Vol.2014-NL-217 No.3
2014/7/3

実文書を自然言語処理技術と適切に繋ぐ技術の重要性

原 忠義[1,a)]　トピチ ゴラン[1,b)]　宮尾 祐介[1,c)]　相澤 彰子[1,d)]

概要：自然言語処理（NLP）ツールの多くが入力として平文テキストを前提とする一方で，実文書中のテキストは多様なレイアウト，文構造，埋め込みのオブジェクトなどによって，より表現豊かに表示されている．このようなテキストを NLP ツールで解析する際には，ツールの利用者が対象テキストをツールに合った入力形式に変換しなければならない．また，利用者の不慣れな変換作業によって得られた入力を用いたところで，そのツールが本来持つとされる性能を発揮することは困難となるであろう．本研究の目的は，平文テキストでは表し切れないテキスト構成がタグを用いて表現されるような XML 文書の解析を題材として，この問題への意識喚起を促すことにある．我々は， XML でタグ付けされたテキストと，NLP ツールの入出力となる平文テキストとの間の一般的な変換枠組を提案し，本枠組を用いて獲得されるテキスト列が，単純にタグを除去して得られるテキストよりも構文解析器で高被覆かつ高効率に処理できることを示し，実文書を NLP 技術と適切に繋ぐ技術を開発することの重要性を浮き彫りにする．

キーワード：半構造化文書，半構造化テキスト，実文書，実テキスト，XML 文書，XML タグ，NLP ツール

Significance of Bridging Real-world Documents and NLP Technologies

Hara Tadayoshi[1,a)]　Topić Goran[1,b)]　Miyao Yusuke[1,c)]　Aizawa Akiko[1,d)]

Abstract:
Most conventional natural language processing (NLP) tools assume plain text as their input, whereas real-world documents display text more expressively, using a variety of layouts, sentence structures, and inline objects, among others. When NLP tools are applied to such text, users must first convert the text into the input/output formats of the tools. Moreover, this awkwardly obtained input typically does not allow the expected maximum performance of the NLP tools to be achieved. This work attempts to raise awareness of this issue using XML documents, where textual composition beyond plain text is given by tags. We propose a general framework for data conversion between XML-tagged text and plain text used as input/output for NLP tools and show that text sequences obtained by our framework can be much more thoroughly and efficiently processed by parsers than naively tag-removed text. These results highlight the significance of bridging real-world documents and NLP technologies.

Keywords: Semi-structured documents, Semi-structured text, real-world documents, real-world text, XML documents, XML tags, NLP tools

1. はじめに

近年の自然言語処理（NLP）技術の進歩により，様々な大規模テキストに対してそれらの技術を適用し，豊富な情報を獲得したり，追加の情報をテキストに付与するなどの応用が期待される．しかしながら，実際にそのような応用を考える際に，我々はある問題に直面する．従来の NLP ツールでは通常，入力テキストが単語列からなる平文テキストであることを想定しているが，実文書においては，テキストが様々な構造（章・段落・箇条書き・数式・図表・脚注など）でより豊かに表現されている．このようなテキ

[1] 国立情報学研究所
National Institute of Informatics, Japan
[a)] harasan@nii.ac.jp
[b)] goran_topic@nii.ac.jp
[c)] aizawa@nii.ac.jp
[d)] yusuke@nii.ac.jp

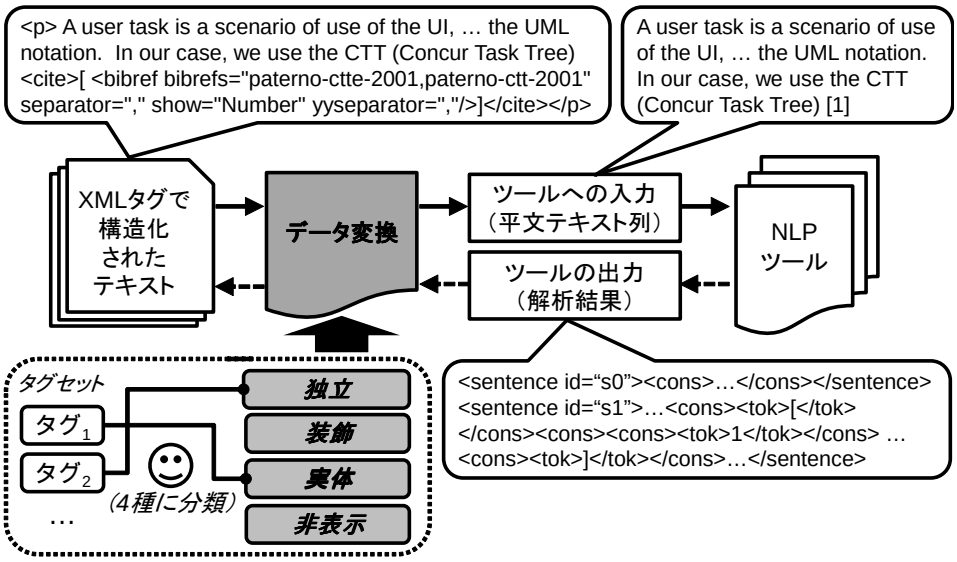


図 1 NLP ツールを XML タグで半構造化されたテキストに適用するための提案枠組

ストから NLP ツールに与えるための正しい入力を得る手間は完全に利用者に委ねられており，利用者は解析対象の文書をツールの入力形式へと変換し，（必要に応じて）解析結果を元の文書に統合するための前・後処理を自身で準備する必要がある．このことはツールの有用性を下げるだけでなく，適切な入力がツールに与えられず，本来ツールを用いることで得られるべき NLP 技術の恩恵を十分に引き出せない恐れがある．

本研究の目的は，この変換と統合のプロセスを簡易化する枠組の提案を通じてこの問題への意識喚起を促すことにある．我々は平文テキストでは表し切れない多様なテキスト構成が XML 文書におけるタグによって捉えられると仮定し，XML タグ付きのテキストと NLP ツール入出力形式間のデータ変換に焦点を絞る．観察により，このデータ変換の処理は対象テキストで用いられる各 XML タグのテキスト構成機能によって決定され，その機能が主に４種のみであるという直観が得られたため，我々はこの４種それぞれに対する変換戦略を考案した．XML タグセット内の全てのタグをこの４種に分類すれば，データの変換・統合は対象テキストのサイズによらず変換戦略により自動的に実行可能となる（図 1）．

実験では，数種の XML 文書群に対して提案枠組を適用し，文書中の使用タグ種のごく一部（20%以下）を分類すれば XML 文書から平文テキスト列を抽出可能であることを示す．また，得られたテキスト列を入力に用いることで，単純にタグを除去したテキストを用いるよりも，２種の典型的な構文解析器で文書全体を高被覆率かつ短時間で解析できることを示す．

本研究の貢献は，実文書を NLP 技術と適切に繋ぐ技術を開発することの重要性を，実際の取り組みを通じて示すことにある．適切な枠組で支援すれば，従来の NLP ツールが，平文テキストに対して元来期待される処理性能で実文書を解析する能力を既に有することを示し，NLP が必要とされる多様な局面においても NLP 技術の恩恵を最大限に享受できるよう，今後の議論促進に繋げていきたい．

2 節では関連研究について述べ，3 節では XML タグが持つ 4 種類のテキスト構成機能を説明し，それら各々に対するデータ変換戦略を記述する．4 節では，提案枠組による変換の効率性と，得られたテキスト列の NLP ツール入力としての適切性を，複数文書において検証する．

2. 関連研究

我々の知る限り，対象テキストと NLP ツールの入出力間でのデータ変換に関する統一的手法に取り組んだ研究は存在しない．NLP ツールによっては，実文書からツールに適した入力テキストを抽出するためのスクリプトが同梱されている場合もあるが，そのスクリプト自体も基本的には何らかの特定文書フォーマットを想定している．例としては，深い統語構造を解析する構文解析器である C&C パーザ [1] や Enju [2] はそれらの入力として品詞タグ付けされた文を想定しているため，配布パッケージ[*1][*2]に品詞タガーが同梱されているが，その品詞タガーも入力として１文毎に区切られた平文を想定している．

本研究に最も関連すると思われる取り組みとしては，UIMA [3] が様々な解析を統合枠組上で取り扱うことを提案しているが，UIMA 自体は枠組のみで，対象テキストをいかにしてその解析プロセスに入力するかに関しては示していない．これに関しては，UIMA 枠組に基づいたプロジェクトである RASP4UIMA [4]，U-compare [5]，Kachako [6]

[*1] [C&C パーザ]: http://svn.ask.it.usyd.edu.au/trac/candc/wiki
[*2] [Enju]: http://kmcs.nii.ac.jp/enju/

タグ分類	分類基準	変換戦略
独立	周囲のテキストから統語的に独立した領域を表す	タグとタグで囲まれた領域 (A) をテキスト (B) から分離 → (A)，(B) にツールを適用後，(A) を (B) に復帰
装飾	周囲のテキストと同等の領域として領域内の表示スタイルのみを変更	タグ (A) のみをテキスト (B) から除去 → ツール適用後に (A) を (B) に復帰
実体	周囲のテキスト構成要素と同等の構成要素として扱われる最小のオブジェクト単位を表す	タグとタグで囲まれた領域 (A) を代替語（文字列）(C) で置換 → ツール適用後に (C) を (A) に復元
非表示	表示方法の設定や追加情報を記述する	タグとタグで囲まれた領域 (A) をテキスト (B) から除去 → ツール適用後に (A) を (B) に復帰

表 1　4 種のタグ分類と各分類に対する変換戦略

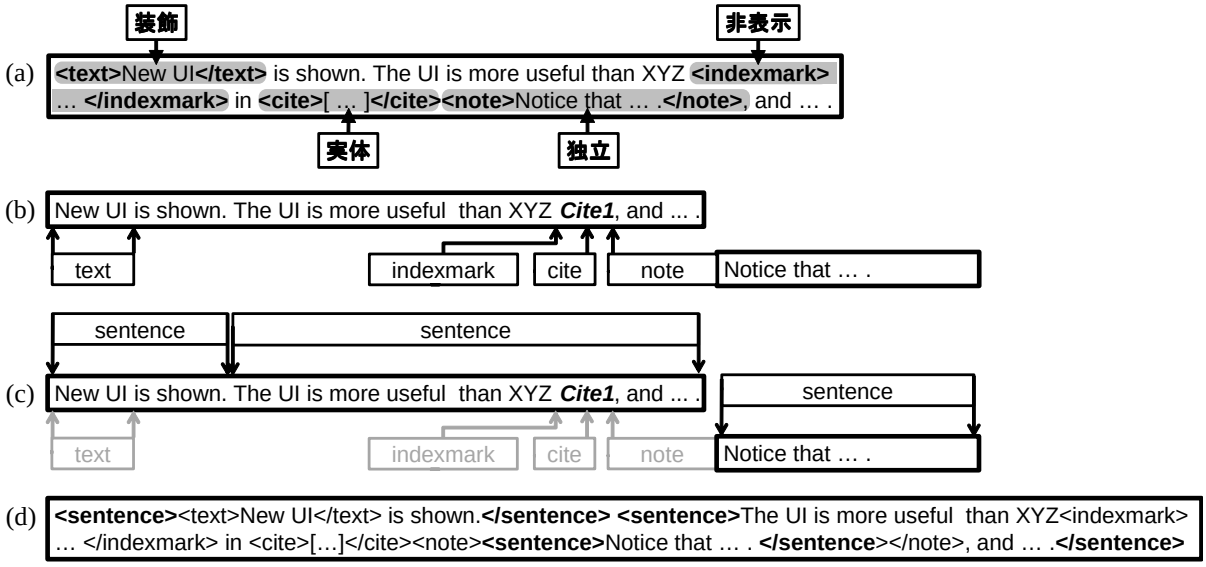


図 2　変換戦略の実施例

などが，文書を様々なツールで解析可能にするシステムを構築しているが，利用者は既にそのシステムに統合されたテキストやツールの組み合わせしか扱うことができない．他に，GATE [7] は UIMA に近いコンセプトに基づくシステムであり，基本的には XML 文書を入力として受け付けるが，利用するためには，ツールをこのシステムに統合する必要がある．

本研究における提案枠組は，対象が XML 文書であることを想定してはいるものの，利用者は文書内で用いられる XML タグを４種に分類することにより，NLP ツールを，それ自体に手を加えることなく対象文書に適用することが可能となる．

3. 提案枠組

本節では，タグが持つ４種のテキスト構成機能に基づき，タグ付きテキストと NLP ツールの入出力間のデータ変換を行うための枠組を記述する．最初に４種の機能およびそれらに基づく変換戦略を導入し，次にその戦略を用いたデータ変換処理の全体を管理する手続きを導入する．

3.1 ４種類のタグに対する変換戦略

タグの機能は**独立**，**装飾**，**実体**，**非表示**の４種に分類され，各分類のタグに対するデータ変換戦略が表 1 のように記述できる．本節では各分類とその変換戦略について，文区切り器を図 2(a) に示すテキストに適用する簡易例を用いて解説する．解析対象のテキストには “$<note>$”，“$<text>$”，“$<cite>$”，“$<indexmark>$” の４つのタグが含まれており，それぞれ**独立**，**装飾**，**実体**，**非表示**タグに分類されるものである．以下，各分類毎に説明する．

独立タグに囲まれた領域は，何がしかのタイトルや１つの節など，統語的に独立したテキストを表す．そのような領域は，図 2(a) の例において脚注を導入する “$<note>$” タグのように他の文の途中に挿入されることがある．このようなタグに対しては，タグで囲まれた領域をテキストから分離し，分離された領域とテキストそれぞれに NLP ツールを適用すれば良いと考えられる．

一方，**装飾**タグで囲まれた領域は必ずしも統語的な独立が保証されず，テキストのフォントや色を変更したり（例における “$<text>$”）節を段落分けする[*3]など，主に対象領

[*3] 論文などの文書においては，１つの文が２つの段落領域に分割さ

域を視覚的に強調することなどに用いられる．これらのタグを含むテキストに対しては，NLP ツールに入力する前にタグのみを除去し，ツール適用後にタグを元の位置に復帰すれば良いと考えられる[*4]．

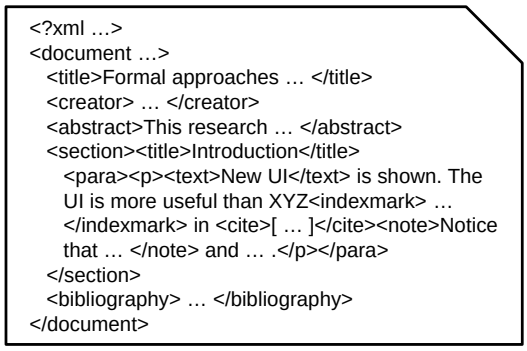
実体タグで囲まれた領域には，テキストを構成する一つの統語的要素として扱われるべきオブジェクトを表すための特殊記述が含まれており，この領域は基本的には自然言語テキストの形で構成されていないため，NLP ツールでは解析できない[*5]．これらのタグが含まれるテキストに対しては，タグで囲まれた領域を何らかの適切な代替語（文字列）に置換した上で NLP ツールに適用し，後ほど代替語の箇所を元の領域に復元すれば良いと考えられる．

非表示タグで囲まれた領域は，実際には表示されず，他の目的，例えば目次の作成（例における “<*indexmark*>”）などに用いられるため，NLP ツールの対象からは外される[*6]．これらのタグが含まれる領域に対しては，タグで囲まれた領域を除去したテキストに NLP ツールを適用し，後に領域を元の位置に復帰すれば良いと考えられる．

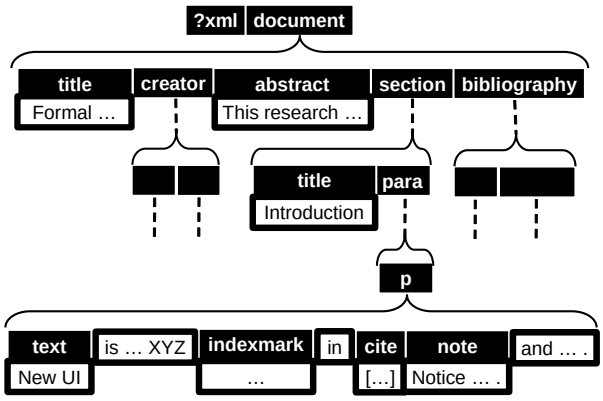
各タグ分類に対する上記の戦略に従い，図 2(a) のテキストは以下のようにして変換される．最初に，各タグ（およびタグで囲まれた領域）をテキストから除去する．その際に，タグが除去された図 2(b) のテキストにおける各タグの復帰位置をオフセットで記憶しておく．図 2(b) において “Cite1” はタグ “<*cite*>” で囲まれた領域の代替語として用いられた文字列である．“<*note*>” タグに関しては，タグで囲まれた領域を**独立**タグ領域として分離した後に，再帰的に各領域内の変換を行う．その結果，２つの平文テキスト領域 “New UI ...” と “Notice that ...” が得られ，文区切り器に入力されることとなる．解析された文区切り情報が図 2(c) のように与えられると，記憶しておいたタグ復帰位置の情報を用いて，文区切り情報と元のタグ情報を統合し，最終的に図 2(d) のような XML タグ付きテキストを生成する．

3.2 効率的なタグ分類とデータ変換のための手続き

実際の XML 文書においては，図 3(a) に示されるように多種多様なタグが導入され，タグ領域は図 3(b) で表されるような多層構造をとる（図 3(b) における黒塗りと白塗りの



(a) XML 文書



(b) XML 文書の構造

図 3 XML 文書の例

四角はそれぞれ XML タグと平文テキスト列を表し，タグで囲まれた領域は出現順にタグの下に並べられている．）．これらのタグを先述の４種に効率良く分類し，かつ同時に XML 文書から平文テキスト列を獲得するための変換手続きを，図 4 の疑似アルゴリズムで示すように実装した．本節では，このアルゴリズムの動作を説明する．

本手続きでは基本的に，４種のタグ分類に対する変換戦略を最上層のタグからより低層のタグへと再帰的に適用する．手続き中の @*independent*，@*decoration*，@*object*，@*meta_info* はそれぞれ，既に利用者が**独立**，**装飾**，**実体**，**非表示**タグに分類を行ったタグ（名）を保存するための配列を表す．これら４つの配列と前節の戦略を利用し，対象の文書（群）から平文テキスト列への変換が試みられる．その際に，未知の（ゆえに変換できない）タグがあれば配列 @*unknown* に格納される．文書全体の処理終了後，利用者は報告された未知のタグを新たに分類し，この分類に基づき再び変換が試みられる．本手続きではこの処理を，未知のタグが現れなくなるまで行うこととなる．

例えば，図 3(a) の文書に対する初回試行として，利用者が上記４配列を空にした状態で変換アルゴリズムを適用すると，関数 “*data_convert*” において，文書内で最上層のタグである “<*?xml*>” と “<*document*>” が未分類タグとして検出され，配列 @*unknown* に加えられるとともに，文書中でこれらのタグとタグで囲まれた領域は “UN1”，

れることがある．段落を表すタグが**独立**あるいは**装飾**タグのどちらに分類されるかは，対象文書での段落の扱い方による．

[*4] **装飾**タグの存在が，タグで囲まれた領域がチャンクを構成するということを暗示することも考えられ，その情報も NLP ツールで活用できるかもしれない．

[*5] 箇条書きや表のように，領域内の一部記述に自然言語テキストが用いられる場合があり，これらを周囲のテキストとどのように関連づけて扱うべきかに関しては今後議論が必要なところであるが，今回の枠組では考慮していない．箇条書きの扱いに関しては [8] や [9] の議論が１つの指針となるであろう．

[*6] 領域内に解析可能なテキストが含まれる場合に，その領域に NLP ツールを適用するか（タグを**独立**タグに分類するか）否かの判断は利用者に委ねられるところである．

```
@plain_text_sequences = ();  # plain text sequences input to NLP tools
@recovery_info = ();         # information for recovering original document after applying NLP tools
@unknown = ();               # unknown tags

function data_convert ($target_sequence, $seq_ID) {

  if ($target_sequence contains any tags) {  # process one instance of tag usage in a target sequence
    $usage = (pick one instance of top-level tag usage in $target_sequence);
    $tag = (name of the top-level tag in $usage);
    @attributes = (attributes and their values for the top-level tag in $usage);
    $region = (region in $target_sequence enclosed by tag $tag in $usage);
    $tag_and_region =  (region in $target_sequence consisting of $region & tag $tag enclosing it);

    if ($tag ∈ @independent)        { remove $tag_and_region from $target_sequence;
                                      add [“independent”, $tag, @attributes,  $seq_ID, $seq_ID + 1,
                                            (offset in $target_sequence where $tag_and_region should be inserted) ] to @recovery_info;
                                      data_convert($region,  $seq_ID + 1);  } # process the tagged region separately
    else if ($tag ∈ @decoration)    { remove only tag $tag enclosing $region from $target_sequence;
                                      add [“decoration”, $tag, @attributes,
                                            (offsets in $target_sequence where $region begans and ends)] to @recovery_info; }
    else if ($tag ∈ @object)        { replace $tag_and_region in $target_sequence with a unique plain word $uniq;
                                      add [“object”, $uniq, $tag_and_region] to @recovery_info; }
    else if ($tag ∈ @meta_info)     { remove $tag_and_region from $target_sequence;
                                      add [“meta_info”, $tag_and_region,
                                            (offset in $target_sequence where $tag_and_region should be inserted)] to @recovery_info; }
    else                            { replace $tag_and_region in $target_sequence with a unique plain word $uniq;
                                      add [“unknown”, $uniq, $tag_and_region] to @recovery_info;
                                      if ($tag ∉ @unknown) { add $tag to @unknown; } }

    data_convert($target_sequence, $seq_ID);  # process the remaining tags
  }
  else {  # a plain text sequence is obtained
     add [$seq_id,  $target_sequence] to @plain_text_sequences;
  }
}

function main ($XML_document) {
  data_convert ($XML_document, 0);
  return @plain_text_sequences, @recovery_info, @unknown;
}
```

図 4　XML テキストから平文テキスト列への変換を行うための疑似アルゴリズム

“UN2” といった固有の代替語（文字列）で置換される．これにより，入力された文書は “UN1 UN2” のように平文テキストのみで構成されるシーケンスに変換され，配列 $@plain\_text\_sequences$ に格納され，変換アルゴリズムは終了となる．利用者は配列 $@unknown$ において報告される未知のタグを４分類のいずれかに分類し，２回目の試行へと移る[*7]．

独立または**装飾**タグの場合，変換アルゴリズムは文書中のタグで囲まれた領域を分離するか，あるいはタグのみ除去した後，得られたテキスト（列）に対して再帰的に処理を行う．この分離・除去の操作においては，配列 $@recovery\_info$ に，得られるテキスト列においてタグが本来挿入されているべき場所をオフセットで記憶しておくことで，NLP ツールを適用後に元のタグを復帰させテキスト構造を復元できるようにする．**実体**，**非表示**タグの場合，タグで囲まれた領域は固有の文字列に置換されるか対象テキストから除去される（この場合も $@recovery\_info$ に復帰のための情報が格納される）が，このことは，タグで囲まれた領域の内部はそれ以上展開・処理されない，ということを意味する．これにより，タグ領域内部でのみ使用されているタグに対する不必要な分類作業を回避し，利用者の負担を最小限に抑えることができる．

変換アルゴリズムが未知のタグを報告しなくなった時点で，利用者は，NLP ツールへの入力として用いる平文テキスト列を取得するのに十分なタグ分類を完了すると同時に，その平文テキスト列は既に配列 $@plain\_text\_sequences$ に格納されていることとなる．得られたテキスト列に NLP ツールを適用した解析結果は，配列 $@recovery\_info$ に基づきオフセット情報をマージすることによって元の XML 文書のタグ情報と統合され[*8]，最終的に元の文書の構造が復元される．

4. 実験

本節では，3.2 節で導入したアルゴリズムが複数の文書

[*7] 利用者は報告された未知のタグ全てを必ずしも一度に分類する必要はなく，判断を後の試行に遅らせることも可能である．

[*8] タグ領域の交差が生じる場合，今回の実装では，ツールによる解析結果の領域をより小さい領域に分割することで回避した．

対象文書	文書数（文）	全タグ数（異なり数）	処理対象として分類されたタグの数 （異なり数）					得られたテキスト列の数
			独立	装飾	実体	非表示	合計	
PMC	1,000	1,357,229(421)	32,109(12)	62,414(8)	48,205(9)	33,953(56)	176,681(85)	25,679
ArX.	300	1,969,359(210*)	5,888(15)	46,962(12)	60,194(8)	7,960(17)	121,004(52)	4,167
Wiki.	300	223,514(60*)	3,530(12)	11,197(8)	1,470(28)	11,360(67)	27,557(115)	2,286
JNL-E	68	142,410(57*)	8,312(25)	12,152(16)	5,953(9)	2,328(19)	28,745(69)	6,241
JNL-J	384	699,107(58*)	50,488(23)	56,219(18)	32,268(10)	13,599(21)	152,574(72)	38,347

（**ArX.**: arXiv，**Wiki.**: Wikipedia，**JNL-E/J**: 論文誌「自然言語処理」（英語／日本語））

表 2 各種文書に対して分類されたタグと得られた平文テキスト列

対象文書	分類タグ	構文解析器 Enju による解析				構文解析器 Stanford パーザによる解析			
		検出文数（文）	時間（秒）	コスト（秒/文）	失敗文数（文（割合））	検出文数（文）	時間（秒）	コスト（秒/文）	失敗文数（文（割合））
PMC	**なし**	159,327	209,783	1.32	4,721 (2.96%)	170,999	58,865	0.39	18,621 (10.89%)
	独/非	112,285	135,752	1.21	810 (0.72%)	126,176	50,741	0.44	11,881 (9.42%)
	全て	126,215	132,250	1.05	699 (0.55%)	139,805	63,295	0.49	11,338 (8.11%)
ArX.	**なし**	74,762	108,831	1.46	2,047 (2.74%)	75,672	27,970	0.43	10,590 (13.99%)
	独/非	41,265	89,200	2.16	411 (1.00%)	48,666	24,630	0.57	5,457 (11.21%)
	全て	43,208	87,952	2.04	348 (0.81%)	50,504	26,360	0.58	5,345 (10.58%)
Wiki.	**なし**	10,561	14,704	1.39	1,161 (10.99%)	14,883	3,114	0.24	1,651 (11.09%)
	独/非	5,026	6,743	1.34	67 (1.33%)	6,173	2,248	0.38	282 (4.57%)
	全て	6,893	6,058	0.88	61 (0.88%)	8,049	2,451	0.31	258 (3.21%)
JNL-E	**なし**	23,196	24,881	1.07	271 (1.17%)	24,942	9,069	0.39	1,577 (6.32%)
	独/非	15,606	21,304	1.37	183 (1.17%)	17,572	7,865	0.48	1,058 (6.02%)
	全て	17,929	18,683	1.04	50 (0.28%)	19,925	10,154	0.53	892 (4.48%)

対象文書	分類タグ	形態素解析器 JUMAN による解析				形態素解析器 MeCab による解析			
		検出文数（文）	時間（秒）	コスト（秒/文）	失敗文数（文（割合））	検出文数（文）	時間（秒）	コスト（秒/文）	失敗文数（文（割合））
JNL-J	**なし**	96,668	122	0.0013	10 (0.01%)	97,312	7	0.000072	174 (0.18%)
	独/非	76,277	86	0.0011	8 (0.01%)	78,461	6	0.000076	119 (0.15%)
	全て	114,250	59	0.0005	2 (0.00%)	116,424	6	0.000052	0 (0.00%)

（**ArX.**: arXiv，**Wiki.**: Wikipedia，**JNL-E/J**: 論文誌「自然言語処理」（英語／日本語），**独**+**非**: **独立タグ**+**非表示タグ**）

表 3 タグ分類に基づき得られた平文テキスト列が構文解析・形態素解析性能にもたらす影響

群に頑健に適用可能であり，かつ得られた平文テキスト列が NLP ツールの入力として適切なものとなっていることを，実験を通して検証する．また，この検証で得られた知見を踏まえ，実文書と NLP 技術を適切に繋げる技術の重要性について議論する．

4.1 対象文書

PubMed Central (PMC)[*9]および arXiv.org[*10]においてそれぞれ公開されている英語の科学論文，ANLP-20 コーパス[*11] において公開されている言語処理学会論文誌「自然言語処理」の日本語および英語論文，Wikipedia[*12]の各エントリのウェブページ，以上５種の文書群を取得し，我々のアルゴリズムを適用した．

提案枠組が対象としている XML 形式の文書がそのまま利用できるのは PMC の論文のみであったが，それ以外の文書群に関しても，arXiv は XHTML（HTML を XML 形式で再定義したもの）形式，「自然言語処理」論文誌は，LaTeX 形式の文書から LaTeXML[*13]を用いた自動変換により得られた XHTML 形式，がそれぞれ公開されていたため、これを利用した．Wikipedia のページは HTML 形式ではあるものの，生成元の中間ファイルが XML 形式であることから，XML 形式に準ずる文書として扱うこととした．

各文書群の文書数に関しては，PMC からランダムに選んだ 1,000 文書，arXiv からランダムに選んだ 300 文書，「自

[*9] http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/
[*10] http://arxiv.org/
[*11] http://nlp20.nii.ac.jp/resources/
[*12] http://www.wikipedia.org/
[*13] http://dlmf.nist.gov/LaTeXML/

然言語処理」論文誌の公開データ中から XHTML 化に成功している 452 文書を日本語論文 384 文書・英語論文 68 文書に分割したもの，Wikipedia からランダムに選んだ 300 記事をそれぞれ利用した．

各文書は様々なテキスト部分から構成されているが，今回の実験では，科学論文に関しては論文および節のタイトル・アブストラクト・主要な節の本文に，Wikipedia のページに関しては記事タイトル・本文の見出し・本文に NLP ツールを適用することを念頭に置いて，タグ分類を行った．

4.2 タグ分類の効率性検証

各 XML 文書群に関して，分類されたタグと得られたテキスト列を表 2 にまとめた．表の第 2 列から第 9 列までにはそれぞれ，使用文書数・文書群に使用されていた全タグ数とその異なり数・実際に分類され処理されたタグの数とその異なり数・変換によって得られたテキスト列の数が示されている[*14]．得られたテキスト列には，簡易的な正規表現のマッチングによってはタグが検出されず，このことから少なくとも，タグ付けされたテキストが提案枠組により平文テキスト列へと変換されたと考えられる．

表の第 3 列と第 8 列を比較すると，PMC 論文に関しては，文書で用いられている全タグのうち，異なり数で 1／5 以下のタグを分類（タグ出現数でみると 15%のみに注目）することで平文テキスト列を得られたことがわかる．これは，**実体・非表示**タグで囲まれた領域の内部がそれ以上展開されなかったことによるものである．arXiv 論文，Wikipedia 記事，「自然言語処理」論文誌に関しても，それぞれ全タグ出現数の 20%程度またはそれ以下に注目するのみで平文テキストが得られていることから，同様の効果が見て取れる．

4.3 得られたテキスト列の NLP ツールへの入力としての適切性検証

我々は各文書群から数文書をランダムに選択し，それぞれ前節で得られたテキスト列が，NLP ツールに直接入力可能で，かつ元の記事全体を被覆可能な「適切な」文から構成されていることを確認した．その上で，この，NLP ツールへの入力としての「適切性」がより実践的な状況においていかに重要であるか評価するため，前節で得られた全平文テキスト列を構文解析器（日本語文書は形態素解析器）に入力し，解析を試みた．用いた構文解析器は深い統語・意味構造を解析するために用いられる Enju パーザと，文構成要素・依存関係を解析するために用いられる Stanford パーザ [10][*15]の 2 種である[*16][*17]．表 3 は（1）単純にタグのみを除去，（2）**独立**および**非表示**タグのみ提案枠組の戦略で変換し，残りのタグは単純除去，（3）提案枠組によりタグを 4 種に分類し変換，の 3 通りの戦略で得られた平文テキスト列に対する構文解析の性能を比較したものである．比較した性能の尺度は，構文解析器–文書群–変換戦略の各組み合わせについて，検出された文の数[*18]・全文の解析に要した時間および 1 文あたりの解析時間[*19]・解析に失敗した文の数とそのパーセンテージである[*20]．

この表より，全ての文書群に対して，各構文解析器（特に Enju パーザ）の適用時に，提案枠組に基づく変換戦略を一部または全てのタグに適用し得られたテキスト列を用いる方が，タグを単純除去して得られたテキスト列を用いるよりもはるかに高被覆率（各構文解析器の第 4 列を参照）かつ短時間（各構文解析器の第 2 列参照）で文書全体を解析できていることがわかる．これは主に，タグを単純除去して得られるテキスト列に含まれ構文解析の混乱原因となるであろう，文の挿入（**独立**タグにより導入）・自然言語テキストではない要素で構成された表現（**実体**タグにより導入）・文の表示に直接関係しない列（**非表示**タグにより導入）を提案枠組により事前に処理できたためと考えられる．その中でも特に，**実体・非表示**タグに対処することで解析の性能が劇的に向上しているのは，非自然言語で構成される箇所を解析対象から適切に排除できたためであろう．

実体・非表示タグのみ変換する場合と比較して，4 種のタグを全て処理する，すなわち**独立**タグ領域の分離（および，残りのタグである**装飾**タグの単純除去）の処理を追加すると，検出される文の数が増加する．これは，脚注の挿

[*14] 「自然言語処理」論文誌および Wikipedia ページに関しては，XHTML，HTML のタグがテキスト構成機能としてはより抽象的な概念を示すため，タグの異なり数は非常に小さくなっている（表内*を参照）．arXiv も XHTML 形式をとっているが，XML 形式に基づく直接的な記述も多く見受けられるため，タグの種類は比較的多い．これらの文書群に関しては，タグのテキスト構成機能を十分に捉えるため，タグ名と属性・属性値の組み合わせをもって 1 つのタグと考えることにした．表に示される分類されたタグの数はこの処置に基づくものである．

[*15] http://nlp.stanford.edu/software/lex-parser.shtml

[*16] 日本語論文に関しては，以下の 2 つの形態素解析器を適用した．
JUMAN: [http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN]
MeCab: [http://mecab.sourceforge.jp/]

[*17] 実験においては，構文解析器によって得られた解析結果をオフセット形式に変換し，元の XML 文書のタグオフセットとマージした後，XML タグ付きのテキストに復元した．得られた XML 文書は，*xmllint*（XML を解析する UNIX のツール）での整合性の検証により，タグの交差等のない，正常なフォーマットで記述されていることが確認された．

[*18] 平文テキストを文単位に区切るために，Stanford パーザには文区切り器が内蔵されているためそちらを用い，Enju パーザには文区切り器 GeniaSS [http://www.nactem.ac.uk/y-matsu/geniass/] を用いた．日本語文書に関しては，全角句点を文境界とみなした．

[*19] Enju パーザに関しては，解析に失敗した文に費やした解析時間も考慮に入れられている．Stanford パーザでは，解析に失敗する可能性のある文は（解析失敗文として）スキップされるため（次の脚注を参照），解析失敗文の処理時間は考慮に入れられない．

[*20] Stanford パーザでは，オプション設定を用いて解析対象文を長さ 50 単語以下に制限した．これは，メモリ不足で解析に失敗するとプロセス全体が止まってしまうのを回避するための措置である．この問題は使用メモリ量を初期設定の 150MB から 2GB に増加しても防ぎ切れなかったため，本実験では使用メモリ量は初期設定のまま，メモリ不足を引き起こし得る極端に長い文の解析を事前に（解析失敗文として）スキップするという形で回避した．

入のように，文区切り器では検出できなかった新たな文境界を**独立**タグが確実に示し得るからであろう．このようにしてより多く得られた文は，必然的に平均文長が短くなるため，Enju パーザにおける検索空間不足，および Stanford パーザにおける（文長制限による）構文解析対象からの除外，をそれぞれ抑え，その結果，構文解析失敗数を減少させることに繋がっていると思われる．４種のタグを全て提案枠組で変換することで，Stanford パーザでの合計解析時間が増加しているが，これは上記の失敗数減少により解析対象が増加しているためである．一方で，Enju パーザでは，合計解析時間は逆に減少している．これは，短くなった文長が探索空間を劇的に狭めたことによるものであると考えられる．

4.4 実文書を NLP 技術と適切に繋ぐ技術の重要性

前節において，提案枠組を用いて得られた平文テキスト列を利用することで，対象の文書全体を高被覆率かつ効率的に構文解析可能であることが示された．このように高被覆・高効率な処理がもたらすものとは一体何であろうか．単語数を数える等の単純なアプローチで得られるような，浅い分析で良いのであれば，単純なタグ除去でも事足りるだろう．すなわち，文の挿入があったとしても，単語の数には影響しないし，非自然言語の要素が混入していたとしても，問題のないテキスト列が大量にあれば無視できるだろう．

一方で，そのような浅い分析では，より詳細かつ正確な分析要求（例えば，談話分析・翻訳・文法抽出など）には応えられない．文書中に現れるわずかな兆候をも逃さないためには，ごく小さなテキスト部分で発せられた情報も慎重に吟味せねばならず，また，その分析は，本文とは関係のない要素の混入を確実に排除したテキストに対して行う必要がある．

このようなタスクは，まさに NLP の研究分野が近年取り組み，確立してきたものそのもののはずである．すなわち，自然言語現象の正確な解析に集中するため，データセットは平文テキスト列の形で整備され，そのデータセットを用いて新たなタスクが発見され取り組まれる一方で，多くの目覚ましい成果が報告されてきた．

それでは，これらの挑戦が目指す最終的な目的は何であろうか．少なくとも，その目的は注意深く整備されたデータセットを構文解析することではないだろう．我々は NLP 分野における多くの有用な成果を，その本来の対象，すなわち実世界文書に還元することを考えたい．本研究で示したように，適切な枠組で支援してやれば，従来の NLP ツールには，既に実文書テキストを本来期待されている性能で解析できる能力が備わっている．このことを踏まえると，対象の実文書と NLP 技術を適切に繋ぐことは，NLP をあらゆる局面で利用する際に，NLP 技術によってもたらされる恩恵を余すことなく享受するための，きわめて重要なタスクであると言えよう．

5. 結論

本研究では，XML でタグ付けされたテキストと NLP ツールの入出力との間の変換枠組を提案した．提案枠組では，一度テキスト中に用いられている XML タグを４種のテキスト構成機能に分類すれば，その分類に基づき平文テキスト列への変換が自動的に行われる．実験において，我々は提案枠組を数種の文書に適用し，文書中に用いられるタグの 20%以下を分類することで，対象文書から平文テキスト列を獲得することに成功した．また，このようにして得られたテキスト列を用いることで，タグを単純に除去して得られるテキスト列よりも，対象文書をより高被覆かつ効率的に構文解析できることが観察され，このことは，実文書を NLP 技術と適切に繋ぐ技術の重要性を示唆するものであった．

本論文で提案した変換枠組はツールとして公開する準備を進めており，本論文およびツールの公開等を通し，NLP ツールを多種多様な形式の実文書に適用することに関して議論を共有し，深めていきたいと考えている．

参考文献

[1] Clark, S. and Curran, J. R.: Wide-coverage efficient statistical parsing with CCG and log-linear models, *Computational Linguistics*, Vol. 33(4), pp. 493–552 (2007).
[2] Ninomiya, T., Matsuzaki, T., Miyao, Y. and Tsujii, J.: A log-linear model with an n-gram reference distribution for accurate HPSG parsing, *Proceedings of the 10th International Conference on Parsing Technologies (IWPT'07)*, Prague, Czech Republic (2007).
[3] Ferruci, D., Lally, A., Gruhl, D., Epstein, E., Schor, M., Murdock, J. W., Frenkiel, A., Brown, E. W., Hampp, T., Doganata, Y., Welty, C., Amini, L., Kofman, G., Kozakov, L. and Mass, Y.: Towards an Interoperability Standard for Text and Multi-Modal Analytics, Technical Report RC24122, IBM Research Report (2006).
[4] Andersen, O., Nioche, J., Briscoe, E. and Carroll, J.: The BNC parsed with RASP4UIMA, *Proceedings of LREC 2008* (2008).
[5] Kano, Y., Miwa, M., Cohen, K., Hunter, L., Ananiadou, S. and Tsujii, J.: U-Compare: a modular NLP workflow construction and evaluation system, *IBM Journal of Research and Development*, Vol. 55, No. 3, pp. 11:1–11:10(2011).
[6] Kano, Y.: Kachako: a Hybrid-Cloud Unstructured Information Platform for Full Automation of Service Composition, *Scalable Deployment and Evaluation, Proceedings in the 1st International Workshop on Analytics Services on the Cloud (ASC), the 10th International Conference on Services Oriented Computing (ICSOC 2012)*, Shanghai, China (2012).
[7] Cunningham, H., Tablan, V., Roberts, A. and Bontcheva, K.: Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics, *PLoS Comput Biol*, Vol. 9(2) (2013).
[8] Aït-Mokhtar, S., Lux, V. and Bánik, É.: Linguistic

parsing of lists in structured documents, *Proceedings of EACL Workshop on NLP and XML* (2003).

[9] 松崎拓也，小林義行，藤尾正和，待井君吉，川端薫：箇条書きを含む文書に対する構文解析, 言語処理学会第 18 回年次大会論文集, pp. 971–974 (2012)

[10] de Marneffe, M.-C., MacCartney, B. and Manning, C. D.: Generating typed dependency parses from phrase structure parses, *Proceedings of LREC 2006*, pp. 449–454 (2006).

正誤表

- ６頁，表３
  誤：「**独**+**非**: **独立タグ**+**非表示タグ**」
  → 正：「**実**/**非**: **実体タグ**+**非表示タグ**」
  誤：「**独**/**非**」（５箇所）
  → 正：「**実**/**非**」
- ７頁，右段４行目
  誤：「（２）**独立**および**非表示**タグのみ」
  → 正：「（２）**実体**および**非表示**タグのみ」