

blogの自動収集と監視

南野 朋之[†] 鈴木 泰裕[†] 藤木 稔明[†] 奥村 学[‡]

概要

近年注目され始めている情報源として blog (Weblog) がある。現在, blog というと, blog ツールと呼ばれる管理ツールを使用して作成される Web ページを指すことが多いが, 日本では blog ツール登場以前から, Web 日記という形で個人による情報発信が行われており, 非常に有用な情報源となっている。そこで本研究では, このような Web 日記も含めて blog と呼び, 特定のツールやメタデータに依存しない, HTML 文書の解析に基づいた手法で, これら個人の発信する時系列に沿って掲載される情報を網羅的に収集, 監視するシステムを提案する。

Automatically Collecting and Monitoring Japanese Weblogs

Tomoyuki NANNO[†] Yasuhiro SUZUKI[†] Toshiaki FUJIKI[†] Manabu OKUMURA[‡]

Abstract

We present a system that tries to automatically collect and monitor Japanese blog collections that include not only ones made with blog softwares but also ones written as normal web pages. Our approach is based on extraction of date expressions and analysis of HTML documents. Our system also extracts and mines useful information from the collected blog pages.

1 はじめに

インターネットの普及に伴い, 一般の多くの人々からの情報発信が盛んになり, その発信されている大量の情報を有効に活用したいという要求も高まっている。こうした状況を背景に, 現在注目されている情報源の一つが掲示板 (BBS) であり, 掲示板を定期的に監視し, そこから情報を抽出, 発掘することで, 一般大衆の「生の声」を製品開発, 企業活動に反映しようという試みも見られる [1]。

同様に近年注目され始めている情報源として blog (Weblog) がある。blog の定義は現在必ずしも定まっているとは言えないが, Web 上の「日記サイト」あるいは「個人ニュースサイト」と言うことができ, 書き手が関心を持ったニュースやできごとについて (何らかのコメントを) 書いた記事を, 元情報へのリンクとともに時系列に沿って掲載しているサイトを指すことが多い。通常の Web ページとは異なり, 速報性, リアルタイム性のある新鮮な情報が発信されることから, 掲示板同様有用な情報源と考えられるようになってきている。

現在「blog」というと, Movable Type¹などの「blog ツール」や, tDiary²などの「Web 日記システム」, ま

た, これらのホスティングサービスを利用して書かれ, Trackback[2] など, blog ツール特有の機能を有しているサイトを指すことが多い。

これらのツールを使用した blog の収集は, RSS(RDF Site Summary)[3] などのメタデータを利用したり, Web ページの構成がある程度一定であるという特徴を利用したり, さらには, ping.bloggers.jp³などをはじめとする, XML-RPC を使用した ping[4] による blog 更新通知サービスを利用することで, 比較的容易に行うことが可能である。

しかしながら, 日本ではこうした blog ツール登場以前から「Web 日記」あるいは「テキストサイト」という形で, 個人による情報発信が行われており, 非常に大きなコミュニティを形成している [5]。これらの情報も blog と同様, 注目すべき情報であるが, その多くがツールやホスティングサービスを利用しない, 個人が管理する Web ページ中に存在するため, 網羅的に収集することはそれほど容易ではない。

そこで本研究では, このような Web 日記も含めた, 時系列に沿ってなんらかの記述を掲載しているサイトを blog と定義する。そして, 特定のツールやメタデータに依存しない, HTML 文書の解析に基づいた手法で, これら個人の発信する時系列に沿って掲載される情報を網羅的に収集, 監視するシステムを提案する。

[†]東京工業大学大学院 総合理工学研究科
Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology
{nanno,yasu,fujiki}@r.pi.titech.ac.jp

[‡]東京工業大学 精密工学研究所
Precision and Intelligence Laboratory,
Tokyo Institute of Technology
oku@pi.titech.ac.jp

¹movabletype.org <http://www.movabletype.org/>

²tDiary.org <http://www.tdiary.org/>

³ping.bloggers.jp <http://ping.bloggers.jp/>

2 関連研究

blog を収集し、検索機能を提供しているサービスはいくつか存在する。

Bulkfeeds⁴は日本国内で RSS 配信されている情報を検索することのできる検索エンジンである。RSS の収集は以下の三種類の方法で行われている。

- ping サーバの更新情報
- blog ホスティングサービスの更新情報
- 人手による登録

また、blog のみに的を絞った検索エンジンとして、Feedback⁵や Myblog japan⁶、ココログ⁷、BLOGNAVI⁸、blog search!⁹などがある。これらは、人手による収集、あるいは ping サーバより blog サイトの情報を得て、RSS を利用することで、blog を収集する検索システムである。

これに対し、本研究では、クローリングした HTML 文書を解析し、その Web ページが blog であるかを判定することによって blog の収集を行う。この手法の利点は、以下の 2 点である。

- blog ツールなど、特定のシステムを利用していない Web 日記なども広く収集することが可能
- RSS は、直前何回かの更新に対するメタデータであることが多いため、RSS ベースの収集では過去の記事を収集できないのに対し、HTML 文書に基づいた収集では、HTML 文書を直接解析することで、過去のものまで収集することが可能

また、海外では、Technorati¹⁰や Blogdex¹¹ などが RSS に基づいた検索サービスや、含まれるリンクを利用したニュースソースのランキングなどを行っている。

3 システムの概要

blog の収集に必要なタスクは以下の二つである。

1. Web ページ集合から blog ページを選択する
2. 選択された blog ページから一日分の記事 (以下、entry と呼ぶ) の集合を切り出す

blog から entry を切り出す理由は、検索単位をページ単位ではなく、一日分の記事にするためである。このように entry を抽出することで、キーワード検索以外に、日付をキーとした検索が可能となり、さらにマイニングを行うことで、「ある期間で話題になったキーワード」などを抽出することが可能になる [6]。

図 1 に、本研究で構築したシステムのフローチャートを示す。

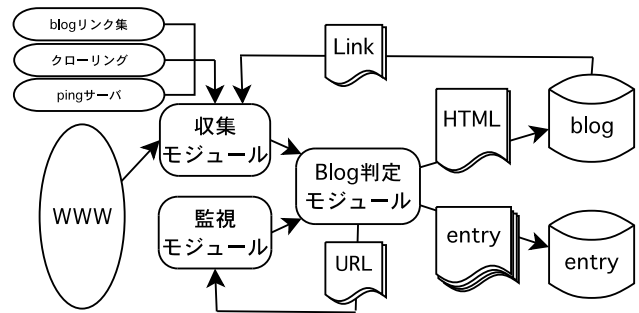


図 1: システムのフローチャート

収集モジュールは、以下の 3 つの方法で Web ページのクローリングを行う。

1. WWW 全体を対象とした収集
2. blog のリンク集や、更新通知サービスを利用した収集
3. blog と判定されたページに含まれるリンクを利用した収集

収集モジュールが収集した Web ページに対して、blog 判定モジュールはそのページが blog であるかどうかを判定し、blog であると判定されたページから、entry を抽出する。監視モジュールは、判定モジュールによって blog と判定された URL に対して、定期的な監視を行い、新たに追加された entry の取得を行う。

次節では、blog 判定モジュールを中心に詳説する。

4 blog 判定モジュール

本研究では、ある Web ページが blog かどうかを直接判定することが難しいため、“Web ページから、ある制約を満たす entry を切り出すことが出来るかどうか”をチェックすることで blog 判定を行う。なお、blog ページの選択および entry の抽出を行う際には、blog の性質として以下の性質を仮定している。

- entry の書かれた日付を表す日付表現を必ず含む
- 日付表現は記事の上部にある
- 一連の entry に対して、各日付表現にかかるタグの種類は一定である
- 一連の日付表現はフォーマットが一定である (ex.) “2003/1/2” と “2-Jan-2003” は別のフォーマット

4.1 前処理

取得した Web ページに対して HTML Tidy [7] を適用し、HTML 文書を well-formed XML 文書にする。この処理により、開始タグと終了タグのバランスが取れていることが保証される。よって、以降の処理では、日付を含む部分のタグの係り方が一定であるという制約を用いる。

⁴Bulkfeeds <http://bulkfeeds.net/>

⁵Feedback <http://naoya.dyndns.org/feedback/>

⁶Myblog japan <http://www.myblog.jp/>

⁷ココログ <http://web.or.tv/>

⁸BLOGNAVI <http://www.blognavi.com/>

⁹blog search! <http://blog.threetree.jp/>

¹⁰Technorati <http://www.technorati.com>

¹¹Blogdex <http://blogdex.media.mit.edu>

表 1: 日付フォーマットの例

2004 年 3 月 5 日	2004. 3. 5	2004/3/5
2004-3-5	2004 03 05	3 月 5 日
March 5	5 Mar. 2004	5 March 2004.
5-March-2004	March 5 2004	3. 5 2004

4.2 日付表現の抽出

次に、前処理を行った HTML 文書から、日付表現を抽出する。日付表現の抽出は、以下の 2 つのステップからなる。

1. 日付表現候補個所の抽出
2. 不足している情報の補完

以下、各ステップについて詳細に述べる。

日付候補個所の抽出

まず、日付候補個所を抽出する。候補個所は年だけのもの、月だけのもの、日だけのものや、年と月だけ、月と日だけのものを含む。

日付の表記法には様々なものがある。区切り文字には、年月日、ハイフン、スラッシュなどが使用されることもあれば、月が英語名で記述されていたり、「平成」のような年号をつけて記述されていることもある。これら一般的に使われているフォーマットを 23 種のタイプに分類し、それぞれを表す正規表現と入力 HTML 文書でパターンマッチを行うことで、日付候補箇所を抽出する。表 1 に、23 種の正規表現にマッチする日付表現の一部を示す。

また、以下のように明らかに entry の日付を示さないような日付は、候補箇所から除外する。

- 日付の範囲を示す表現
ex.) 2004 年 3 月 4 日, 5 日
- 文中に含まれる日付表現
ex.) 自然言語処理研究会は 2004 年 3 月 4 日から行われる。

不足している情報の補完

blog では、最上部に年が記載され、各 entry の日付では年を省略し、月日のみを記載することがある。このような場合、不完全な日付表現に対して、不足している情報を補完することを試みる。(ただし、ここで補完するのは年と月だけであり、日の補完は行わない。)

補完すべき年月の推定にはいくつかの手がかりを使用することができるが、信頼度の高さを考慮し、以下に示す優先順位でルールを適用する。

1. 現在の日付表現より前に出現し、かつ、現在の日付表現の DOM ツリー中での深さと同じもしくは浅い、もっとも近い位置にある日付表現を使用して補完

2. 現在の日付表現より前に出現し、かつ DOM ツリーを深さ優先探索した場合、もっとも近い位置にある日付を使用して補完
3. Web サーバが通知する last-modified¹² があればそれを、無ければ取得日時を使用して補完。(ただしこのルールでは月は補完しない。)

また、年が 2 桁で表現されている場合、以下の二通りの解釈が可能である。

- 上位二桁 (19 もしくは 20) が省略されている
- 年号 (例えば、昭和や平成) が省略されている

このような日付表現に対しては、以下のヒューリスティックを用いて、補完を行う。

- 年が 64 以下の場合
この場合、年号だけ欠けている可能性が高いと考える。よって、以前に年号がついた日付候補が出現していれば、それが省略されたものとみなし、西暦に変換する。ただし、この変換の結果、直前に出現した日付候補の年の値との差が 10 年以内であることを条件とする。
- 年が二桁の数字 (03 など二桁と考える) で、かつ以前に年号がついた日付候補が出現していない場合、4 桁の西暦が 2 桁に省略されたものである可能性が高いと考える。よって、以下のルールにより、4 桁の西暦に変換する。
 - 年が 19 以下なら、“20” を補完
 - 年が 20 以上なら、“19” を補完

以上の補完処理を行った後、年月日すべての情報がそろっている日付表現候補を、<date> タグで囲む。

4.3 entry の抽出

ある Web ページが blog かどうかの判定は、その Web ページから entry 集合を抽出し、それら entry 集合が後述する制約を満たすかどうかによって行う。しかしながら、Web ページ中の複数箇所に、日付を伴って出現する記述が現れる可能性がある。

そこで、システムはまず前節で発見された日付表現を“同じタイプの日付表現”毎にグルーピングする。“同じタイプの日付表現”とは以下の条件をすべて満たす日付表現の集合を指す。

- タグの係り方が同じ
- 日付表現を囲む HTML タグ中において、先頭もしくは末尾から日付表現までの byte 数が同じ
ex.) 2004/1/1 元旦の出来事
2004/1/12 成人式に行って来ました。
以上二つの日付表現は、先頭からの byte 数が同じである。
- 日付表現のフォーマットが同じ
ex.) “2004/1/1” と “1/2” は同じフォーマット
“1-Jan-2004” と “2004/1/2” は別フォーマット

¹²Hypertext Transfer Protocol – HTTP/1.1 <http://www.w3.org/Protocols/rfc2616/rfc2616.html>

これら同じタイプの日付表現は、ブラウザによってレンダリングされた際に同じように表示されるため、システムは各タイプに対して、Web ページから entry の集合を抽出することを試みる。

そして、最終的にはそれら複数の entry 集合の中から、もっともらしい entry 集合を決定するため、後述する非 blog ページのフィルタリング条件によってフィルタリングされなかった entry 集合のうち、もっともサイズが大きいものをその Web ページに対する entry の集合であると考ええる。

entry の開始位置の決定

以下では、あるタイプに属する日付表現に対し、各日付表現によって修飾される entry の開始位置を決定する手法について述べる。

まず、すべての日付表現に対して、それぞれの日付表現部分にかかっているタグの HTML 文書中での出現位置を比較する。例として、以下の例を考える。

- (1) /body(0)/div(3)/table(279)/
tr(280)/td(281)/date(283)
- (2) /body(0)/div(3)/table(350)/
tr(351)/td(352)/date(354)

括弧の中の数字は、HTML 文書中での位置を示す。これら二つの日付表現にかかるタグの位置を見ていくと、div(3) の直下にある table タグの位置が異なっている。これは、深さ 3 の位置にある table タグでセグメンテーションを行えば、各セグメントにちょうど一つの日付表現が属することを示している。つまり、この深さ 3 の位置にある table タグは entry の開始位置を示すと考えることが出来る。

ただし、三つ目の同じタイプの日付表現として、以下の様なものが加わった場合は、

- (3) /body(0)/div(3)/table(350)/
tr(359)/td(360)/date(362)

div(3) の直下にある table タグではなく、さらに一階層深い 4 階層目にある tr タグが entry の開始位置となる。これは、先ほどの table タグでセグメンテーションを行うと、(2) と (3) の日付表現が同じセグメントに属してしまうためである。

また、前節で述べたように、日付表現は date タグで囲われているため、このような entry の開始を示すタグは、同じタイプの日付表現があればどのような場合にも必ず存在する。

entry の終了位置の決定

次に終了位置の決定について説明する。

日付部分が date タグで囲まれているという前提条件の下で、entry のタイプは図 2,3 の二種類に限定することが出来る。

図 2 に示す一つ目のタイプは、各 entry が何らかのタグで囲まれている場合である (図 2 の場合は、table タグによって囲まれている)。このような場合、entry の

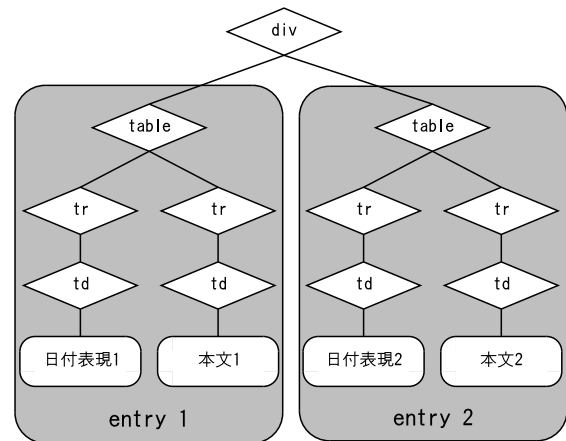


図 2: タイプ 1

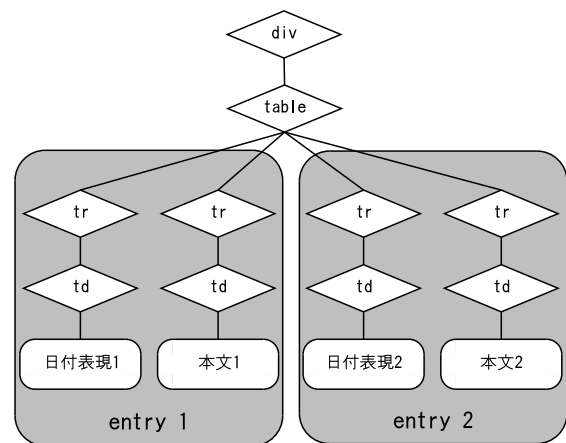


図 3: タイプ 2

開始位置を与えるタグで囲まれた部分を抽出することで、entry を抽出することが出来る。

図 3 に示す二つ目のタイプは、entry がタグで囲まれていない場合である。このような場合は、本来、南野ら [8] が提案したような繰り返しパターンの解析を行わなければならないが、パターンの解析には非常にコストがかかるという問題点がある。そこで、「日付が entry の上部に存在する」と仮定することで、次の entry の開始位置の直前が終了位置であるというヒューリスティックを用いることが可能になる。

よって、以上の二つのタイプを考慮すると、entry の終了位置は以下のような条件で決定することが出来る。

1. 開始位置のタグに対応する終了タグが現れる位置
2. 次の entry の開始位置の一つ前の位置

ただし、最後の entry については、上記 2. の条件が使用できないため、以下の条件を追加する。

3. 開始位置のタグと DOM 中の深さが同じタグのうち、それ以前の entry で一度も出現していないタグが現れる一つ前の位置

以上のような処理により各日付表現が修飾する entry の開始位置と終了位置を決定することで、それぞれの日付表現のタイプに対して entry 集合を決定することができる。

4.4 非 blog ページのフィルタリング

前節の処理によって抽出された entry には、BBS、Chat、メーリングリストのアーカイブ、サイトの更新情報、イベントの告知なども含まれる。なぜなら、これらのものもそれぞれのコンテンツが日付表現を伴って現れるからである。

よって、発見された複数の entry 集合に対してフィルタを適用し、blog 以外のものを除去を試みる。そして、すべての entry 集合がフィルタリングされた場合、そのページは blog で無いと判断する。

以下では、使用しているそれぞれのフィルタについて説明する。フィルタは、entry を抽出する Web ページに関するフィルタと抽出された entry 候補に対するフィルタの 2 種類を用意している。

Web ページに対するフィルタ

- URL に、“bbs”、“chat”、“session” を含む Web ページ
- ページのタイトルに「掲示板」、「bbs」、「メールマガジン」を含む Web ページ

entry 候補に対するフィルタ

- 未来の日付を含む entry が存在する entry 集合 (イベントの告知などをフィルタリング)
- 更新間隔が一ヶ月以上開いている entry 集合 (blog は、一か月に一度以上の頻度で書かれることが多い)
- 同じ日付が 3 回以上繰り返す entry 集合 (掲示板やチャットをフィルタリング)
- 日付が昇順もしくは降順に並んでいない entry 集合
- 2 番目に大きい entry のサイズが 150byte 以下 (UTF-8 で計算) の entry 集合 (短いものには、ページの更新情報などが多い)
- すべての entry からアルファベット、数字、スペースを除いた byte 数の entry 集合での平均が 150byte 以下 (UTF-8 で計算) の entry 集合 (日本語の blog のみを対象とするため)
- 切り出した entry において、日付表現が上から 2 行以内に出てこない entry 集合 (行数は改行効果のあるタグを数えることによって計算する)
- 以下の単語のいずれかを、すべての entry に含む entry 集合
「管理者」、「管理人」、「生年月日」、「日時」、「発言」、「内容」、「返事」、「返信」
(掲示板、誕生日リスト、メーリングリストのアーカイブ、チャットのフィルタリング)
- 切り出した entry の半数以上に「re:」を含む entry

集合 (メーリングリストのアーカイブのフィルタリング)

- 以下の条件を満たす文が一つもない entry が過半数を占める entry 集合
 - 動詞あるいは助動詞 + 過去の「た」を含む文
 - サ変名詞 + 動詞「する」 + 過去の「た」を含む文
 - 係助詞あるいは格助詞を含む、体言止めの文
 - 名詞あるいは未知語 + 助動詞「だ」を含む文
 - 形容詞を含む文

(blog は、何らかのイベントに対する記述を含むことが多い)

4.5 更新から得られる情報に基づいた非 blog ページのフィルタリング

前節までの処理で blog と判定されたページは監視され、更新がある度に新しい entry を抽出し、データベースへの追加を行う。その際、更新後のページに対しても前節で述べたフィルタを適用し、もし違反があればそのページは blog でなかったと判断し、過去のものまで遡りデータベースから消去する。ただし、entry のサイズに関する制約は、一時的に記事が少なくなるというケースを考えると不相当であるため、一度 blog と判定されたページには適用しない。(前節までの blog 判定処理は、一定期間を開けて数回行っているため、一度でもサイズに関する制約を満たせば、blog と判定される。)

また、更新から得られる情報のみでも、非 blog ページのフィルタリングが可能なケースが存在する。例として以下のようなケースを考える。

1. 2004 年 1 月 1 日に Web ページを取得し、2003 年 12 月の entry を取得した。
2. そのページは監視対象となる。
3. 2004 年 1 月 7 日に更新を検知し、再び entry を取得した。

本来、そのページが blog であるのであれば、新たに追加された entry は、2004 年 1 月 1 日以降、2004 年 1 月 7 日以前であるのが妥当である。

このように、前回 entry を取得した日時より過去の entry が取得される場合、システムはそのページを blog ではないと判断する。ただし、例えば一ヶ月書きためるようなケースを考慮して、現在の実装では、前回 entry を取得した日時より一ヶ月以前の entry が発見された場合のみ、フィルタリングを行うことにしている。

このようなケースが発見される理由の一つに、日付の補完誤りが考えられる。4.2 節の日付の補完処理では、ページ内に年に関する情報が全く無い場合に、直近の年で補完を行う。よって、Web ページ中に未来のイベントについて記述された、年を含まない日付表現がある場合、補完処理により一年前の年が補完される。そして、そのページに未来のイベントが追加された場合、

表 2: blog 判定精度

blog である	283 ページ	(94.3%)
blog でない	17 ページ	(5.7%)

表 3: 判定ミスの内訳

ページの種類	ページ数	割合
BBS	7	(41.2%)
イベント案内	3	(17.6%)
更新情報	3	(17.6%)
ニュースリリース	2	(11.8%)
メールマガジン	1	(5.9%)
Amazon.com のレビュー	1	(5.9%)

補完の誤りによって、過去の日時が追加されるという現象が発生する。

本研究では、このような未来のイベントに関する記述は blog でないと考えているため、この処理によって収集対象から除去される。

5 評価

本節では、本研究で構築した blog 収集システムの評価を行う。

構築したシステムは、2 週間運用され、39,272 の blog ページから 466,809 の entry を抽出した。以降の評価は、これらの blog, entry に対して行う。

5.1 blog 判定に関する評価

まず blog 判定の精度に関して述べる。

先ほどの 4 万弱の blog ページから、host 名、ページタイトルが重複しないように、300 ページをランダムサンプリングし、人手によって評価を行った。表 2 に結果を示す。ただし、正しく判定された 283 ページ中、24 ページで、entry の抽出が完全ではなかった。原因は以下の三つである。

- entry の日付表現にかかるタグの係り方が一定でない
- 不足しているタグを tidy が正しく修正できなかった
- 日付が entry の最後に示されている

また、blog であると判定されたにもかかわらず、実際は blog でないページの内訳を表 3 に示す。なお、17 ページ中 5 ページは、その後更新を監視することで、blog でないと判断されている。

次に、再現率に関する評価について述べる。

収集した Web ページが膨大であり、また、Web 中における blog の割合が、我々が調査したところ 0.1%以

表 4: blog でないと判定されたページに対する評価

blog でない	201 ページ	(67.0%)
blog である	99 ページ	(33.0%)

表 5: 判定ミスの原因

原因	ページ数	割合
更新間隔が一ヶ月以上	53	(53.5%)
文中の日付と誤認識	17	(17.1%)
一日分しか entry が無い	9	(9.1%)
同じ日付が連続	6	(6.1%)
サイズが小さい	6	(6.1%)
昇順降順違反	2	(2.0%)
日付の補完ミス	2	(2.0%)
日付が認識不可	2	(2.0%)
NG ワード	1	(1.0%)

下であるため、再現率を直接評価することは現実的でない。そこで、blog でないと判断された Web ページのうち、日付表現を 3 つ以上含む Web ページを host 名、ページタイトルが重複しないように、300 ページをランダムサンプリングし、それらのページが blog でないかどうかを、人手によって評価を行った。表 4 に結果を示す。

また、blog でないと判定された 175,057 ページから、30,000 ページをランダムサンプリングし、日付を 3 つ以上含むページの割合を調査したところ、3,715 ページ (12.38%) であった。これらの結果から推定される再現率は、blog と判定されたページ中の推定 blog 数、blog でないと判定されたページ中の推定 blog 数から、以下のように計算される。

$$\begin{aligned} recall &= \frac{39272 * 0.943}{39272 * 0.943 + 175057 * 0.1238 * 0.330} \\ &= 0.838 \end{aligned}$$

次に、blog であるにも関わらず blog でないと判断されたページに対して、フィルタリングされた原因を調査した。結果を表 5 に示す。

次に、一度 blog と判定されたページがその後の監視によって、blog ページでないと判定されたケースの評価を行う。監視は 1 週間行い、blog と判定されたページの内 76 ページが blog でないと判定された。表 6 にフィルタリングの原因と、その判定が正しかったどうかを人手で確認した結果を示す。

5.2 取得された blog に関する評価

2 節で述べたように、本論文で提案した HTML 文書を直接解析する手法は、

- blog ツールなど、特定のシステムを利用してい

表 6: 監視によるフィルタリング

原因	正	誤
文法フィルタ	7	8
更新間隔が一ヶ月以上	8	0
未来の日付	1	22
同じ日付が連続	21	0
昇順降順違反	8	0
NG ワード	0	1
合計	45(59.2%)	31(40.8%)

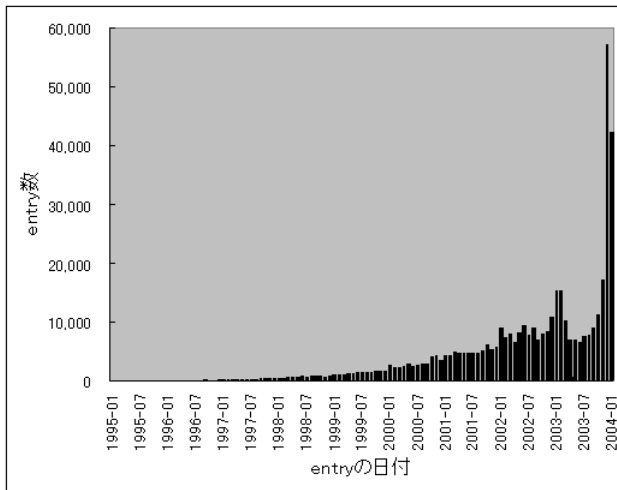


図 4: 取得された entry の日付の分布

ない Web 日記なども広く収集することが可能

- 過去のものまで遡って収集することが可能

な点に利点がある。

よって、取得された entry の日付分布と、取得された blog の内、どれくらいの数が blog ツール等を使用していないかについて評価を行った。

取得された entry の日付分布

図 4 に取得された entry の日付分布を示す。

システムの運用は、2003 年 12 月末から 2004 年 1 月にかけて行った。そのため、直近の entry が多数を占めるが、同時に過去のものも取得されていることがわかる。なお、現在のシステムは、1995 年以前の entry は取得しない仕様になっている。

blog ツールを使用していない blog の割合

あるページが blog ツールを使用しているか、していないかの判定には、使用されている blog ツールを判定する Perl モジュールである WWW::Blog::Identify[9] を使用する。ただし、このモジュールは日本で多く用いられている blog ツールや hosting サービスに対応して

表 7: blog ツールを使用している blog の割合

blog ツール不使用	32,219	(82.0%)
movable type	2,243	(5.7%)
hns ¹⁴	1,893	(4.8%)
楽天広場 ¹⁵	922	(2.5%)
a-News ¹⁶	423	(1.1%)
tDiary ¹⁷	307	(0.8%)
ココログ ¹⁸	163	(0.4%)
Akiary ¹⁹	131	(0.3%)
livedoor Blog ²⁰	127	(0.3%)
MEMORIZE ²¹	114	(0.3%)
Tomsoft Diary System ²²	110	(0.3%)
さるさる日記 ²³	106	(0.3%)
その他のツール使用	444	(1.1%)
合計	39,272	

いないため、「ブログツールリスト¹³」で紹介されているツールに対応させたものを使用する。現在、77 種類のツールを認識することが出来る。

表 7 に blog ツールを使用している blog の割合を示す。

6 考察

前節の評価を踏まえ、現状のシステムの問題点について考察する。

判定精度・再現率について

判定精度については、非常に良い結果が得られている。

判定ミスとなったページのほとんどは、ページの構成は blog と同じ様な構成で書かれていたため判定を誤ったものであった。例えば BBS は、頻繁に書き込まれる BBS であれば、「同じ日付が連続」フィルタで削除できるものが多数であるが、更新間隔も blog と同じ程度しかなく、また、NG ワードも出現しないと、実質的な blog との違いは、一人で書いているか、複数人で書いているかくらいしかないケースも存在する。また、未来のイベントについて書かれたページについては、その後更新があればフィルタリングすることが出来るが、更新が無く、また書かれた日時が不明な場合、判定が非常に難しい。

精度が良い一方、再現率には問題がある。

blog の更新間隔が思っていたよりも開くものが多かつ

¹³ ブログツールリスト <http://artifact-jp.com/weblog/>

¹⁴ hns <http://www.h14m.org/>

¹⁵ 楽天広場 <http://plaza.rakuten.co.jp/>

¹⁶ a-News <http://www.appleple.com/cgi/>

¹⁷ tDiary <http://www.tdiary.org/>

¹⁸ ココログ <http://www.cocolog-nifty.com/>

¹⁹ Akiary <http://www.hi-ho.ne.jp/yakira/akiary/>

²⁰ livedoor Blog <http://blog.livedoor.com/>

²¹ MEMORIZE <http://www.memorize.ne.jp/>

²² Tomsoft Diary System <http://tds.dive-in.to/>

²³ さるさる日記 <http://www.diary.ne.jp/>

たのが一番の原因である。しかしながら、更新間隔による条件によって、blog でないと判定されている非 blog ページも多数あるので、今後更新間隔の閾値について、吟味する必要がある。

また、日付の認識、補完は改善の余地がある。日付に類似したものとして、シリアル番号や URL があるが、これらのものと日付を区別するのが非常に難しい。今回は blog 判定にかかる時間の短縮を重要視し、明らかに日付であるケースのみにタグ付けを行ったため、このような結果が生じたと考えられる。よって、今後、このような文中の日付などにもタグを付けた結果が、判定精度・再現率や entry の抽出にどのような影響を与えるかを調べる必要がある。また、補完に関しては誤った年を補完してしまうケースが目立ったが、これについても Web ページの構造解析、あるいはレンダリングされたページのレイアウト解析など、さらなる深い処理が必要になると思われる。

別の問題として、抽出された entry 数が少ないケースでは、「昇順降順」に関するフィルタが機能しなかったり、また逆に「文法」や「NGワード」、「サイズ」に関するフィルタが効きすぎてしまう場合があった。このような entry 数が少ない場合に対する処理について、今後考察を行う必要がある。

取得された blog の性質について

前節に示したように、本手法により、過去の entry も多数取得出来ている。これは、RSS の収集に基づいた手法では、取得できない blog が数多く収集できていることを示していると考えられる。

また、blog ツールを利用しない blog も多数取得できている。現在、Web ページのクローリングは、blog 間にはリンクが張られることが多いという特徴を利用し、blog ページと判定されたページからリンクを 2 階層辿ることで収集を行っている。現状は収集を開始したばかりで、まだデータにかなりの偏りがあると考えられるため、今後クローリングが進み、ある程度網羅的に収集が出来た時点で再び考察する必要がある。

7 おわりに

本研究では、Web 上にある、速報性、リアルタイム性のある新鮮な情報が発信されることの多い blog を網羅的に収集し、監視する手法を提案した。

従来の blog 収集手法は、RSS などの計算機が扱いやすい情報を利用することの出来る特定のツールを使用した blog のみに限定されていたため、その収集対象は非常に限定されたものであった。そのような現状に対し、我々は HTML 文書を直接解析し、そのページが blog であるかどうかを判定することで、blog ツール登場以前からある日本固有の文化である Web 日記などを対象に含め、大量な個人の発信する新鮮な情報を網羅的に収集することの出来る手法を提案した。

現状のシステムは、まだプロトタイプではあるが、結果はこの手法の有効性を示していると考えられる。

今後の課題として、さらなる blog 判定精度の向上と網羅的な収集が上げられる。また、現在のシステムの問題点として、重複した entry を取得してしまうという問題がある。例えば、movable type などの blog ツールでは、同じ entry が週単位や月単位の entry 一覧ページなど、複数のページに含まれるケースがある。また、同様の問題として、現在は URL ベースで entry を管理しているために、Duplicate Hosts 問題 [10] により複製を取得してしまう可能性もある。これらの問題は、RSS ベースの手法では問題にならないため、今後解決方法を研究していく必要がある。

また、現在、収集した blog を検索することのできるシステム、また、収集した blog に対してマイニングを行うことで有用な情報を発見することのできるシステムを公開に向け開発中である。

謝辞

本研究の一部は、独立行政法人情報処理推進機構 (IPA) 「未踏ソフトウェア創造事業」喜連川 優 PM による「blog ページの自動収集と監視に基づくテキストマイニング」の成果の一部である。また、本研究の一部は文部科学省科学研究費 (21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」) の補助のもとに行われた。

参考文献

- [1] 松村真宏. チャンス発見のためのコミュニティマイニングに関する研究. 平成 14 年度東京大学大学院工学系研究科電子工学専攻博士論文, 2003.
- [2] Benjamin and Mena Trott. Trackback technical specification. <http://www.movabletype.org/docs/mttrackback.html>, 2002.
- [3] Dan Libby. Rdf site summary (rss) 0.9 official dtd. <http://my.netscape.com/publish/formats/rss-0.9.dtd>, 1999.
- [4] Dave Winer. Weblogs.com xml-rpc interface. <http://www.xmlrpc.com/weblogsCom>, 2001.
- [5] yomoyomo. Hotwired japan - 日本における blog の過去・現在・未来. <http://www.hotwired.co.jp/matrix/0305/004/index.html>, 2003.
- [6] 藤木稔明, 南野朋之, 鈴木泰裕, 奥村学. document stream における burst の発見. 情報処理学会研究報告, 2003-NL-160, 2004.
- [7] Dave Raggett. Clean up your web pages with html tidy. <http://www.w3.org/People/Raggett/tidy/>.
- [8] Tomoyuki NANN0, Suguru SAITO, and Manabu OKUMURA. Structuring web pages based on repetition of elements. In *Second International Workshop on Web Document Analysis (WDA2003)*, 2003.
- [9] Maciej Ceglowski. Www::blog::identify - identify blogging tools based on url and content. <http://search.cpan.org/~mceglows/WWW-Blog-Identify-0.06/Identify.pm>, 2003.
- [10] Monika R. Henzinger, Rajeev Motwani, and Craig Silverstein. Challenges in web search engine. In *Proc. International Joint Conference on Artificial Intelligence*, pp. 1573-1579, 2003.