリング性能も向上するが，テンプレート数 170 くらいで，性能の改善がほとんど得られなくなる．これ以上検出回数を増やすと「よろしくお願いします」「ありがとうございました」など，スパムテンプレートではない定型的文句が出てくるようになり，フィルタリング精度の劣化が起こるためである．

提案手法では，図 2 や図 3 に示すように最大で 0.75 程度の F 値が得られており，本稿で行った実験の範囲では，提案手法のほうが優れたフィルタリング性能を示した．

スパムテンプレートに基づく splog フィルタリングは，表 6 に示されているように高い presicion を実現している．このことから，顕著なスパムテンプレートは正確に発見できていることが予想される．一方で，十分な大きさで同一テンプレートを持ったブログエントリ（splog）がコーパス中になければならないため，コーパスに現れない多種多様なテンプレー

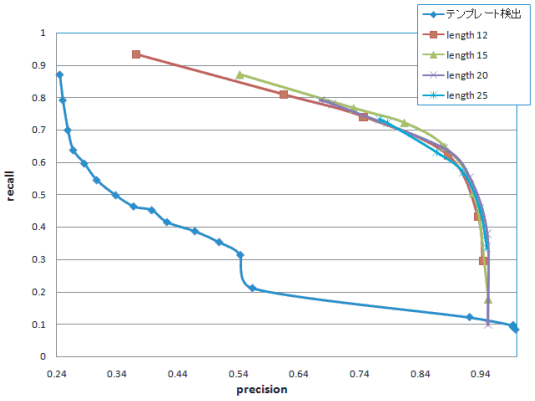

図 6　テンプレート検出との比較
Fig. 6　Performance comparison between proposed method and template detection method.

トの検出が難しい．つまり，出現頻度の小さいテンプレートの発見ができないため，十分な recall は実現できず，高い F 値を達成できないと考えられる．**図 6** に示すように，提案手法ではテンプレート検出に比べて高い recall を実現している．

本手法との主な違いは，テンプレート検出法が，文字列のコーパス中での出現回数に着目しているのに対し，提案手法では，出現回数に加え文字列の長さにも着目しているところにある．

なお，成澤ら[12] の研究は，スパムテンプレートを検出することを主たる目的としており，本実験で splog フィルタリングへ適用するあたって，チューニングなどは行わなかった．本実験では，splog の判定においては，スパムテンプレートを 1 つでも含むブログエントリは splog とする，という単純な判定方法を用いたが，判定方法を工夫することによって，より良い性能を引き出せる可能性もあると考えている．

7. おわりに

本稿ではコピー文字列検知に基づいた splog フィルタリング手法を提案した．提案手法は，IDF に基づいた重み付きコピー文字列長を導入し，そのコピー率に基づいて，splog 判定を行う．重み付きコピー文字列長を効率良く計算するために，suffix array と動的計画法を用いたアルゴリズムを提案し，また，splog フィルタリングの性能を評価するためのコーパスを作成して，提案手法の性能を評価した．

提案手法によって一定のフィルタリング性能を実現できたが，提案手法を実データに適用するためには，精度と処理性能の両面でさらなる改善が望まれる．現在，筆者らは，提案手法を用いた splog フィルタリングシステムを実データに適用すべく実装を進めている．

実データに提案手法を適用する場合，ブログエントリが日々増加するため，コピー検知のためのブログデータベースを逐次更新していく必要が生じる．データ量も膨大になるため，suffix array の更新性能と更新頻度が課題になる．そこで，更新を高速化するたに並列化をはかるとともに，suffix array の適切な更新頻度を定める必要がある．図 3 に示されるデータベースのサイズとフィルタリング性能の実験結果より，提案手法では，ある程度の規模以上のデータに対し，フィルタリング性能の変化はわずかになる．そのため，いったん大規模なデータベースを構築すれば，その更新頻度はそれほど高くする必要はないと予想している．しかし，提案手法は，流行語などの最新の情報を積極的に用いるメカニズムを備えていないので，時間情報を考慮する場合は，データベースの更新頻度についてもさらに検討が必要と考えている．

3 章の表 1 に示したように，splog には，さまざまなタイプがあり，タイプごとに特性が異なる．提案手法は，十分に長い文字列一致がある news update，full template などに有効であるが，短い文字列を組み合わせる word salad タイプの splog の検知にはあまり適していない．そこで，タイプ別にフィルタを開発し併用することも精度の高いフィルタを作成するうえで有効であると考えられる．

4.2 節では，式 (1) に示すように，部分文字列のコピー長として IDF で重みづけされた値を用いたが，これ以外にも文字列が他の情報源からコピーされた度合いを表す指標としてさまざまなものが考えられる．たとえば，Narisawa らは，自然な文字列の出現頻度は Zipf 則に従うと仮定し，そこからの解離の度合いをスパムテンプレートの検出に用いたが[6]，同様の指標をコピーの度合いとして考えることができる．今後，さまざまな指標を式 (1) のコピー長に組み込むことで，splog フィルタリングに適した指標を見つけたいと考えている．

参 考 文 献

1) Kolari, P., Finin, T. and Joshi, A.: SVMs for the blogosphere: Blog identification and splog detection, *AAAI Spring Symposium on Computational Approaches to Analyzing Weblogs* (2006).
2) Kolari, P., Java, A. and Finin, T.: Characterizing the Splogosphere, *Annual Workshop on Weblogging Ecosystem: Aggregation, Analysis and Dynamics* (2006).
3) Kolari, P., Java, A., Finin, T., Oates, T. and Joshi, A.: Detecting Spam Blogs:

A Machine Learning Approach, *21st National Conference on Artificial Intelligence* (*AAAI 2006*) (2006).
4) Lin, Y.R., et al.: The splog detection task and a solution based on temporal and link properties, *Proc. 15th Text REtrieval Conference* (*TREC'06*) (2006).
5) Lin, Y.R., Sundaram, H., Chi, Y., Tatemura, J. and Tseng, B.L.: Splog detection using self-similarity analysis on blog temporal dynamics, *3rd Intl. Workshop on Adversarial Information Retrieval on the Web*, pp.1–8 (2007).
6) Narisawa, K., Inenaga, S., Bannai, H. and Takeda, T.: Efficient Computation of Substring Equivalence Classes with Suffix Arrays, *18th Annual Symposium on Combinatorial Pattern Matching* (*CPM'07*), pp.340–351 (2007).
7) Salvetti, F. and Nicolov, N.: Weblog classification for fast splog filtering: A url language model segmentation approach, *Proc. Human Language Technology Conference of the NAACL*, pp.137–140 (2006).
8) Takeda, T. and Takasu, T.: UpdateNews: a news clustering and summarization system using efficient text processing, *Intl. Conf. on Digital Libraries* (*JCDL 2007*), pp.438–439 (2007).
9) Takeda, T. and Takasu, T.: A Spam Blog Filitering Method Based on Text Copy Detection, *The 1st IEEE Intl. Conf. on the Applications of Digital Information and Web Technologie*, pp.543–548 (2008).
10) 奥村　学：blog マイニング：インターネット上のトレンド，意見分析を目指して，人工知能学会誌，Vol.21, No.4, pp.424–429 (2006).
11) 佐藤有記，宇津呂武仁，福原知宏，河田容英，村上嘉陽，中川裕志，神門典子：キーワードの時系列特性を利用したスパムブログの収集・類型化・データセット作成，*DEWS2008* (2008).
12) 成澤和志，山田泰寛，池田大輔：分文字列の数え上げによるブログスパムの検出，情報学基礎研究会報告，Vol.2006, No.59，pp.45–52 (2006).
13) 石田和成：スパムブログの定量的調査と分離の試み，データベースと Web 情報システムに関するシンポジウム（DBWeb2007）(2007).

(平成 20 年 9 月 20 日受付)
(平成 21 年 1 月 7 日採録)

(担当編集委員　奥村 学)

竹田　隆治（学生会員）
平成 16 年会津大学大学院コンピュータ理工学研究科修了．現在，総合研究大学院大学複合科学研究科情報学専攻博士課程在学中．電子情報通信学会会員．

高須　淳宏（正会員）
昭和 59 年東京大学工学部航空学科卒業．平成元年同大学大学院工学系研究科博士課程修了．工学博士．同年学術情報センター研究開発部助手．同センター助教授．国立情報学研究所助教授を経て平成 15 年より同研究所教授．データ工学，特にデータ解析と解析モデルの学習の研究に従事．電子情報通信学会，人工知能学会，日本データベース学会，ACM，IEEE 各会員．