ウェブアーカイブにおける正確な時系列推定を伴う収集手法

2002MT065 小川 晃弘　　2002MT072 尾崎 忠
指導教員　河野 浩之

1　はじめに

近年, インターネットは急速な発展を見せ, それにともない WEB 上のデータの量は増加していくとことが見込まれる. このように新たなページが生まれる一方でさまざまな理由により閲覧できなくなってしまうページも数多く存在する. そこで WEB ページを文化財として後世に残していくために WEB アーカイブという技術が生まれた.

WEB アーカイブは WEB サーチエンジンと異なり, ページ収集はページの更新にあわせて何度もされなければならず, 収集したページは同じ URL であっても上書きせず, 時系列に沿って保存していく必要がある [1]. WEB ページの更新日時を正確に検知することができれば時系列にそったページ収集を行うことが可能となる.

本稿では WEB アーカイブにおける課題の 1 つになっている再収集頻度の決定方法ついて提案, 検証を行う. 2 章で紹介する Cho 氏の提案した検索のタイミングのスケジューリングを最適化する方法を基に, Last-Modified タグから得られる最終更新日時を組み合わせることで高い収集率と時系列に沿ったデータの収集の両立を目標とする. 2 章では WEB アーカイビングの概要の説明, および再収集に関する先行研究の紹介を, 3 章では我々の提案する手法, 4 章で実験を行うためのシステムの構成について説明, 5 章では実験の結果を示す. 最後に 6 章でまとめを述べる.

2　WEB ページ収集手法の背景と先行研究

WEB アーカイビングにおいては再収集頻度の決定はまだあまり研究発表されていない. しかし, サーチエンジンの分野では既にいくつかの論文が発表されており効率的な WEB ページの収集手法が提案されている. 本節ではそれらの関連研究で提案された手法について紹介していく.

2.1　WEB アーカイブの技術背景

現在, 世界中の国立図書館を中心に WWW 上の情報を収集, 保存, 利用を目的に WEB アーカイビングが行われている. WEB アーカイビングを行っている代表的な機関として Internet Archive, MINERVA, PANDORA, AOLA, WARP, などが挙げられる [2]. これらの機関は収集方針において, 選択的収集とバルク収集を採用する機関の 2 つに分類することができる. 選択的収集では WEB 上のデータをサイト単位, あるいは資料単位で収集する方法であり, きめ細かいアーカイビングが可能な反面, 多数のページを収集することは困難であるという性質を持つ. もう一方のバルク収集では WEB 上のデータを国単位, もしくは世界全体など非常に大きな規模で一括して収集をおこなう. 選択的収集を比較し低いコストで大規模なアーカイビングが可能であるが, 個々の精度は選択的収集に比べ見劣りするものとなってしまう.

表 1　世界の主な WEB アーカイビングプロジェクト

組織	名称	収集方法
インターネットアーカイブ	–	バルク
米国議会図書館	MINERVA	選択的
フランス国立図書館	–	バルク
スウェーデン王立図書館	Kulturarw3	バルク
デンマーク王立図書館	netarchive.dk	両方
オーストラリア国立図書館	PANDORA	選択的
オーストリア国立図書館	AOLA	バルク
国立国会図書館	WARP	選択的

2.2　リンク構造を利用した WEB ページの更新判別手法

まずリンク構造を利用した研究を紹介する [3]. 観覧者が最も訪れやすいサーバのトップページに「更新の知らせ」のようなものが設置されていると考えられることから, トップページの更新を検知してから, 下位ページの収集を行う. しかし, 大規模サーバの場合, 1 つのサーバの中に複数の個人ページが存在している. この場合, それぞれの個人ページにトップページが存在し, 更新の目安になっていると考えられる. そこで, サーバを小さなサイト毎に分割して, そのトップページを元に更新を判別する. 結果, どの期間でも各手法が WEB ページ全体より高い更新率を示す結果となった.

2.3　アグリゲーターによる間接的ページ収集手法

アグリゲーターによって RSS フィードから新しいポスティングを収集し, ユーザがアグリゲーターから間接的にポスティングの回収を行う [4]. このアグリゲーターにとっての 1 つの重要な挑戦というのがサイトから新しいポスティングをすばやく回収し収集の遅れを最小限化することである.

多数のソースが非常に似通ったポスティングパターンをもつことから, 少数のクラスタに分類することで, クラスタのパターンに基づいて最適な検索スケジューリングを見つける. 全てのソースを均等に検索する実験基準の標準スケジュールに比べ, 検索は時間あたりに同じ回数行われるが, 検索ポイントは最適化される検索のみの予定化では 12%, ソース毎に異なった回数ポスティングを検索するソースの配置のみでは 33%, 検索回数, 検索ポイントともに最適化された組み合わせでは 40% 遅れが減少した.

3 Last-Modified を用いた提案手法

本稿では 2.3 節で紹介したアグリゲーターによる間接的ページ収集手法に, 更新日時を知るための Last-Modified タグを組み合わせることで, WEB アーカイブにおける効率的な再収集頻度の決定方法を提案する.

3.1 Last-Modified とは

Last-Modified ヘッダは HTTP/1.1 に標準で含まれるヘッダのひとつであり最終更新時刻を記述している [5]. サーバとクライアントが HTTP に基づいた形式でデータのやりとりをする際に, サーバ側から送られるデータのヘッダの中に Last-Modified は含まれている.

3.2 収集遅延の定義

WEB ページをどの頻度で収集すべきか, WEB ページの更新が行われてから, クローラーが収集するまでの遅れを求める. 時間 t_1,...,t_k でページを更新する WEB ページ O を検討する. クローラーは τ_1,...,τ_m で WEB ページ O から新しく更新されたページを回収する. 更新 t_i で関連づけられる遅れは式 (1) のように定義される.

$$D(t_i) = \tau_j - t_i \tag{1}$$

τ_j は $t_i \geq \tau_j$ のとき最小値をとる. WEB ページ O からのトータルの収集遅延を表す式 (2) が定義される.

$$D(O) = \sum_{i=1}^{k} D(t_i) = \sum_{i=1}^{k} (\tau_j - t_i) \tag{2}$$

各 WEB ページ O_i が重み w_i で関連づけられるとき, クローラーによって観測されたトータルの重みづけされた遅れ $D(A)$ は式 (3) のように定義される.

$$D(A) = \sum_{i=1}^{n} w_i D(O_i) \tag{3}$$

3.3 収集方針

1 日につき m_i 回回収される WEB ページ O_i を考える. ポアソンモデルの下で, 回収が一定間隔でスケジュールされるとき, WEB ページ O_i からの更新のトータルの遅れ $D(O_i)$ が最小値と示すことができる. 更新率 λ_i, 重み w_i の場合 $D(A)$ は 式 (4) で表される.

$$D(A) = \sum_{i=1}^{n} \frac{\lambda_i w_i T}{2m_i} \tag{4}$$

$D(A)$ は Le'grange 乗数方法を使うことによって式 (5) で表されるよう最小限にできる.

$$\frac{\partial D(A)}{\partial m_i} = -\frac{\lambda_i w_i T}{2m_i^2} = -\mu \tag{5}$$

よって一日につき WEB ページ O_i に何回回収すればよいのか, 以下の式 (6) で示すことができる

$$m_i = \sqrt{\lambda_i w_i T / 2\mu} = k\sqrt{\lambda_i w_i} \tag{6}$$

3.4 ページ収集アルゴリズム

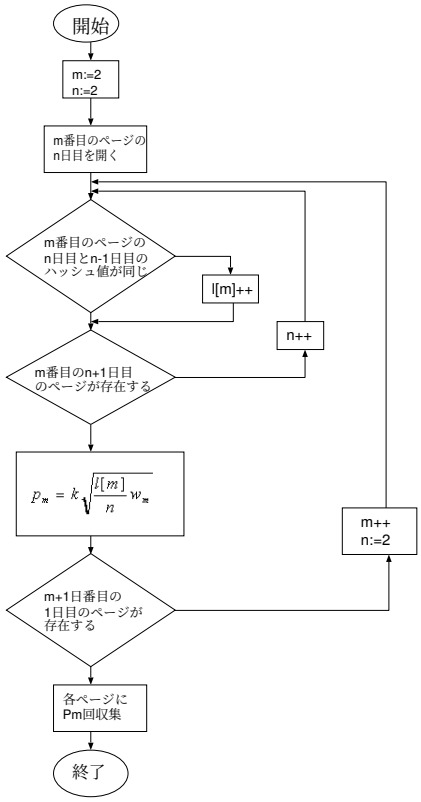

図 1 ページ収集アルゴリズム

最適な収集回数 P_m を求めるために, まずページの更新頻度 $l[m]/n$ を調べる (図 1). ページの更新回数は前日のページとのハッシュ値の比較によって求める. そのため, m 番目のページの更新回数 $l[m]$ を調べるには m 番目のページの n 日目と n-1 日目のハッシュ値を比較し, 値が異なる場合更新が行われているとみなし, 更新回数 l[m] の値を 1 増やす. 値が等しい場合は更新がされていないものとし翌日のハッシュ値と比較して行く. 40 日間での更新回数を調べたところで, 最適な収集回数 P_m を求める. 式 (6) より以下の式で表す.

$$P_m = k\sqrt{\frac{l[m]}{n} W_m} \tag{7}$$

M は時間あたりの収集回数, M は時間あたりの再収集の回数, W_m は重みを表す. m 番目の最適な収集回数 P_m を求めたら, m+1 番目に対しも同じように更新回数を調べ, 最適な収集回数 P_m+1 を求める. このようにして最適な収集回数を収集した全てのページに対して求めることによって, 各ページに対して, 最適な再収集の回数を決定する.

4　ページ収集におけるシステム構成

様々な種類の WEB ページから更新傾向を調査し, 効率的な WEB ページの再収集頻度を決定する. そこで以下のような条件の下実験を行う.

4.1　調査対象

特長をもったサイト毎に収集することで, 各サイトに対して再収集の頻度がどれほど適切であるか検証が可能である. そこで我々は 4 つのカテゴリにわけて収集を行う. ニュースサイトは更新が多く大規模なサイ, 個人サイトは小規模のサイト, 大学サイトは更新が頻繁に行われないサイト, ブログは動的なサイトと考える. ニュースサイトから 10 サイト, 個人サイトから 30 サイト, 大学のサイトから 50 サイト, ブログサイトから 30 サイト選び収集を行う.

4.2　robots.txt と META タグ

goo や Google などのロボット型検索エンジンに対する命令を記述する robots.txt というものがある. 収集ロボットでページ収集する場合この制約を守らなければならない. robots.txt の場合, サイトのトップにおかなければならないが, META タグを利用することで, 各ページに対して制約をつけることができる.

4.3　収集ツール

我々は WEB ページの収集に http ヘッダを取得できる WEB 自動巡回ツール Wget(Ver1.9) を用いる. Wget にはリトライ機能, 処理が軽いと言った利点があげられる. また, 様々なオプションを用いることによって, 必要な情報を収集する. 表 2 に設定値を示す.

表 2　Wget の主なオプション

コマンド	説明
-i file	file から URL 取得
-s	HTTP サーバからヘッダを保存する
-r	再帰的に回収を行う
-l depth	再帰回収の深さを depth に指定する
-R rejlist	持って来ないファイルを指定する

大学のサイトでは pdf ファイルが多く存在し, ファイル用量が多く収集に時間がかかることから, -R オプションを用いて, 拡張子が jpg, gif, pdf のファイルは収集しないようにする. また深さは, 実験の規模, 重要なページが深度の浅いところに存在すると考え 3 回リンクを辿った時点で収集を打ち切ることにする.

1 日 1 回 Web ページ巡回ツール Wget を用いて WEB ページを収集していく. そのため cron を用いて, 決められた時間に Wget を動かし自動で WEB ページを収集する. 収集したページから Last-Modified, ページの取得日時, URL, リンク URL, 更新の有無, ページのハッシュ値といった必要な情報を抽出して PC 内に WEB 空間を実現する.

5　重み付けによる比較検証

4 つのカテゴリに分類した WEB サイトに対しページ収集をし, その収集された, ニュースサイト 55279 ページ, 大学サイト 9184 ページ, 個人サイト 7859 ページ, ブログサイト 33677 ページを各ページ毎に収集頻度を変えることにより, 検索システム内のデータの新鮮さを常に一定に保つ.

5.1　拡張子による重み付けの結果

拡張子による重み付けを行ったさいの再収集の効率, カバー率の変化, 時系列の誤差を図 2 に示す. 拡張子が jpeg, png, pdf, gif のファイルの重みを減らすことで, 相対的にその他の拡張子のファイルに対して重みを与える. グラフ中の横軸は jpeg, png, pdf, gif のファイルの重みを表す. まず再収集効率であるが, 重みが 1 のときと重みが 0.25 のときを比較して, 全てのカテゴリについて低下し, 最も変化が少ない大学サイトで 0.01%, 最も変化の大きい個人サイトでは 0.18% の低下がみられた. カバー率についても全カテゴリで低下し, 最も低下した個人サイトで 0.97% 低下した. ディレイについては, 個人サイトで 4.69 日から 5.91 日, 約 26% の増加がみられた. その他のカテゴリでは 0.1 日以下のわずかな増加にとどまった.

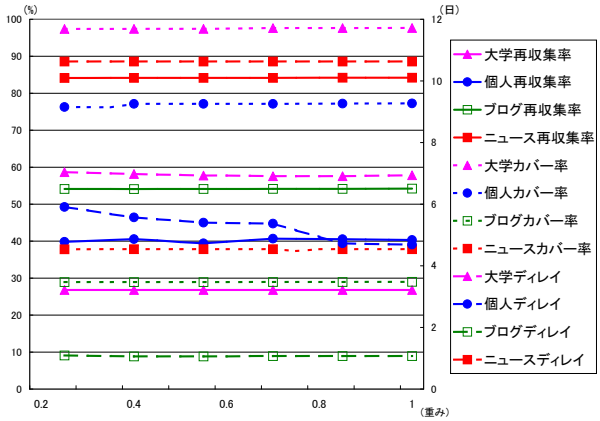

図 2　拡張子による重み付け

5.2　ファイル名による重み付けの結果

ファイル名による重み付けを用いた再収集の結果をそれぞれ図 3 に示す. main, top, index などの重要と思われる単語を含むファイルの重みを大きくして再収集を行う. グラフの横軸は main, top, index などの単語を含むファイルの重みを表している. ファイル名による重み付けでは, ニュースサイトとその他のカテゴリで変化のしかたに大きな差がみられた. ニュースサイトでは再収集率が 25.69% の低下, カバー率が 11.54% 低下した. それに対しその他のカテゴリでは再収集率で 0.04% から 4.67%, カバー率については 0.23% から 3.8% の低下と, ニュースサイトと比較してわずかな変化にとどまった.

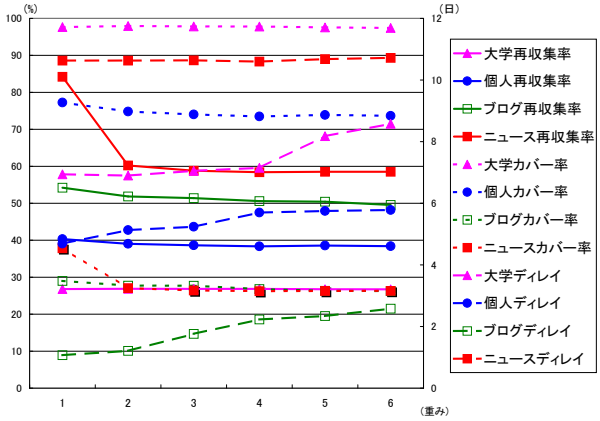

図 3　ファイル名による重み付け

5.3　Last-Modified による重み付けの結果

次に我々の提案する Last-Modified を用いた重み付けについて検証を行う．Last-Modified ヘッダを持たないファイルの重みを増加させ再収集を行い，その変化を図 4 に示す．最も重み付けの効果が低かったニュースサイトでは再回収効率が 9.94% 低下，カバー率が 3.46% 低下，ディレイが 0.11 日低下した．次にブログサイトでは再回収効率が 1.48% 低下，カバー率が 0.79% 低下，ディレイが 0.27 日低下というように大きな変化は見られなかった．これらに対し大きな効果が確認できたのが個人サイトと大学サイトである．個人サイトでは再回収効率が 3.82% 低下，カバー率が 6.33% 低下したものの，ディレイが 5.19 日と大きな低下を見せた．また大学サイトでは再回収効率が 0.4%，カバー率が 0.3% とわずかな低下にとどまるのに対し，ディレイは 2.41% という大きな減少を見せた．

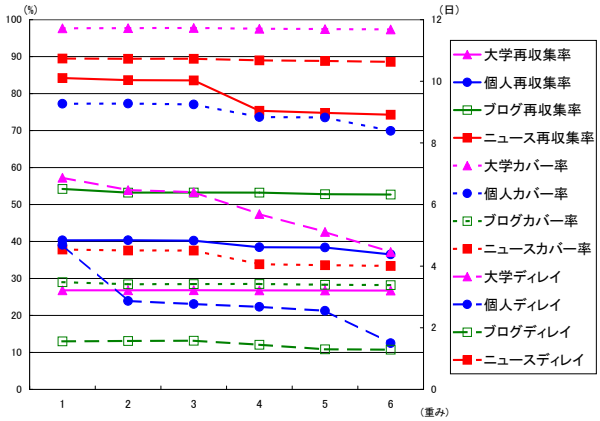

図 4　Last-Modified による重み付け

5.4　考察

今回の実験のうち，拡張子による重み付けとファイル名による重み付けでは重みを大きくするほど，わずかではあるが再回収の効率も低下し，それにともないカバー率も減少，ディレイについても増大するという結果になった．この原因として考えられるのは主に以下の 2 つのことである．まず重みを変化させる拡張子，およびファイル名のファイル自体の WEB 上における存在数が少なかったこと．次に重みをつけるほど結果が悪くなった原因としては，実験に用いた収集アルゴリズムの中に更新率が反映されていたことが原因と考えられる．それに対し我々が提案した Last-Modified による重み付けは今回の実験でカバー率，および再収集効率には悪影響を与えるものの，ディレイの減少については効果があることが実証された．我々の提案手法が効果を上げた大きな要因は，更新率を反映しない手法であることが考えられる．先に考察した 2 つの方法はともに拡張子やファイル名から更新率を推測するという方法を用いているため，今回のような更新率を調査することで正確な更新率を用いるアルゴリズムでは十分な効果を発揮することは難しい．それに対し Last-Modified を用いる手法では更新率を用いたアルゴリズムにおいてディレイの減少に大きな効果を発揮することが可能である．

6　まとめ

本稿では Last-Modified を用いた重み付けの有用性を，他の重み付けと比較を行うことで検証してきた．その結果，WEB アーカイビングという目的に特化した場合，Last-Modified は時系列の正確さを向上させるうえで大きな効果があることが明らかになった．また，再収集アルゴリズムとの相性についても比較的汎用性が高く，容易に導入が可能であるなどの利点も明らかとなった．今後の課題としては，より大きな規模での実験が挙げられる．

参考文献

[1] 小城正士, 廣瀬信己, 河野浩之: Web アーカイブにおける時系列閲覧:単一コレクションへの適用, DBSJ Letters, Vol.4, No.1, pp.153-156, 2005.

[2] 廣瀬信己: 国立国会図書館におけるウェブ・アーカイビングの実践と課題, 情報処理学会研究報告 No.51, pp.95-111, 2003.

[3] 熊谷英樹, 山名早人: リンク構造を利用した Web ページの更新判別手法, DEWS2004 5-B-02, 2004.

[4] K.C.Sia, J.Cho: Efficient Monitoring Algorithm for Fast News Alert, Technical report, University of California Los Angeles, 2005. http://oak.cs.ucla.edu/~cho/papers/sia-blog.pdf (accessed 2006.1.18)

[5] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee: RFC2616, http://www.ietf.org/rfc/rfc2616.txt(accessed 2005.9.16)