# 順序保存ダイジェスト法による Web ページ間の部分複製検出

櫻井 俊之† 松尾 義博† 菊井 玄一郎†

†日本電信電話株式会社 NTT サイバースペース研究所
〒239-0847 神奈川県横須賀市光の丘 1-1
E-mail: †{sakurai.toshiyuki, matsuo.yoshihiro, kikui.genichiro}@lab.ntt.co.jp

**あらまし** Web ページの一部を複製して生成されるスパムブログ等の「部分複製コンテンツ」が，検索エンジンのユーザ満足度の低下などの問題を引き起こしている．

そこで我々は，部分複製を効率的に検出する手法を提案する．提案手法は Web ページを「セグメント」と呼ばれる単位に分割し，各セグメントをハッシュ値に変換し，ハッシュ値を連結し， Suffix Array に格納することにより，部分複製の高速な検索を可能にするものである（順序保存ダイジェスト法）．提案手法に対して，検出速度，精度，再現率の評価実験を行った結果，有効性を確認した．

**キーワード** 複製，スパムブログ，類似文書検出，Near-Duplicate Detection

# Sequence Preserving Digest: A New Method for Efficiently Detecting Partially Duplicated Web Pages

Toshiyuki SAKURAI† Yoshihiro MATSUO† and Genichiro KIKUI†

†NTT Cyber Space Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka-shi, Kanagawa, 239-0847 Japan
E-mail: †{sakurai.toshiyuki, matsuo.yoshihiro, kikui.genichiro}@lab.ntt.co.jp

**Abstract** We propose a new method for efficiently detecting partially duplicated web pages or weblogs, which often case adverse effects on search results. Our method converts each web page into a digest, which is a concatenated string of hash values, each of which derived from a segment, currently a sentence, of the original web page. Since the digest preserves the order of segments in the original web page, partial match of two digests means partial match of their original documents. Our method stores digests into suffix arrays in order to efficiently detect partial match. Experimental results show our method detects partial match correctly while still keeping space and time efficiency.

**Keyword** Duplicate, weblog, Near-Duplicate Detection

## 1. はじめに

近年，CMS（Content Management System）やブログサービスの発展に伴い，誰でも気軽に情報発信を行うことが可能になった．また，クリック報酬型広告やアフィリエイトプログラムの登場により，ブログから利益を得ることが可能になった．これらの要因により，特定サイトへの誘導や成果報酬を目的としたスパムブログが大量発生するようになり，検索エンジンの検索結果上位に大量に表示されるようになった結果，検索エンジンのユーザ満足度を低下させるようになった．スパムブログは検索結果上位に自身を表示させるために，Wikipedia やニュースサイトの記事，EC サイトの商品紹介記事，企業等のプレスリリース，歌詞等の検索ランキング上位のキーワードを含む Web ページの文章を複製して生成される複製コンテンツであることが多い．そのため，複製コンテンツを効率良く検出し，それらを排除する技術が求められている．

そこで，本論文では従来技術では検出が難しかった部分複製を検出可能な手法を提案し，提案手法をシステム化し，その評価結果を述べる．

以下 2 章では複製コンテンツ検出技術の課題について述べ，3 章では関連研究について述べる．4 章では提案手法の概要について述べ，5 章では提案手法を用いたシステムの構成について述べ，6 章では 5 章で述べたシステムを用いた評価実験の結果について述べる．

## 2. 複製コンテンツ検出技術の課題

本章では，複製コンテンツの種別，検出技術に対する課題について述べる．

### 2.1. 検出対象となる複製コンテンツの種別

複製コンテンツは図 1 に示すとおり，全体複製，包

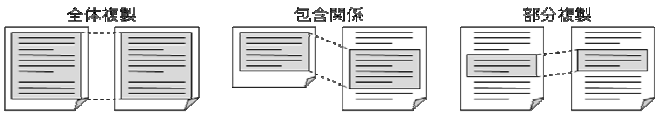


図 1. 複製コンテンツの種類

含関係，部分複製の 3 種類に分類される

全体複製とは，Web ページ全体が他の Web ページからの複製であるものを指す．包含関係とは，ある Web ページ全体を別の Web ページが包含しているものを指す．部分複製とは，ある Web ページの一部が，別の Web ページの一部として複製されているものを指す．

全体複製や包含関係の検出に関しては 3 章で述べるように広く研究が行われており，検出手法がいくつか提案されている．しかし，部分複製に関しては従来手法では検出が困難であるため，部分複製の検出が可能な手法を検討する必要がある．

### 2.2. 要求される文書規模, 検出速度の問題

スパムブログ等の複製コンテンツは前述の通り，Wikipedia やニュースサイトの記事等を複製して記事を生成している．そのため，これらの複製コンテンツを検出するためには，これらの複製元となりやすい文書（原典文書）を収集し，複製検出のための原典として利用すればよい．

しかし，原典文書は Wikipedia だけでも 55 万記事（2009 年 1 月 8 日現在）存在し，Wikipedia の過去の版や各種ニュースサイト，各種 EC サイト，企業のプレスリリースまでを網羅しようとすると数千万記事にまで膨らむ．そのため，複製コンテンツ検出を効率良く行うためには，何らかの方法でデータ量を圧縮する必要がある．

また，複製検出対象となるブログも膨大な量であるため，検出速度の高速化も重要な課題である．

## 3. 関連研究

本章では，複製コンテンツ検出における関連研究について述べる．

複製コンテンツ検出に関する方法としては，類似文書検出技術（Near-duplicate Detection）を用いる方法と，検索エンジンを用いる方法の 2 つがある．

類似文書検出技術には，Charikar の手法，Broder らの手法，Bayordo らの手法がある．

Charikar は文書毎に simhash と呼ばれる特殊なハッシュ値を計算し，文書間の類似度を simhash のハミング距離から算出する手法を提案した[1]．この手法における simhash のハミング距離は，文書間のコサイン距離を近似しているため，単語ベクトルや概念ベクトル等における文書全体の類似性を判定することはできるが，包含関係や部分複製を検出することができない．

Broder らは文書から得られた N-gram を基にした Shingle と呼ばれるものを定義し，2 つの文書間の類似度を Jaccard 係数により算出する手法を提案した[2]．この手法では，N-gram が一定割合以上一致した文書の組み合わせを類似と判定する．この方法では，全体複製だけでなく，包含関係も検出可能であるが，2 つの文書間の部分複製を検出することができない．

Charikar や Broder らの手法は文書長に対して複製部分が少ないと，類似度が低く見積もられてしまうため，複製を含んでいても複製と検出できない場合がある．また，これらの手法はともに文字列の出現順序を意識しない方法であるため，似通った単語分布で異なる内容の文書では類似度が高くなってしまい，複製でないものまで複製と誤検出してしまう問題がある．

Bayordo らは事前に閾値を設定し，閾値以上の文書ペアのみの類似度を計算する手法を提案した[3]．この手法は類似度の高そうな候補を絞り込むことによって計算量を削減する手法であるが，候補を絞り込む処理における類似度関数として，コサイン類似度や Jaccard 係数等を用いるため，Charikar や Broder らの手法と同様の問題をはらんでいる．

検索エンジンを用いる手法に関しては，田代らの手法がある．田代らは，検出対象のテキストから N-gram を生成し，各 N-gram を検索キーワードとして，複数の検索エンジンに入力し，各 N-gram における検索結果の和集合から複製コンテンツかどうかの判定を行う手法となっている[4]．この手法では，全体複製，包含関係，部分複製の 3 種類の複製コンテンツを検出可能な手法であるが，検索エンジンに複数回検索キーワードを入力する手法であるため，Web 全体のような大規模な複製コンテンツ検出には，速度や規模の面で向いていないと考えられる．

## 4. 提案手法の概要

本章では，従来技術の問題点を解決し，Web 規模にも適用可能な複製コンテンツ検出手法について述べる．

### 4.1. 複製を検出可能なデータ圧縮方法について

高速な複製コンテンツ検出を実現するためには，何らかのデータ圧縮方法が必要であり，そのデータ圧縮方法は部分文字列の比較ができなければならない．

また，複製は一般的に文もしくは文に準じた単位で行われる．そうしないと複製部分が日本語としての意味をなさない単なる文字列になってしまうためである．

そこで我々は，上記の性質を利用して，部分文字列の比較を行うことが可能なデータ圧縮方法を提案する．具体的には，複製されやすいであろう部分文字列の単位をセグメントと定義し，文書をセグメントに分割する．次に，セグメント毎にハッシュ関数を用いてフィンガープリントを生成する．フィンガープリントをセ

グメントの出現順序を保持したまま連結し，これをダイジェストとする．

上記により，セグメントの出現順序は，フィンガープリントの出現順序で表現されるため，データ量を圧縮しつつも,セグメントの出現順序は保持されている．これにより複製の検出は，原典文書と複製検出対象文書のダイジェストを比較し，ダイジェスト間の共通部分を検出する問題となる．

### 4.2. 高速な複製検出手法

ダイジェスト間の共通部分を検出する問題は，文字列間の共通部分を検出する問題と等価である．文字列間の共通部分を総当たりで見つけようとすると，組み合わせ数が爆発してしまい現実的ではない．そこで我々は文字列の一致部分を高速に検索することが可能なデータ構造である Suffix Array[5]を用いることとした. Suffix Array は事前に検索用インデックスを作成しておく必要があるが，あらゆる部分文字列の検索が可能であり，シンプルなアルゴリズムである．

## 5. 複製検出システム概要

本章では，前章の手法を用いた複製検出システムの概要について述べる．

図 2 に本システムの機能ブロックの構成を示す．

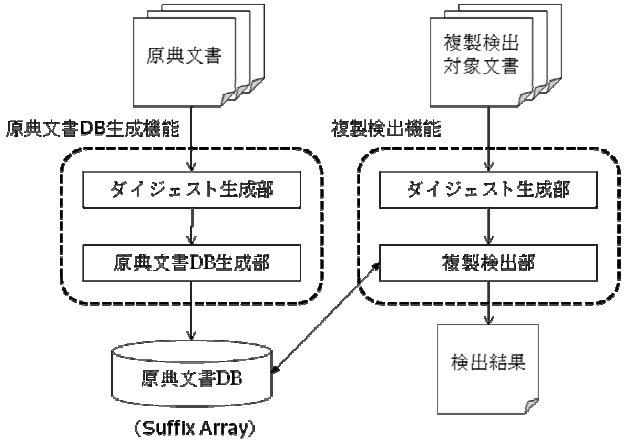


図２．複製検出システムの機能ブロック図

本システムは, 事前に原典文書集合から Suffix Array の検索用インデックス（原典文書 DB）を生成する機能（原典文書 DB 生成機能）と，複製検出対象文書を入力し，原典文書からの複製が含まれていないかを検出する機能（複製検出機能）の 2 つの機能からなる．また，上記機能は以下の 3 つの機能部からなる．

- ダイジェスト生成部
- 原典文書 DB 生成部
- 複製検出部

以下で，各機能部の詳細について述べる．

### 5.1. ダイジェスト生成部

図 3 にダイジェスト生成部の処理の流れを示す．

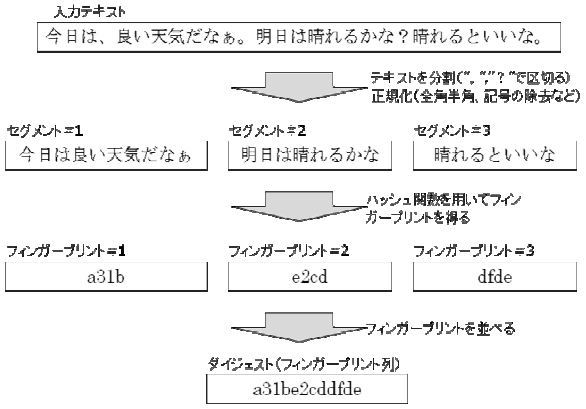


図３．ダイジェスト生成部

まず，入力された文書をセグメントに分割し，各セグメントに対して全角半角正規化，記号類の除去などの各種正規化を行う．複製の中には，記号の追加や全角半角の変更など，複製元文章に微妙な差異を付けたものが存在するため，微妙な差異を吸収するために上記の処理を行っている．

次に，セグメント長が短いものは無いものとして扱う（足切り）. これは,「おはよう」,「おやすみ」など，セグメント長が短いものは，偶然の一致である確率が高く，複製でない場合が多いため，それを取り除くために上記の処理を行っている．

最後に，ハッシュ関数を用いて，セグメントに対するフィンガープリントを計算し，フィンガープリントをセグメントの出現順序を保持したまま連結し，ダイジェストを得る．

本システムではハッシュ関数として Rabin Fingerprint[6]の 32bit ハッシュを採用した．

### 5.2. 原典文書 DB 生成部

ダイジェスト生成部から出力されたダイジェストと文書 ID の組み合わせをストアし，インデックスをフィンガープリント長単位とした Suffix Array を構築する．

### 5.3. 複製検出部

図 4 に複製検出部の処理の流れを示す．

複製検出対象文書が入力されると，まず，ダイジェスト生成部を用いてダイジェストを得る．次に，ダイジェストをクエリとして，原典文書 DB に対して，前方最長一致のエントリを検索する．もし，前方最長一致のエントリが見つかった場合は，一致長をカウント

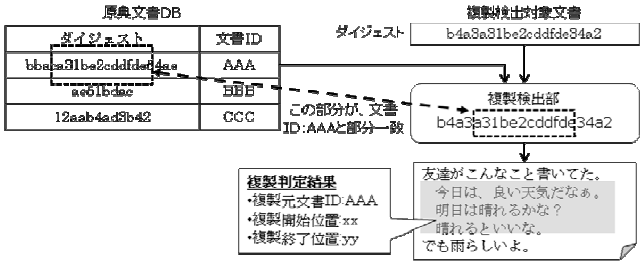


図４．複製検出部

し，事前に設定した最短一致長以上一致していれば複製とみなし，複製部分を確定する．

## 6. 評価実験

本章では，実機を用いた実験結果について述べる．本評価実験では，提案手法の原典文書規模と照合速度の評価，精度の評価，再現率の評価の 3 つを行った．

なお，本評価実験では，セグメント長 5 文字未満を足切りし，最短一致長は 3 セグメントとした．

以降で，各実験の詳細について述べる．

### 6.1. 原典文書規模と照合速度

原典文書集合にどれだけの文書が格納可能かを計るため，文書数を変化させながら原典文書 DB を作成する実験を行った．

実験には，CPU:Intel Xeon E3070（2.66GHz），メモリ:4GB の計算機を用い，原典文書として，Wikipedia 50 万記事（http://download.wikimedia.org/jawiki/ より 20080726 版をダウンロード）とブログ 2000 万記事を用いた．

また，原典文書 DB のサイズが大きくなるに従って，照合速度がどれほど低下するかを確認するため，文書規模毎に照合速度の計測も合わせて行った．

実験結果を図 5 に示す．

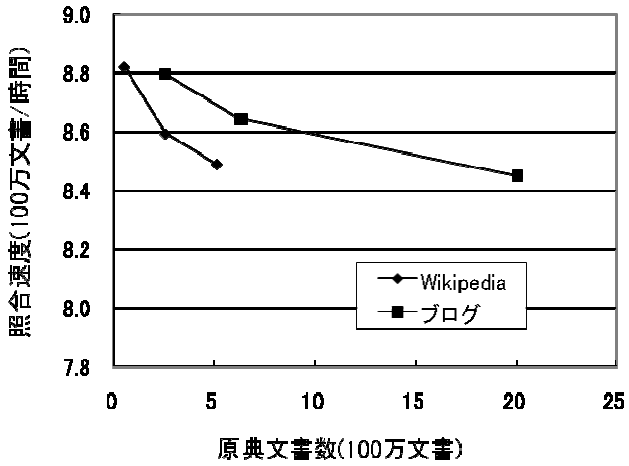


図５．文書規模と照合速度

実験結果より，4GB 程度のメモリであっても約 2000 万文書を格納可能であることがわかった．

また，照合速度に関しては，2000 万文書をインデックスしていた場合でも，840 万文書/時間の速度での照合が可能であった．

### 6.2. 精度

ブログにどれだけ Wikipedia からの複製が含まれているかを判定し，判定結果がどれだけ正しいかを確認することにより，本システムの精度を評価した．具体的には，原典文書として Wikipedia 50 万記事，複製検出対象としてブログ 120 万記事を用い，本システムにより複製検出を行い，複製と判定された 378 記事に対して Wikipedia からの複製が含まれていないかを目視にて確認した．

評価を行った結果，378 記事中 374 記事が Wikipedia からの複製という結果を得た．これにより精度は 98%となった．不正解の内訳は以下の通りである．

（1） Wikipedia 自体が複製
（2） 文章が Wikipedia と偶然一致

誤り種別（1）に関しては，本システムが原典文書からの複製がないかを検出するシステムである以上，原典文書自体に他の文書からの複製が含まれていると，どうしても発生してしまう問題である．ただ，ある文書が複製元か複製先か判定するのは人間でさえ難しい問題であるため，原典文書から，他の文書からの複製を含む文書を完全に除去することは難しい．

誤り種別（2）に関しては，定型文，固有名詞等の羅列が偶然一致してしまったことが原因である．今回の実験では足切り条件を 5 文字未満にしていたが，足切り条件を大きめに取ることによって，偶然一致する可能性を低下させることは可能である．しかし，検出すべきものまで検出できなくなる可能性があるため，文字列の情報量を考慮する等，検討の余地がある．

### 6.3. 再現率

あらかじめ目視により作成した正解データをどれだけ再現することができるか評価を行った．具体的には，ブログ 10204 記事に対して，目視により複製が含まれているかどうかを判定し，複製元記事（82 記事）を収集し，これを原典文書とした上で，上記ブログを複製検出対象として本システムに入力し，何記事検出できるか評価した．

評価を行った結果，66 記事に複製が含まれているという結果を得た．それにより，再現率は 80%であった．

不正解の内訳は以下の通りである．

（A） 複製範囲が短い
（B） セグメンテーション誤り

誤り種別（A）に関しては，今回の実験条件では複製判定条件が連続 3 セグメント以上一致であったため，複製の長さがこれより短い場合には，複製と判定できなかった．複製判定基準を短くすれば，検出漏れを防ぐことも可能であるが，「本年もよろしくお願いします.」のような定型句が偶然一致した場合に，複製と誤検知してしまう恐れがある．そのため，判定条件を一律に短くするのではなく，セグメントの出現しにくさ（情報量）を加味するなどして，改善していく必要がある．

誤り種別（B）に関しては，改行文字をセグメンテーションのセパレータに含めていなかったため，箇条書きのような，複数の項目が改行文字によって分割されている場合，それぞれの項目を独立したセグメントとして認識することができなかったことが原因である．改行文字をセパレータに含めれば，このような誤りは発生しなくなるが，そうすると，今度は改行位置を変更して引用しているような事例を検出できなくなってしまう．従って，文体や係り受け解析結果等に応じて，分割位置を柔軟に変更できるようにする等の検討をする余地がある．

## 7. まとめ

本研究では，大規模かつ高速な複製検出を行うための手法を考案し，プロトタイプを実装した．実験結果から，大規模かつ高速な複製検出が可能であることを確認した．また，精度と再現率の実験を行い，効果を確認した．今後は，セグメンテーション方法や正規化処理の検討を行い，誤検出や検出漏れを改善していく予定である．

## 参　考　文　献


[1] M. Charikar: "Similarity Estimation Techniques from Rounding Algorithms", In Proc. of Symp. On Theory of Computing, pp380-388.ACM, May 2002.

[2] A. Broder, S. Glassman, M. Manasse and G. Zweig: "Syntactic Clustering of the Web", In Proc. of Int. World Wide Web Conference, pp.393-404, Apr 1997.

[3] R. J. Bayardo, Y. Ma, and R. Srikant: "Scaling up all pairs similarity search". In Proc. of the 16th Int'l Conf. World Wide Web Conference, pp.131-140, 2007.

[4] 田代崇，上田高徳，堀奏佑，平手勇宇，山名早人，"Web ページを対象とした著作権違反自動検出システム"，信学技報 DE2006-54, 2006.

[5] U. Manber and G. Myers: "Suffix arrays: a new method for on-line string searches". In 1st ACM-SIAM Symp. on Discrete Algorithms, pp.319-327, 1990.

[6] Michael O. Rabin: "Fingerprinting by random polynomials". Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.