2D-1

BLOG からの自動情報抽出システムの開発

関口　友樹†　　石川　祥‡　　木村　昌臣†
芝浦工業大学工学部情報工学科†　芝浦工業大学大学院電気電子情報工学専攻‡

1．はじめに

近年 BLOG とよばれる Web サイトの数が急増している．BLOG には商品や出来事に対する個人の感想や意見などが書かれているので，それらの感想や意見を解析することで，BLOG の作者の商品や出来事に対する評判情報等を抽出することが出来る．ただし，感想や意見を抽出するためには本文を抽出しなければならない．これは，感想や意見は BLOG の Web ページ内の広告などではなく，主に本文に書かれているからである．そこで本研究では BLOG から評判情報等を抽出するために，BLOG の Web ページから本文を自動的に抽出するシステムを作成することを目的とする．

2．提案手法

BLOG の複数の記事を比較した際，形式や長さが大体決まっているその他のテキストと比べて，本文のテキストの長さには大きな差がある．また，同じ作者が書いた BLOG 記事同士ならば，本文の HTML 内での位置は等しいと考えられる．よって，BLOG の Web ページ内の本文は，同じ作者が書いた他の記事と比較してテキストの長さの分散が最大である部分と考えられる．この考え方にもとづき，ある記事と同じ BLOG の RSS から同じ作者の別の記事を取得する．RSS とは BLOG などの Web サイトの更新情報の見出しや要約を XML 形式で記述したもので，ここから同じ作者の別の記事を取得できる．そしてそれらの HTML を XML に変換する．さらにその結果得られた XML(XHTML)を DOM ツリーに変換し，XPath が等しい部分同士のテキストの長さを比較して，その分散がもっとも大きくなる位置を本文とみなし，抽出を行う．

ここで DOM とは，XML 文書の要素や内容を追加,修正，削除などの操作をする API のことで，DOM ツリーとは図 1 のような，DOM に読み込む際に作成されるタグの入れ子構造を木構造で表したもののことである．XPath とは XML で特定の要素を指すためにタグの入れ子構造をパス形式で記述したものである．例えば，図 1 の「本文」のノードの XPath は HTML, BODY, DIV というノードに囲まれているので，/HTML/BODY/DIV となる．

3．システムの実装

3．1　HTML から XHTML への変感

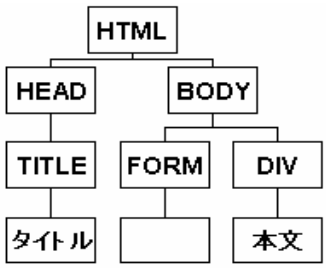
まず URL から BLOG の Web ページを読み込む．その際，BLOG の Web ページは BLOG サービス毎にそれぞれ文字コードが異なるため，一度文字コードを指定せずに BLOG の Web ページを読み込み，そのソース内の文字コードが記述されている箇所から文字コードを特定した後,文字コードを指定して再度読み込むようにする．

次に，BLOG の Web ページ内の特定の HTML タグを取り除く処理を行う．これは，HTML をそのまま XHTML に変換して DOM ツリーを構築すると，本文中のアンカーや改行などのタグによって本文のテキストが複数のノードに分割されてしまうためである．よって，それら本文を分断する恐れのあるタグは，DOM に読み込む前に正規表現を使って取り除く．またそれ以外にも，BLOG 記事の本文が 2 つのタグに分割されてしまっている場合があるので，そのような場合は 2 つに分割されている本文を，同じタグの中に入れる処理を行う．

その後，BLOG の Web ページを HTML 形式から XHTML 形式へと変換する.本研究で使用している HTML 文書を XHTML 文書へ変換するツールは，文字コードがシフト JIS 型のデータを扱うことを前提としているので，文字化けを防ぐためにツールを使用してデータをシフト JIS 型に変換してから XHTML 形式への変換を行う．

3．2　DOM ツリーの構築

変換した XHTML から DOM ツリーを構築し，全ノードを走査して，全てのテキストの内容とそのテキストの長さ，XPath を求め配列に保存する．図 1 は構築した DOM ツリーの例で，BLOG の Web ページのタグの入れ子構造を木構造で表したものである．また DOM ツリーの全ノードを走査する際に，RSS の URL も同時に抽出する．



(「本文」の XPath=/HTML/BODY/DIV)

図 1　構築する DOM ツリーと XPath の例

3．3　本文箇所の特定と抽出

3.2 節で抽出した URL から RSS を取得し，そこから同じ作者が書いた別の記事を 1～4 件取得し，その記事に対しても 3.1 節，3.2 節の同様の操作を行う．ただし，そこで得られた記事に対しては文字コードの特定や RSS の URL の取得は同じ BLOG サービスならば同じものが得られるだけであるので省略する．また，BLOG によって RSS から得られる記事の数が異なるため，そこから得る記事を最大 4 件とした．そして,図 2 のように，3.2 節で得られた XPath とテキストの長さを使って,複数の記事を比較し，同じ XPath を持つテキストの長さの分散を求め，最も分散が大きいところを本文と判定する．そして，その XPath に含まれるテキストを取得する．

Development of an automatic information extraction system from BLOG
† Tomoki Sekiguchi , Masaomi Kimura
† Shibaura Institute of Technology
‡ Sho Ishikawa
‡ graduate school of Shibaura Institute Technology

最後に,本システムでは文字コードの変換を行うツールに nkf を,HTML を XML に変換するツールに TagSoup を使用しており,すべてのプログラムは JDK5.0(Java)を使って作成した.

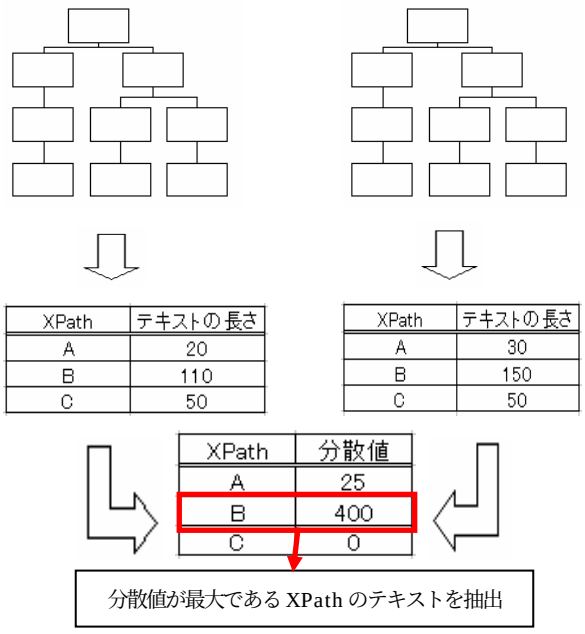


図 2. 本文の特定方法

4. 評価実験

本システムの評価として,5 万件のテストデータの処理が終わるまでの実行時間と,5 万件の実行結果中の 1000 件に対して本文抽出が正しくできているか調べ,本文抽出処理の精度をだした.

5. 結果

最初に,実行時間は約 67 時間で,1 件あたりの実行時間は約 4.8 秒になった.1 件の一般的な実行時間は約 3〜4 秒だが,特定の夜の時間帯にネットワークの状態により URL の読み込みが遅くなる場合や,BLOG の Web ページの容量が多い URL などでは,処理に数十秒かかる場合があったため,1 件あたりの実行時間がこの程度の時間になったと思われる.

次に本文抽出に対するテストの結果は表1,表2のようになった.表 1 の URL が読み込めなかったものとは,URL を読み込む際 HTTP404 エラーなどで,Web ページが存在しないと返されたものである.そして表 1 の URL が読み込めたもの 935 件の中での,本文抽出の成否の内訳は表 2 のようになった.そして本文抽出に失敗したものの内訳は表 3 のようになっている.ただし,表 1 の「URL が読み込めなかったもの」,表3の,「BLOG の削除による失敗」「BLOG 記事の削除による失敗」「本文なし」「記事毎のページではないための失敗」「BLOG ではないための失敗」の 5 項目,計 83 件については,データ側に不都合があり本システムでは対応することができない.そのため,それらのデータ側に不都合があるデータに関しては,本文抽出の精度にいれないこととする.

表3の「タイトル,コメント,トラックバックの誤抽出」で,タイトルの誤抽出については,比較した記事の同士の本文の XPath が一致しなかったため,コメント,トラックバックの誤抽出は,本文箇所よりもテキストの長さの分散が大きくなってしまったのが原因だった.「上記以外の誤抽出」とはそれら以外のサイドバーの情報などを誤抽出してしまったもので,原因はタイトルの誤抽出と同じだった.また,「本文とその他のテキストの混在」というのは,不要な HTML タグを取り除く処理のミスで,コメント,タイトル,本文,その他のテキストなどが,全部混ざって抽出されてしまったものである.次の「本文の分断」は,本文の省略や,本文の途中の引用部分に HTML タグの Table タグを使っている場合,その部分で本文が分断されてしまうことが原因となっていた.

今回のテストでは全体の 78%程の本文が抽出できていたが,データ側に不都合がある 83 件を全体の母数から取り除くと,本システムの本文抽出精度は約 85%となるといえる.

表 1　実行結果 1

全1000件中	1000件
URLが読み込めたもの	935件
URLが読み込めなかったもの	65件

表 2　実行結果 2

URLが読み込めたもの935件中	935件
本文抽出に成功したもの	781件
本文抽出に失敗したもの	154件

表 3　本文抽出に失敗したもの内訳

本文抽出に失敗したもの内訳	154件
タイトルの誤抽出	31件
コメントの誤抽出	15件
トラックバックの誤抽出	7件
上記以外の誤抽出	27件
本文とその他のテキストの混在	18件
本文の分断	38件
BLOGの削除による失敗	9件
BLOG記事の削除による失敗	3件
本文なし	1件
記事毎のページではないための失敗	2件
BLOGではないための失敗	3件

6. おわりに

本研究では BLOG から本文を抽出するために,BLOG の Web ページを XML に変換し,DOM ツリーを構築して XPath を求め,同様の処理を行った RSS から得た記事と,テキストの長さの分散を比較することで,本文箇所を特定し,抽出するシステムを構築した.本文抽出の精度は 85%,実行時間は 1 件当たり 4.8 秒程だったが,本文の抽出の精度に関しては,取り除く HTML タグの見直しなどの改良で,本文抽出の精度を上げることが出来ると思われる.また,実行時間に関しては,不要な HTML タグを取り除くとき,正規表現を用いて不要な HTML タグと一致する部分があるかどうかを調べているが,その処理に多くの時間がかかっているので,方法の見直しを検討する予定である.また,複数のマシンで本システムを並列的に稼働することによる実行時間短縮も期待できる.

参考文献

[1] 南野,鈴木,藤木,奥村:
blog の自動収集と監視,人工知能学会論文誌,
Vol.19,No.6,pp.511-520(2004)
[2] 谷口,松尾,石塚:
Blog コミュニティの抽出と分析,
人口知能学会研究会資料,SIG-SWO-A401-08(2004)
[3] 石川,木村:
BLOG のトラックバック構造における話題の可視化,
電子情報通信学会総合大会 , D-4-14 (2006)