スプログの調査と実システムにおける判別手法*

高橋 哲朗† 野田 雄也† 岩倉 友哉††
† ニフティ株式会社
†† 株式会社富士通研究所
takahashi.tetsuro@nifty.co.jp, noda.yuya@nifty.co.jp
iwakura.tomoya@jp.fujitsu.com

1 はじめに

ブログの情報は市場調査や新商品開発といったマーケティングの目的に有用な情報であり，すでにいくつもの実サービスが提供されている[1]．また自然言語処理技術の適用分野としても注目されており，この数年の間評判分析などへの応用が進められている [2].

一方で，ブログの中にはスプログ (スパムブログ) が大量に混入しており，ブログの分析結果に悪影響を与えている．ブログを価値ある情報として使うためには，このスプログを取り除く必要がある．本稿ではまずスプログの種類と割り合いについて行なった調査の結果を報告する．そしてその結果から，スプログの種類の一つである引用型スプログの判定手法の提案と評価実験の報告を行なう.

2 スプログの現状調査

主要なブログサイト 27 サイトを含め，日本語で書かれたほぼ全ブログを対象とし，そこからランダムに抽出したブログ記事 3719 件を人手により分析し，スプログの判定を行なった．その結果を 表 1 に示す.

表 1: スプログの割り合い

スプログ種類	件数	割り合い (%)
引用型	449	12.1
アフィリエイト	249	6.7
アダルト	191	5.1
ワードサラダ	148	4.0
ギャンブル・金融	107	2.9
ショッピング	39	1.0
スプログ記事	909	24.4
非スプログ記事	2810	75.6
計	3719	100.0

スプログの種類はそれぞれ以下のように定義し分類を行なった.

引用型 ニュース記事などをコピーし生成した記事.

アフィリエイト アフィリエイトリンク (画像) のみで構成され，オリジナルの記述をほとんど含まない.

アダルト 卑猥な文書が書かれている記事

ワードサラダ 複数の情報源から単語を抽出し，それらを並べ替え生成した記事.

ギャンブル・金融 ギャンブルや，FX で儲ける方法などの話題に閉じている記事.

ショッピング 商品が列挙されたショピングサイト

なお，一つの記事に対して複数のスプログ種類を割り当てることがあるため，各スプログ種類の件数の合計値とスプログ記事数は一致しない.

本稿ではここで分類したスプログのうち，引用型スプログの検出を行なう手法を提案する．引用型スプログは主にニュース記事から引用された記事を元としたブログであり，さらにその引用部分の前後に，以下の例に示すような引用部分とは関係のない話題や，引用部分への感想のように生成された文章が置かれている.

- ではそろそろ気持ちを切り替えて、ちょっと外出してきます。
- 最近 { 記事タイトル } が気になります。

3 提案手法

引用型スプログの判定は，その記事の性質から，単独の記事そのものの特徴のみから行なうことは難しく，ある記事集合を対象として，その中から引用型スプログの集合を見付ける必要がある．本稿では引用型スプログ検出を，類似文書集合を探す問題ととらえる.

記事数を M としたときに，単純に類似度計算を行なうと，$O(M^2)$ の計算量が必要となる．そのためブログ記事など日々生成される莫大な記事集合を扱うためには，効率的な計算手法が必要となる.

そこで本稿では転置インデックス形式を応用した，高速に類似記事を判定する手法を提案する．この手法では，まず記事間のそれぞれの文の類似度を計算し，その後各記事間の文の類似度を用いて最終的な類似記事判定を行なう.

$\{\mathbf{x_1}, .., \mathbf{x_N}\}$ を類似度計算を行なう N 文とする．各文 $\mathbf{x_i}$ $(1 \leq i \leq N)$ は，異なり数 $|\mathbf{x_i}|$ の単語の集合

*Splog detection using a similarity based approach.
TAKAHASHI Tetsuro†, NODA Yuya†, IWAKURA Tomoya††.
† Nifty Corporation
†† Fujitsu Laboratories LTD.

[1]Kizashi, SHOOTI, blogViz センサー, goo ブログ評判分析, Buzz Marketing Solution

1 入力は $\{\mathbf{x_1}, .., \mathbf{x_N}\}$，類似度閾値 s.
$FQ(i,j)$ を文 $\mathbf{x_i}$,$\mathbf{x_j}$ に共通して出現する単語数とし 0 で初期化する.

2 入力の各文 $\mathbf{x_i}$ $(1 \leq i \leq N)$ の各単語をキーとし，値に「文 id i」を登録するような転置インデックスを作成.

3 各単語に対応する値 (文 id のリスト) をその id の文のサイズに基き昇順にソートする.

4 インデックスに登録された各単語 w について処理

4-1 w の値の参照位置 $i = 1$ とする.

4-2 for $(j = i + 1; j \leq |\mathbf{x_i}|/s; j++)$ $FQ(i,j)$++

4-3 i が w の値の最後の位置であれば終了．それ以外は，$i = i + 1$ とし，**4-2** へ戻る.

5 $s \leq FQ(i,j)/|\mathbf{x_j}|$ $(|\mathbf{x_i}| \leq |\mathbf{x_j}|)$ を見たす $(\mathbf{x_i}$,$\mathbf{x_j})$ を類似文ペアとして出力.

図 1: 類似度計算手法

$\{x_{i,1}, .., x_{i,|\mathbf{x_i}|}\}$ で表現されるとする．$|\mathbf{x_i}|$ を文サイズと呼ぶこととする．このとき本タスクの目的は，類似度 $SIM(\mathbf{x_i}, \mathbf{x_j})$ $(1 \leq i, j \leq N, i \neq$j) が閾値 s 以上となる文のペア集合を発見することである．SIM は，文間の類似度を計算する関数であり，今回の実験では，

$$SIM(\mathbf{x_i}, \mathbf{x_j}) = \frac{\sum_{k=1}^{|\mathbf{x_i}|} x_{i,k} \subseteq \mathbf{x_j}}{|\mathbf{x_j}|} \quad (1)$$

とする．ここで $w \subseteq \mathbf{x_i}$ は $\mathbf{x_i}$ が単語 w を含んでいる場合は 1，それ以外は 0 を返すとし，$|\mathbf{x_i}| \leq |\mathbf{x_j}|$ とする．今回の実験で，類似度 $SIM(\mathbf{x_i}, \mathbf{x_j})$ は $0 \leq s \leq 1$ となる.

式 (1) から，$\mathbf{x_i}$ との類似度が閾値 s 以上となる文 $\mathbf{x_j}$ を見つけるためには，$\mathbf{x_i}$ に含まれる単語の異なり数 $|\mathbf{x_i}|$ が $|\mathbf{x_j}|$ の s 以上 $(s \leq |\mathbf{x_i}|/|\mathbf{x_j}|)$ が最低条件であることから，

$$|\mathbf{x_j}| \leq \frac{|\mathbf{x_i}|}{s} \quad (2)$$

を満す文 $\mathbf{x_j}$ だけを計算すれば良いことがわかる．この条件を基に，類似度計算手法を提案する．アルゴリズム概要を図 1 に示す.

図 1 の **3** では，式 (2) に基づいた枝刈りを行なえるように文サイズに基づきソートを行なう．これにより **4-2** における処理の範囲を，式 (2) を満たす i, j のみに制限することが可能となる.

ここで提案した手法では転置インデックスを作成するコストが必要となるが，ブログの検索サービスを行なっているような場合では，検索のために転移インデックスは作られており，それを利用できる．そのため，このコストは問題にならないことが多い.

4 評価実験

4.1 問題設定

入力とする記事集合において，それぞれの記事からは本文テキスト部分があらかじめ抽出されており，文単位の区切りが行なわれているものとする．この入力に対し，類似している記事集合を発見する問題をここでは解く．2 つの記事を比較し，記事中の文のうち類似している文の割り合いが 0.3 以上の場合に類似記事とした.

文の類似度については 3 節で述べた手法を用い，閾値 s は 0.8 とした．つまり文サイズ大きい方の文の単語数のうち重複している単語の割り合いが 80% 以上であった場合に類似した文であると見なす.

これは「類似記事」の定義としては粗い定義であるが，上記のパラメータを適切に設定することにより，引用型スプログはこの記事集合に含まれる．類似記事集合の大きさはすべての文の組み合わせ数と比較すると非常に小さいため，上記の定義により引用型スプログの候補を見付けた後に，より厳密な手法により引用型スプログを判定することは現実的である.

提案手法で述べた「単語」については，形態素解析結果から品詞により内容語のみを選択し，それらの表層文字列を用いた.

実験で用いる記事集合については，実験結果がスプログの含まれる割り合いに依存することが考えられたため，以下の 2 種類のデータを用いた.

window ある時間間隔に投稿された記事集合

query 特定のクエリを含む記事集合

window 内の記事の内容はランダムであると想定できるのに対し，**query** 内の記事は特定の話題に偏っている．そのためこの話題に関する記事が引用元として使われていた場合，**query** はスプログを多く含む可能性がある.

これらの記事集合から 1000 件を抽出し，評価のため以下の 3 種のアルゴリズムにより類似記事の発見を行なった.

(A1) 単純な類似度判定
記事集合から 2 つの記事の組み合わせを抽出し類似度判定を行なう

(A2) 提案手法 (枝刈りなし)

(A3) 提案手法 (枝刈りあり)

表 2: 類似記事発見の処理時間 (sec)

データ	手法	入力記事数 100	200	500	1000
window	**(A1)**	11.728	39.356	217.45	872.88
	(A2)	0.841	2.336	11.35	49.21
	(A3)	0.737	0.781	2.83	10.22
query	**(A1)**	30.267	151.658	751.65	2627.25
	(A2)	2.904	12.513	68.16	271.56
	(A3)	0.758	2.763	14.77	55.65

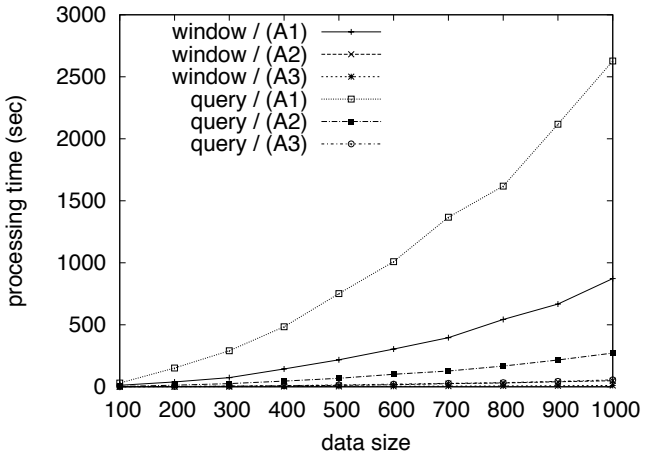

図 2: 入力データサイズと類似記事発見の処理時間

4.2 提案手法による速度向上

実験結果を 表 2 と図 2 に示す． この結果から，提案手法による処理速度の向上が明らかである．**(A1)** と **(A3)** を比較すると入力記事数が 1000 のときに約 47 倍 (**query**)〜85 倍 (**window**) の速度の差があった．

検出されたスプログの記事数は，入力記事数が 1000 のときに 163 件 (**window**), 300 件 (**query**) であった．データの種類によってスプログの記事数も異なり，それに依存して処理時間にも **(A3)** で約 5 倍と大きな差がある．スプログの記事数が及ぼす処理時間への影響は転置インデックスを用いる手法の方が単純な手法 **(A1)** よりも大きい．

(A2) と **(A3)** の比較から枝刈りの効果も表われており，入力記事数が 1000 のときに約 5 倍程度の速度向上となっていた．3 節で述べたように，$|\mathbf{x_i}| \leq |\mathbf{x_j}|$ と式 (2)，閾値 $s = 0.8$ から，探索空間は

$$|\mathbf{x_i}| \leq |\mathbf{x_j}| \leq 1.25 * |\mathbf{x_i}| \tag{3}$$

に制限される．すなわち x_i の長さの 1.25 倍より長い文は比較対象とならずそのため処理速度が向上している．

4.3 DF による使用単語の選択

提案手法の性質から，DF の高い単語は類似度計算において長い処理時間を必要とする．そこで，使用する単語のうち DF の高い方から一定の割り合いの単語を削除し，**(A3)** のアルゴリズムによる実験を行なった．入力記事数は 1000 記事とした．

表 3: 単語選択による処理時間 (sec)

データ	DF に基づく使用単語の削除 (%) 0.0	0.1	0.2	0.3	0.5	0.9
window	10.22	3.68	2.70	2.40	2.20	1.56
query	55.65	15.77	10.75	8.70	6.94	4.69

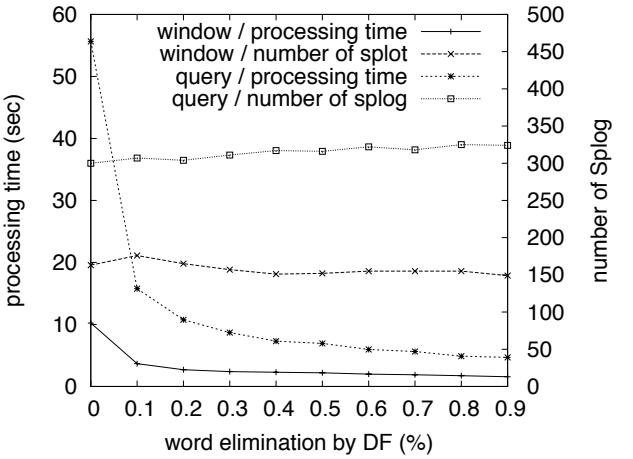

図 3: 単語選択による処理時間と抽出件数

実験結果を表 3 と図 3 に示す． DF の上位 0.2% の単語を削除した場合，**window** で 3.8 倍，**query** で 5.2 倍の速度の向上が見られる．

提案手法では粗い定義に基づきスプログの候補を抽出している．そのため問題となるのは，誤抽出よりも抽出漏れであり，抽出数が減るのは問題である．そこで検出するスプログ数を見てみたところ，図 3 に示すように単語を削除しても大きな変化はなかった．この実験では使用する単語を DF が高い語から順に削っているが，これらの単語は多くの文に共通に含まれており，そのため類似性の判定には大きな影響は与えないと考えられる．実際に削除した単語を見ると,「する，いる，の，今日」などの単語であり，これらの単語が使われなかったために類似文書と見なされなくなるような記事は類似しているとは言い難い．

5 関連研究

スプログ判定に関する関連研究は，主に以下の 3 種類にまとめられる．

5.1 教師あり機械学習による手法

北村ら [7] や Jindal ら [3] は，スプログ判定のタスクを二値分類問題として定式化し，教師あり機械学習により解いている．これらの手法では，特定の文字列が繰り返されている SEO 目的の記事などの特殊な記事を発見することを目的としている．そのため素性として，

文長，単語の重複回数，評価表現の数，内容語，などを用いて分類を行なっている．

この手法はアダルト系やギャンブル系などのスプログのように，記事そのものが何らかの特徴を持つ場合には有効であるが，ワードサラダや引用型スプログに対しては効果がないと考えられる．

5.2 単語の出現分布による手法

Misra ら [4] は，通常の文書は少数の topic しか述べていないという仮説に基づき，文書の topic distribution から得られるエントロピーに基づいてワードサラダ型の文書の判定を行なっている．

Uemura ら [5] は，ワードサラダ型のスプログは，少数のテンプレートから生成されているという仮説に基づき，document complexity というエントピーに類する値によって，スプログかどうかの判別を行なう．

これらの手法は教師データを必要とせず，またワードサラダ型のスプログには効果的であると考えられる．しかし，引用スプログは，正しい文章をそのまま引用しているため，スプログの場合には上述の仮説が成り立たないと考えられる．

5.3 類似度による手法

Yoshida ら [6] は，電子メールのスパム判別において，類似する文面を持つメールを数えあげ，閾値以上の類似メール集合をスパム検出する手法を提案している．この手法では，メール本文の先頭から生成される N 個の L-gram を選択し，それぞれの hash 値を使って表現される特徴ベクトルを使うことにより類似メールの判別を行なっている．

この手法での特徴ベクトル生成は，スパムメールのように，文面全体が類似する集合を発見する場合であれば，先頭からの部分文字列だけでも対応できると考えられる．しかし，引用型スプログの判別では，本文以外にも広告などが挿入される，引用部分以外の記述もある，などの特徴を考慮する必要があり，先頭からの部分文字列だけを利用した判別は難しいと考えられる．

また，Haider ら [1] は，教師ありクラスタリング手法に基づき類似するメールをまとめあげた後，スパム判別を行なう手法を提案している．この手法では，クラスタ生成のための計算量の削減のために，メールを stream として扱い，それぞれのメールを決定的にクラスタに割り当てていく方法を提案している．その後，判別されたメール集合に対して，教師あり学習手法に基づいたスパム判別を行う．

この手法では，最終的なクラスタ生成のための計算量は，決定的に生成する手法により削減されてはいるが，各メールの類似度計算回数という観点では通常の手法と同等である．本手法では，教師データの作成が必要なく，類似度計算式に基づき探索空間の枝刈りを行なっている点が異なる．

6 まとめ

本稿では，まず日本語で書かれるスプログの分類を行なった．その結果引用型スプログの割り合いが多く，またこの種のスプログについては先行研究でも有効な手段が提案されていなかった．そこで本稿では引用型スプログに対する検出方法についての提案と評価を行なった．(a) 転置インデックスを用い，(b) 類似度の閾値による枝刈りを行ない，(c) 類似度計算に使用する単語を DF により選択する，という特徴を持った手法を提案し，それぞれにおいて速度向上を確認した．またこれらを総合的に用いたとき，単純に処理した場合と比較したときと比べておよそ 100 倍の速度向上ができた．

実システムで運用していくためには 1 日当たり約 100 万記事を処理できなければならない．引用型スプログを判定するための記事集合としてどれくらいの時間範囲の記事を対象としなければならないかを決める必要があり，今後の課題となる．

また今回提案した手法では，入力としてバッチで与えられた記事集合から類似記事集合を探す問題を解くものであったが，類似記事に関する情報を蓄積しておき，ストリームとして入力されるブログ記事を特定できるよう今後拡張していく．

参 考 文 献

[1] Peter Haider, Ulf Brefeld, and Tobias Scheffer. Supervised clustering of streaming data for email batch detection. In *International Conference on Machine Learning (ICML-2007)*, 2007.

[2] 乾孝司, 奥村学. テキストを対象とした評価情報の分析に関する研究動向. 自然言語処理, Vol. 13, No. 3, pp. 201–241, 2006.

[3] Nitin Jindal and Bing Liu. Opinion spam and analysis. In *1st ACM International Conference on Web Search and Data Mining(WSDM-2008)*, 2008.

[4] Hemant Misra, Olivier Cappe, and Francois Yvon. Using lda to detect semantically incoherent documents. In *Conference on Computational Natural Language Learning (CoNLL)*, 2008.

[5] Takashi Uemura, Daisuke Ikeda, and Hiroki Arimura. Unsupervised spam detection by document complexity estimation. In *the 11th international conference on discovery science(DS2008)*, pp. 319–331, 2008.

[6] Kenichi Yoshida, Fuminori Adachi, Takashi Washio, Hiroshi Motoda, Teruaki Homma, Akihiro Nakashima, Hiromitsu Fujikawa, and Katsuyuki Yamazaki. Density-based spam detector. *IEICE transactions on information and systems*, Vol. E87-D, No. 12, pp. 2678–2688, 2004.

[7] 北村順平, 青野雅樹. ウェブサイト間の類似度を用いたウェブスパムの検出. 自然言語処理研究会 No.113 2008-NL-188, pp. 45–50, 2008.