

HTML 構造の類似性を利用したスプログ検出方式

Splog Detection using Similarities of HTML Structures

片山 太一* 宇津呂 武仁* 芳中 隆幸† 河田 容英‡
Taichi Katayama Takehito Utsuro Takayuki Yoshinaka Yasuhide Kawada

福原 知宏§
Tomohiro Fukuhara

概要: 本研究では, ブログにおいてアフィリエイト収入を得ることを目的とするスパム (スパムブログ, スプログ) のうち, 特に, 同一のスパムブログ作成者が自動的に大量生成したと推測されるスプログの検出において, HTML 構造の類似性が効果的であることを示す. 具体的には, ブログの HTML ファイルにおける DOM ツリーから, コンテンツの最小単位に相当するブロックを抽出し, 複数のスプログの間でブロック構造の類似性を測定する. その結果, 同一ブログホストにおけるスプログのうち, 同一のスパムブログ作成者が自動的に大量生成したと推測されるスプログ同士では, ブロック構造が類似する傾向があることを示す. また, ブロック構造の類似性を素性として, SVM を用いたスプログ検出を行った結果において, スプログ検出の性能が向上することを示す.

キーワード: スパムブログ, 機械学習, HTML 構造, 信頼度, SVM

1 まえがき

ブログには個人の意見情報が記されており, 市場の動向を推測するための手掛かりや製品についての意見調査をする上で有益であるとして, 近年注目を集めている. そのため, 従来からあるインデクシングのみを行う検索エンジンとは異なる, ブログ特有の情報検索サービスが出現している.

具体的には, ブログ解析サービスとして, *Technorati*¹, *BlogPulse*², *kizasi.jp*³, *blogWatcher*⁴ [1] などが存在する. 多言語ブログサービスとしては, *Globe of Blogs*⁵ が

言語横断ブログ記事検索機能を提供している. また *Best Blogs in Asia Directory*⁶ がアジア言語ブログの検索機能を提供している. *Blogwise*⁷ もまた多言語ブログ記事の分析を行っている.

一方で, ブログのウェブコンテンツの作成と配信は非常に容易になっており, そのことが引き金となって, アフィリエイト収入を得ることを目的とするスパムブログ (以下, スプログ) が急増している [2, 3, 4, 5, 6]. スプログにおいては, 通常, 広告主への誘導または対象サイトのページランクを増加する目的のもとで, 機械的な文書作成や他サイトの引用という手段を用いて自動的に記事を生成し, 大量のリンクを有するブログを機械的に自動生成する. [4] は英語ブログにおいて, 約 88% のブログサイトがスプログであり, それは全ブログポストの 75% を占めると報告している. このことから, [3, 7] に述べられているように, スプログは情報検索品質の低下やネットワークと格納資源の多大な浪費などといった問題を起こす要因となる. そのため, 近年, スプログの分析や検出を目的とした研究が進められている. [5] では, TREC⁸Blog06 データコレクションを用いて, スプログのピング時系列特性, 入力度数/出力度数の分布

*筑波大学大学院システム情報工学研究科, 〒 305-8573 茨城県つくば市天王台 1-1-1,

Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, 305-8573, Japan

†東京電機大学院工学研究科, 〒 101-8457 東京都千代田区神田錦町 2-2,

Graduate School of Engineering, Tokyo Denki University, Tokyo, 101-8457, Japan

‡(株)ナビックス, 〒 141-0031 東京都品川区西五反田 8-3-6, Navix Co., Ltd., 8-3-6 Nishi-Gotanda, Shinagawa-Ku Tokyo 141-0031, Japan

§東京大学 人工物工学研究センター, 〒 277-8568 千葉県柏市柏の葉 5-1-5, Research into Artifacts, Center for Engineering, University of Tokyo Kashiwa, Chiba 277-8568, Japan

¹<http://technorati.com/>

²<http://www.blogpulse.com/>

³<http://kizasi.jp/> (日本語のみ)

⁴<http://blogwatcher.pi.titech.ac.jp/> (日本語のみ)

⁵<http://www.globeofblogs.com/>

⁶<http://www.misohoni.com/bba/>

⁷<http://www.blogwise.com/>

⁸<http://trec.nist.gov/>

表 1: スブログ/非スブログデータセット

(a) スブログ/非スブログ数

ブログホスト	C 社	S 社	その他	合計
スブログ数	198	293	277	768
非スブログ数	210	259	2849	3318
合計	408	552	3126	4086

(b) 大量生成型スブログ数

ブログホスト	C 社				S 社			
大量生成型スパマー ID	1	2	3	4	1	2	3	4
スブログ数	140	26	31	33				

特性, 典型的な単語群を分析している. また, [4, 6] は, *BlogPulse* データセットを用いたスブログ分析の結果を報告している. 一方, [8, 9, 10, 4] では, スブログを機械的に特定し, 排除する技術について報告している.

以上の先行研究をふまえて, 本論文では, 同一のスパムスブログ作成者が自動的に大量生成したと推測されるスブログは, HTML 構造が類似していることを示す. また, 機械学習のひとつである Support Vector Machines [11] (SVM) を用いた枠組みによって, スパムスブログの判定を行うタスクを設定する. 未判定のスブログが入力されたとき, SVM の分離平面によって, スパムスブログかそうでないかを決定する. さらに, SVM では分離平面との距離を出力できるので, 分離平面との距離を信頼度として利用する [12]. 出力である信頼度を用いて, スブログを, 高信頼度スパムスブログ判定結果, 高信頼度非スパムスブログ判定結果, 低信頼度判定結果の三種類に分ける. 特に, HTML 構造の類似性を素性として, SVM を用いたスブログ検出を行った結果において, スブログ検出の性能が向上することを示す.

2 スブログ/非スブログデータセット

本論文では, 2007 年 9 月 ~2008 年 2 月の期間において収集した日本語スブログ/非スブログデータセットを用いる. 日本語スブログ/非スブログデータセットは, [13, 14] で提案された基準によって, スブログ/非スブログを判定した結果が付与されている. また, 日本語スブログ/非スブログデータセットの中で, 極めて構造が類似するスブログを, 同一作成者が自動生成している「大量生成型」のスブログ [13, 14] として同定し, 大量生成型スパマー ID を付与している. それ以外のスブログを「単発」スブログとした.

本論文の評価のうち, 特に大量生成型スブログを対象とした評価においては, スブログ・非スブログデータセッ

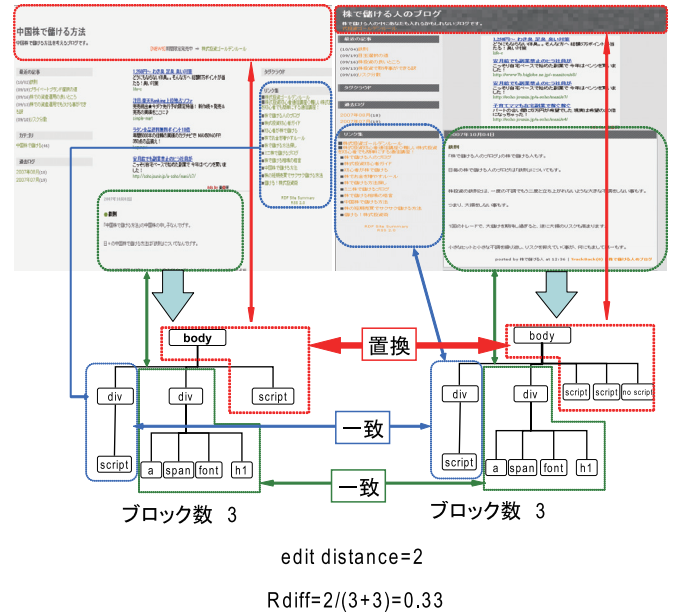


図 1: HTML 文書からの DOM 系列抽出および DOM 系列差分算出の例

ト中において, 一定数以上の大量生成型スブログを収集済の大量生成型スパマー ID を対象とする. 具体的には, 表 1 に示すように, ブログホスト C 社におけるスパマー ID=1 の大量生成型スブログ, および S 社におけるスパマー ID= 2, 3, 4 の大量生成型スブログを対象とする. なお, [13, 14] においては, 2 社以上のブログホストにまたがって, 同一の大量生成型スパマーによって自動生成されたと推測される大量生成型スブログも収集されている. しかし, 異なるブログホストのスブログ・非スブログの間では, HTML 文書の構造が大きく異なることが多いため, 本論文では, 同一の大量生成型スパマーによって自動生成された大量生成型スブログのうち, 特にブログホストが同一のものに限定して評価を行う.

3 HTML 構造の類似度

3.1 HTML 文書の DOM 系列の抽出

本論文では, [15] で提案されたブロック抽出の方式をふまえて, HTML 文書から DOM 系列を抽出する.

まず, 図 1 に示すように, HTML 文書 s 中の全ての HTML タグを木構造で表現する. 次に, この HTML タグの木構造に対して, ブロックレベル要素として用いられるタグのうち, P タグおよび DIV タグによって木構造を分割し, これらのタグの下位にあるタグを取り込むことによって, 個々のブロックを構成する. ここで, 一般に, ブロックレベル要素としては, P タグおよび DIV タグ以外のタグも用いられるが, 本論文では, 簡単化のために, P タグおよび DIV タグに限定する. また, [15]

と同様に、BODY タグも、P タグおよび DIV タグと同様に扱い、BODY タグの位置において、HTML タグの木構造の分割を行う。さらに、[15] では、ブラウザにレンダリングされない SCRIPT と STYLE の二タグ及びその下位ノードはブロック内に含めないとしているが、本論文では、ブロックの中身の詳細を区別するために、これらのタグ以下もブロック内に含める。

次に、ブロックにまとめあげられた HTML タグの木構造を横型探索することにより、ブロックのリスト構造を形成し、HTML 文書 s の DOM 系列 $dm(s)$ とする。

3.2 DOM 系列の差分の割合

HTML 文書 s および t に対して、それぞれから抽出された DOM 系列 $dm(s)$ 、および $dm(t)$ の差分を DP マッチングによって求める。DP マッチングの際、挿入および削除のコストを 1、置換のコストを 2 として、DP マッチングのより求まる編集距離を *edit distance* ($dm(s), dm(t)$) とする。次に、以下の式で s, t の DOM 系列の差分の割合 $Rdiff(s, t)$ を計算する。

$$Rdiff(s, t) = \frac{\text{edit distance}(dm(s), dm(t))}{|dm(s)| + |dm(t)|}$$

図 1 に、二つのプログラムの HTML 文書から DOM 系列を求めた後、差分の割合を模式的に算出する様子を示す。

3.3 DOM 系列の差分の割合の分布

次に、プログラムもしくは非プログラムの HTML 文書の集合 S および T の間で、HTML 文書 $s \in S$ および $t \in T$ の間の DOM 系列の差分の割合を求め、その分布を分析する。具体的には、大量生成型のプログラム同士、大量生成型のプログラムと単発プログラムの間、大量生成型のプログラムと非プログラムの間、および、単発プログラム・非プログラム同士の間で DOM 系列の差分の割合の分布を比較する。そのためにまず、HTML 文書 $s \in S$ に対して、HTML 文書集合 T の要素 $t \in T$ との間で、差分の割合 $Rdiff(s, t)$ が最も小さい 10 個を求め、その差分の割合の平均値を $AvMinDF_{10}(s, T)$ とする。

$$AvMinDF_{10}(s, T) = T \text{ 中で、} Rdiff(s, t \in T) \text{ の値が最も小さい 10 個の } t \text{ に対する } Rdiff(s, t) \text{ の平均値}$$

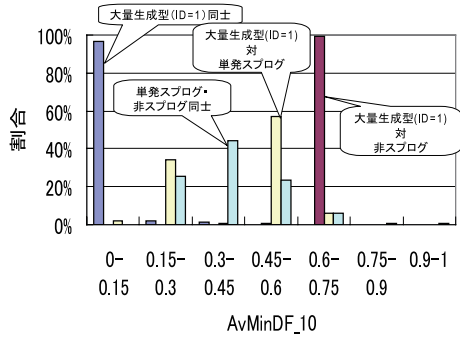
次に、大量生成型プログラム (ID=1, C 社)、および、大量生成型プログラム (ID=2, 3, 4, S 社) を対象として、大量生成型のプログラム同士、大量生成型のプログラムと単発プログラムの間、大量生成型のプログラムと非プログラムの間、

および、単発プログラム・非プログラム同士の間で DOM 系列の差分の割合の分布を求め、これを図 2 において比較する。ここで、大量生成型のプログラム同士の場合には、同一の大量生成型スパマー ID (例えば、ID=1) を持つ大量生成型プログラムの集合を S および T として、 S の要素 s に対して $AvMinDF_{10}(s, T)$ の分布を求める。一方、大量生成型のプログラムと単発プログラムの間の場合には、例えば、大量生成型プログラム (ID=1, C 社) を対象とする場合には、大量生成型スパマー ID=1 を持つ大量生成型プログラムの集合を S 、プログラムホスト C 社における全単発プログラムの集合を T として、 S の要素 s に対して $AvMinDF_{10}(s, T)$ の分布を求める。大量生成型のプログラムと非プログラムの間の場合も同様である。一方、単発プログラム・非プログラム同士の間の場合には、プログラムホスト C 社およびプログラムホスト S 社のそれぞれについて分布を求める。例えば、図 2(a) の場合は、プログラムホスト C 社中の単発プログラム・非プログラムの和集合を S および T として、 S の要素 s に対して $AvMinDF_{10}(s, T)$ の分布を求める。図 2(b), (c), (d) の場合は、プログラムホスト S 社中の単発プログラム・非プログラムを対象として同様の分布を求める (図 2(b), (c), (d) とともに全く同一の分布を示している。)

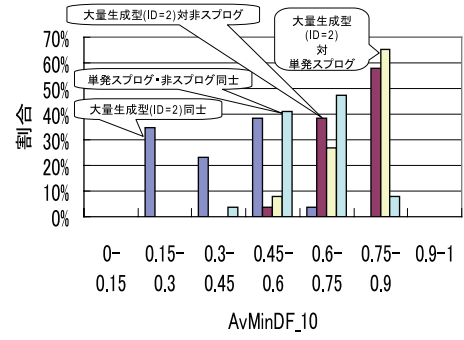
図 2(a)~(d) のいずれの分布を見ても、大量生成型のプログラム同士の分布に対して、それ以外の、大量生成型のプログラムと単発プログラムの間の分布、大量生成型のプログラムと非プログラムの間の分布、および、単発プログラム・非プログラム同士の分布がほぼ分離される傾向にあることが分かる。例えば、図 2(a) では、大量生成型のプログラム同士では、ほぼ全ての大量生成型に対する $AvMinDF_{10}(s, T)$ の値が 0 から 0.15 の範囲に分布しており、それ以外の三種類の分布とはほぼ分離される⁹。また、ID=2 の大量生成型プログラムにおいて、大量生成型のプログラム同士において $AvMinDF_{10}(s, T)$ の値が 0.5 程度のものが存在するが、これらにおいては、プログラムの本文部分は極めて類似しているが、サイドバー部分の違いにより、 $AvMinDF_{10}(s, T)$ が大きくなっていた。

以上の結果から、同一の大量生成型スパマー ID を持つ大量生成型プログラムの HTML 文書から抽出した DOM 系列は、相互に高い類似性を持ち、この類似性を用いることにより、同一プログラムホストにおける単発プログラムや非プログラムとの識別において利用できる可能性があることがわかった。

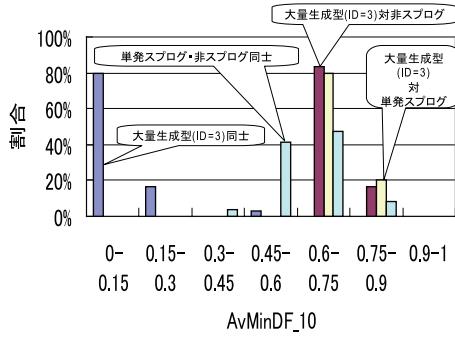
⁹大量生成型のプログラムと単発プログラムの間の分布において、0 から 0.15 の範囲の分布が観測されているが、これらの単発プログラムを精査した結果、人手による判定作業に誤りがあり、ID=1 の大量生成型プログラムとして判定されるべきプログラムであった。



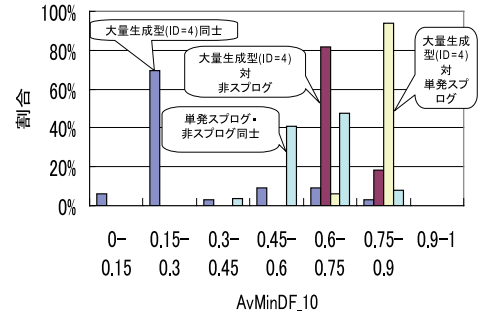
(a) 大量生成型スプログ (ID = 1, C社)



(b) 大量生成型スプログ (ID = 2, S社)



(c) 大量生成型スプログ (ID = 3, S社)



(d) 大量生成型スプログ (ID = 4, S社)

図 2: スプログ・非スプログの DOM 系列の差分の割合の分布

4 スプログ検出のための素性

本節では SVM によるスプログ判定において用いる素性について述べる。

4.1 DOM 系列の差分素性

本論文では、3 節で述べた手法により、HTML 文書から DOM 系列を抽出し、それらの差分の割合を求め、素性値とする。以下では、訓練・評価事例となるスプログまたは非スプログを s とし、3.3 節の手順により $AvMinDF_{10}(s, T)$ を求めた後、

その対数 $\log AvMinDF_{10}(s, T)$ を素性値とする。この手順において、差分を求める対象の集合 T を二通り用意することにより、以下の二種類の素性を設定する。

1. 集合 T を、訓練事例中の全スプログの集合とする。この素性を、DOM 系列差分 (スプログ) 素性とよぶ。
2. 集合 T を、訓練事例中の全大量生成型スプログの集合とする。この素性を、DOM 系列差分 (大量生成型) 素性とよぶ。

4.2 従来からの素性

文献 [16, 17] で使用した素性を従来からの素性とする。

4.2.1 ブラックリスト/ホワイトリスト URL 素性

訓練事例として、スプログ/非スプログが与えられると、その HTML 文書からアウトリンクとなっている URL を抽出する。その中から以下の条件を満たすものを選定し、ホワイトリスト URL とした。

- i) 訓練事例中のスプログの HTML ファイルのいずれにも含まれない URL である。
- ii) 訓練事例中の非スプログの HTML ファイルの中で、2 回以上出現する URL である。

次に、各ホワイトリスト URL u に以下のように重みづけを行い、ホワイトリスト URL 素性の値を算出した。

$$\log \sum_u \left(\begin{array}{c} \text{訓練事例全体の中の} \\ \text{非スプログにおける} \\ u \text{ の総出現頻度} \end{array} \right) \times \left(\begin{array}{c} \text{評価事例} \\ \text{における } u \text{ の} \\ \text{出現頻度} \end{array} \right)$$

4.2.2 名詞句素性

[18, 13, 14] の知見より、スプログおよび非スプログ中における単語の分布には異なりがあり、特定の種類の単語は非スプログよりもスプログに現れやすいということがわかっている。そこで、特定の名詞句とスプログ、非スプログとの間の相関をとらえるために、名詞句素性を導入する。

具体的には、スプログ/非スプログの本文テキストを形態素解析¹⁰した結果から名詞句を抽出し、以下の分割表にしたがって、訓練データ中のスプログ/非スプログにおける名詞句 w の出現頻度を用いて、スプログと名詞句 w との間の ϕ^2 統計量を求めた。

	w	$\neg w$
訓練データ中のスプログ	$\text{freq}(\text{スプログ}, w) = a$	$\text{freq}(\text{スプログ}, \neg w) = b$
訓練データ中の非スプログ	$\text{freq}(\text{非スプログ}, w) = c$	$\text{freq}(\text{非スプログ}, \neg w) = d$

$$\phi^2(\text{スプログ}, w) = \frac{(ad - bc)^2}{(a + b)(a + c)(b + d)(c + d)}$$

また、評価事例に対しては、この名詞句素性の値として以下の式を用いた。

$$\log \sum_w \phi^2(\text{スプログ}, w) \times \left(\begin{array}{c} \text{評価事例における} \\ w \text{ の出現頻度} \end{array} \right)$$

4.2.3 アンカーテキスト名詞句・リンク URL 素性

ブラックリスト/ホワイトリスト URL 素性および名詞句素性よりもより詳細な条件を設定することにより、より有効な性能を示す素性として、アンカーテキストの名詞句およびそのリンク先 URL の (緩い) 組み合わせを用いる。以下、まず、名詞句 w およびブログサイト s に対して、以下の尺度 $AncfB(w, s)$ および $AncfW(w, s)$ を定義する。

$$AncfB(w, s) = \left(\begin{array}{c} \text{ブログサイト } s \text{ 中で名詞句 } w \text{ が} \\ \text{アンカーテキストに含まれ} \\ \text{そのリンク先がブラックリスト} \\ \text{URL もしくは訓練事例中の} \\ \text{スプログとなっている回数} \end{array} \right)$$

$$AncfW(w, s) = \left(\begin{array}{c} \text{ブログサイト } s \text{ 中で名詞句 } w \text{ が} \\ \text{アンカーテキストに含まれ} \\ \text{そのリンク先がホワイトリスト} \\ \text{URL もしくは訓練事例中の} \\ \text{非スプログとなっている回数} \end{array} \right)$$

そして、訓練事例中のスプログ全体の中での総出現頻度 $\sum_s AncfB(w, s)$ が 2 以上であるものを選定し、「ブラックリスト URL へのアウトリンクを持つスプログアンカー

¹⁰日本語形態素解析器 茶釜 (<http://chasen-legacy.sourceforge.jp/>) および ipadic 辞書を用いた。

テキスト名詞句」として、評価事例 t に対して以下の重みを算出し、評価事例 t に対する「ブラックリスト URL へのアウトリンクを持つアンカーテキスト名詞句素性」の値とする。

$$\log \sum_w \left(\sum_{\substack{\text{訓練事例中の} \\ \text{スプログ } s}} AncfB(w, s) \right) \times AncfB(w, t)$$

同様の手順で、訓練事例中のスプログ全体の中での総出現頻度 $\sum_s AncfW(w, s)$ が 2 以上であるものを選定し、「ホワイトリスト URL へのアウトリンクを持つスプログアンカーテキスト名詞句」とする。次に、 w を「ホワイトリスト URL へのアウトリンクを持つスプログアンカーテキスト名詞句」として、評価事例 t に対して以下の重みを算出し、評価事例 t に対する「ホワイトリスト URL へのアウトリンクを持つアンカーテキスト名詞句素性」の値とする。

$$\log \sum_w \left(\sum_{\substack{\text{訓練事例中の} \\ \text{スプログ } s}} AncfW(w, s) \right) \times AncfW(w, t)$$

5 スプログ検出および信頼度尺度

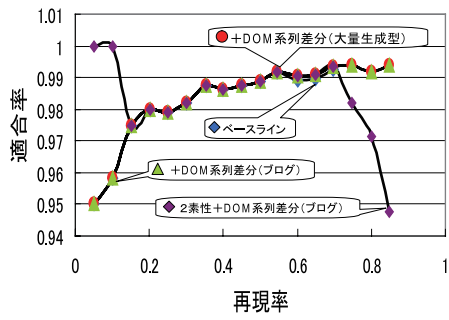
5.1 SVM を用いたスプログ検出

SVM 機械学習を行うためのツールとして、TinySVM (<http://chasen.org/~taku/software/TinySVM/>) を用いた。カーネル関数としては、線形および 2 次多項式を比較し、2 次多項式の方が性能が良かったため、6 節においては、2 次多項式カーネルを用いた場合の結果を示す。また、全ての素性に値がないものは訓練データから除外する。

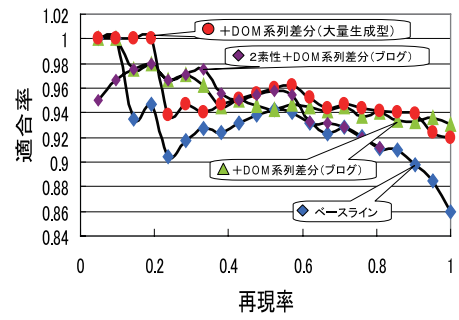
5.2 信頼度尺度

SVM 機械学習での信頼度尺度として、分離平面から各評価事例への距離を用いた [12]¹¹。具体的には、正例として判定される事例に対する分離平面からの距離の下限 LBD_p 、および、負例として判定される事例に対する分離平面からの距離の下限 LBD_n をそれぞれ設定する。

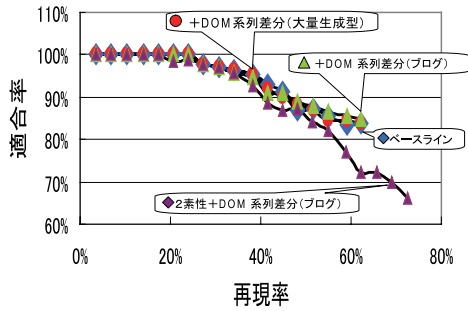
¹¹機械学習および統計的自然言語処理の分野における能動学習 (例えば, [19, 12, 20] 等) 手法の研究事例においては、未知事例のうちで信頼度の低い事例を選別して訓練事例に追加する過程において、信頼度尺度が利用される。



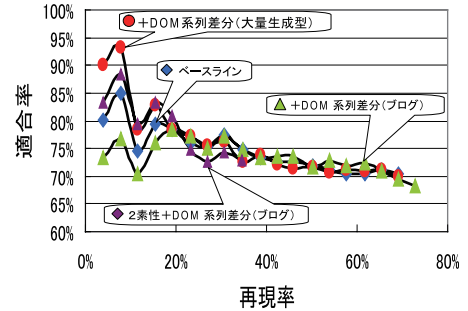
(a-1) スプログ検出性能 (C 社)



(a-2) 非スプログ検出性能 (C 社)

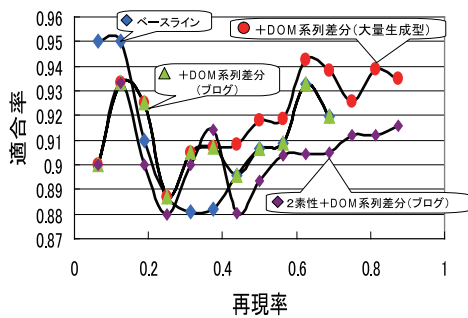


(b-1) スプログ検出性能 (S 社)

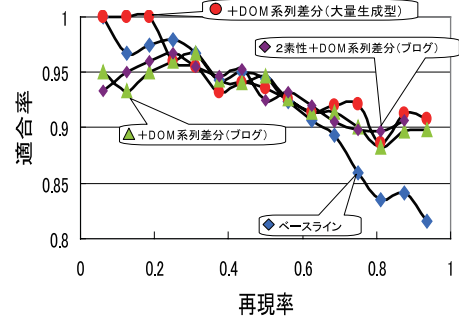


(b-2) 非スプログ検出性能 (S 社)

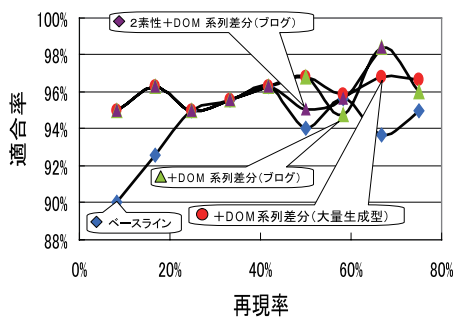
図 3: スプログ/非スプログ検出性能



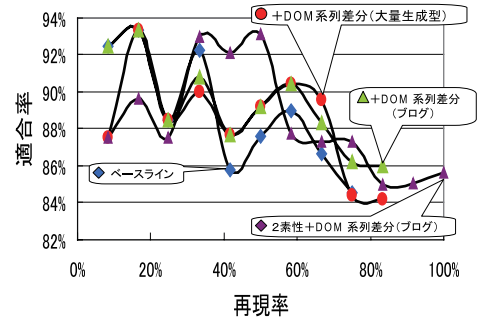
(a-1) 大量生成型スプログ検出性能 (C 社)



(a-2) 単発スプログ・非スプログ検出性能 (C 社)



(b-1) 大量生成型スプログ検出性能 (S 社)



(b-2) 単発スプログ・非スプログ検出性能 (S 社)

図 4: 「大量生成型スプログ」/「単発スプログ・非スプログ」検出性能

6 評価

6.1 評価手順

本節では、表 1(a) に示すデータセットのうち、C 社および S 社のスプログ・非スプログデータセット用いて、プログ宿主別に、スプログ/非スプログ検出性能 (図 3), および「大量生成型スプログ」/「単発スプログ・非スプログ」検出性能 (図 4) の評価を行った。スプログ/非スプログ検出性能の評価においては、C 社を対象とした評価では全 408 プログサイトを、S 社を対象とした評価では全 552 プログサイトを、それぞれ用いた。「大量生成型スプログ」/「単発スプログ・非スプログ」検出性能の評価においては、C 社の場合は、大量生成型スプログ 140 サイトおよび単発スプログ・非スプログ計 140 サイトを用い、S 社の場合は、大量生成型スプログ 90 サイトおよび単発スプログ・非スプログ計 90 サイトを用いた。評価においては、これらのデータセットを用いて、10 分割交差検定を行った。

6.2 評価尺度

以下の説明においては、スプログ/非スプログ検出性能の評価においては、スプログを正例、非スプログを負例とする。また「大量生成型スプログ」/「単発スプログ・非スプログ」検出性能の評価においては、大量生成型スプログを正例、単発スプログ・非スプログを負例とする。

以下では、評価用事例集合、および、正例として判定される事例に対する分離平面からの距離の下限 LBD_p を用いる。分離平面からの距離が LBD_p 以上となる評価事例に対して、正例として判定した場合の再現率、適合率を測定する。そして、 LBD_p を変化させた場合の再現率、適合率の推移をプロットする。同様に、評価用事例集合、および、負例として判定される事例に対する分離平面からの距離の下限 LBD_n を用いて、分離平面からの距離が LBD_n 以上となる評価事例に対して、負例として判定した場合の再現率、適合率を測定する。そして、 LBD_n を変化させた場合の再現率、適合率の推移をプロットする。

6.3 評価結果

図 3 および図 4 において「ベースライン」のプロットは、従来からの素性だけを用いて訓練した分類器の性能を示す。「+DOM 系列差分 (大量生成型)」のプロットは、従来からの素性に DOM 系列差分 (大量生成型) 素性を追加して訓練した分類器の性能を示し、「+DOM 系列差分 (プログ)」のプロットは、従来からの素性に DOM 系

列差分 (プログ) 素性を追加して訓練した分類器の性能を示す。図 3 および図 4 の (a-1), (a-2) において「2 素性 + DOM 系列差分 (プログ)」のプロットは、ホワイトリスト URL 素性、名詞句素性、DOM 系列差分 (プログ) 素性を用いて訓練した分類器の性能である。一方、図 3 および図 4 の (b-1), (b-2) において「2 素性 + DOM 系列差分 (プログ)」のプロットは、ホワイトリスト URL 素性、ブラックリスト URL へのアウトリンクを持つアンカーテキスト名詞句素性、DOM 系列差分 (プログ) 素性を用いて訓練した分類器の性能である。

図 3 において、(a-1) および (b-1) のスプログ検出性能においては、ベースラインからの改善は達成できていないが、(a-2) および (b-2) の非スプログ検出性能において、DOM 系列差分 (大量生成型) 素性を用いた場合、ベースラインからの改善が達成できている。(a-2) においては、DOM 系列差分 (プログ) 素性についても、ベースラインからの性能改善が達成できている。これらの結果から、訓練データ中で大量生成型スプログの情報を付与すれば、未知のスプログ検出の性能を確実に改善できることが分かる。プログ宿主によっては、大量生成型スプログの情報を付与しなくても、未知のスプログ検出の性能を改善できる場合もある。

一方、図 4 においては「大量生成型スプログ」検出性能および「単発スプログ・非スプログ」検出性能の両方において、ベースラインからの改善が達成できている。また、DOM 系列差分 (大量生成型) 素性だけでなく DOM 系列差分 (プログ) 素性においても、ベースラインからの改善が達成できている。この場合は、訓練事例中において大量生成型スプログの情報を用いていることがその大きな理由であると考えられる。

また、図 3 および図 4 を通して「2 素性 + DOM 系列差分 (プログ)」および「+DOM 系列差分 (プログ)」の性能の比較においては、極端に大きな差は見られない。この結果から、DOM 系列差分 (プログ) 素性を用いれば、それ以外の従来からの素性を全て用いなくとも、従来からの素性 2 種類を適切に選定さえすれば十分であることが分かる。

以上の結果より、従来からの素性と比較して、DOM 系列差分 (大量生成型) 素性および DOM 系列差分 (プログ) 素性が高い性能を示すことが確認できた。

7 おわりに

本論文では、同一のスパムプログ作成者が自動的に大量生成したと推測されるスプログの検出において、HTML 構造の類似性が効果的であることを示した。機械学習のひとつである SVM を用いた枠組みによって、

スパムブログの判定を行うタスクを設定し，HTML 構造の類似性を素性として，SVM を用いたスパム検出を行った結果において，スパム検出の性能が向上することを示した．

大量生成したと推測されるスパムの HTML 構造は類似性があるという特徴を用いれば，訓練事例となるスパムを用意しなくても，DOM 系列の差分の割合が極端に小さいブログの組を自動収集することにより，教師なしスパム収集を実現できる可能性がある．現在，この考え方に基づいて，数百万件のブログをクローリングした結果から，スパムの候補を収集し，人手によるスパム・非スパム判定を行う作業を進めており，DOM 系列の差分の割合が極端に小さいブログの組の中にスパムが含まれることを確認済みである．この結果の詳細については，別の機会に報告する予定である．

参考文献

- [1] T. Nanno, T. Fujiki, Y. Suzuki, and M. Okumura. Automatically collecting, monitoring, and mining Japanese weblogs. In *WWW Alt. '04: Proc. 13th WWW Conf. Alternate Track Papers & Posters*, pp. 320–321, 2004.
- [2] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *Proc. 1st AIRWeb*, pp. 39–47, 2005.
- [3] *Wikipedia*, *Spam blog*. http://en.wikipedia.org/wiki/Spam_blog.
- [4] P. Kolari, A. Joshi, and T. Finin. Characterizing the splogosphere. In *Proc. 3rd Ann. Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*, 2006.
- [5] C. Macdonald and I. Ounis. The TREC Blogs06 collection : Creating and analysing a blog test collection. Technical Report TR-2006-224, University of Glasgow, Department of Computing Science, 2006.
- [6] P. Kolari, T. Finin, and A. Joshi. Spam in blogs and social media. In *Tutorial at ICWSM*, 2007.
- [7] Y.-R. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. L. Tseng. Splog detection using self-similarity analysis on blog temporal dynamics. In *Proc. 3rd AIRWeb*, pp. 1–8, 2007.
- [8] 石田和成. スパムブログの推定と抽出. 日本データベース学会 Letters, Vol. 6, No. 4, pp. 37–40, 2008.
- [9] 石田和成. 共起クラスターシードと連鎖的抽出にもとづくスパムブログのフィルタリング. Web とデータベースに関するフォーラム (WebDB Forum)2008 論文集. 情報処理学会, 2008.
- [10] P. Kolari, T. Finin, and A. Joshi. SVMs for the Blogosphere: Blog identification and Splog detection. In *Proc. 2006 AAAI Spring Symp. Computational Approaches to Analyzing Weblogs*, pp. 92–99, 2006.
- [11] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [12] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proc. 17th ICML*, pp. 999–1006, 2000.
- [13] 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 論文集, 2008.
- [14] Y. Sato, T. Utsuro, T. Fukuhara, Y. Kawada, Y. Murakami, H. Nakagawa, and N. Kando. Analysing features of Japanese splogs and characteristics of keywords. In *Proc. 4th AIRWeb*, pp. 33–40, 2008.
- [15] 吉田光男, 山本幹雄. 教師情報を必要としないニュースページ群からのコンテンツ自動抽出. 日本データベース学会論文誌, Vol. 8, No. 1, pp. 29–34, 2009.
- [16] 片山太一, 佐藤有記, 宇津呂武仁, 芳中隆幸, 河田容英, 福原知宏. 機械学習を用いたスパムブログ検出における信頼度の利用. データ工学と情報マネジメントに関するフォーラム—DEIM フォーラム— 論文集, 2009.
- [17] T. Katayama, Y. Sato, T. Utsuro, T. Yoshinaka, Y. Kawada, and T. Fukuhara. An empirical study on selective sampling in active learning for splog detection. In *Proc. 5th AIRWeb*, pp. 29–36, April 2009.
- [18] Y.M. Wang, M. Ma, Y. Niu, and H. Chen. Spam double-funnel: Connecting web spammers with advertisers. In *Proc. 16th WWW*, pp. 291–300, 2007.
- [19] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proc. 17th SIGIR*, pp. 3–12, 1994.
- [20] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th ICML*, pp. 839–846, 2000.