Significance of Bridging Real-world Documents and NLP Technologies

Tadayoshi Hara Goran Topić Yusuke Miyao Akiko Aizawa
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
{harasan, goran_topic, yusuke, aizawa}@nii.ac.jp

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.

1 Introduction

Recent advances in natural language processing (NLP) technologies have allowed us to dream about applying these technologies to large-scale text, and then extracting a wealth of information from the text or enriching the text itself with various additional information. When actually considering the realization of this dream, however, we are faced with an inevitable problem. Conventional NLP tools usually assume an ideal situation where each input text consists of a plain word sequence, whereas real-world documents display text more expressively using a variety of layouts, sentence structures, and inline objects, among others. This means that obtaining valid input for a target NLP tool is left completely to the users, who have to program pre- and postprocessors for each application to convert their target text into the required format and integrate the output results into their original text. This additional effort reduces the viability of technologies, while the awkwardly obtained input does not allow the expected maximum benefit of the NLP technologies to be realized.

In this research, we raise awareness of this issue by developing a framework that simplifies this conversion and integration process. We assume that any textual composition beyond plain text is captured by tags in XML documents, and focus on the data conversion between XML-tagged text and the input/output formats of NLP tools. According to our observations, the data conversion process is determined by the textual functions of the XML-tags utilized in the target text, of which there seem to be only four types. We therefore devise a conversion strategy for each of the four types. After all tags in the XML tagset of the target text have been classified by the user into the four types, data conversion and integration can be executed automatically using our strategies, regardless of the size of the text (see Figure 1).

In the experiments, we apply our framework to several types of XML documents, and the results show that our framework can extract plain text sequences from the target XML documents by classifying only 20% or fewer of the total number of tag types. Furthermore, with the obtained sequences, two typical parsers succeed in processing entire documents with a much greater coverage rate and using much less parsing time than with naively tag-removed text.

Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT, pages 44–52,
Dublin, Ireland, August 23rd 2014.

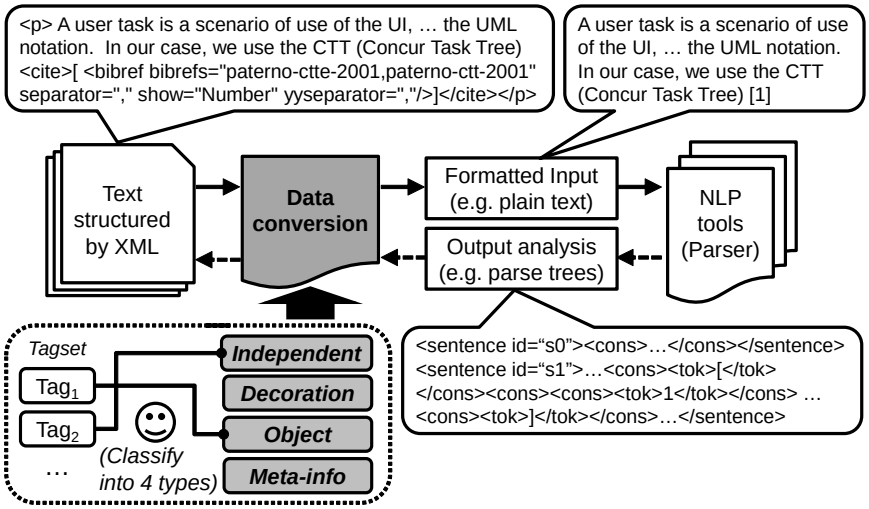


Figure 1: Proposed data conversion framework for applying NLP tools to text structured as XML

Tag type	Criteria for classification	Strategies for data conversion
Independent	To represent a region syntactically independent from the surrounding text	Remove the tag and tagged region → (apply tools to the tagged region independently) → recover the (analyzed) tagged region after applying the tools
Decoration	To set the display style of the tagged region at the same level as the surrounding text	Remove only the tag → recover the tag after applying the tools
Object	To represent the minimal object unit that should be handled in the the same level as the surrounding text	Replace the tag (and the tagged region) with a plain word → (do not process the tagged region further) → recover the tag (and region) after applying the tools
Meta-info	To describe the display style setting or additional information	Remove the tag and tagged region → (do not process the tagged region further) → recover the tag and region after applying the tools

Table 1: Four types of tags and the data conversion strategy for each type

The contribution of this work is to demonstrate the significance of bridging real-world documents and NLP technologies in practice. We show that, if supported by a proper framework, conventional NLP tools already have the ability to process real-world text without significant loss of performance. We expect the demonstration to promote further discussion on real-world document processing.

In Section 2, some related research attempts are introduced. In Section 3, the four types of textual functions for XML tags and our data conversion strategies for each of these are described and implemented. In Section 4, the efficiency of our framework and the adequacy of the obtained text sequences for use in NLP tools are examined using several types of documents.

2 Related Work

To the best of our knowledge, no significant work on a unified methodology for data conversion between target text and the input/output formats of NLP tools has been published. Some NLP tools provide scripts for extracting valid input text for the tools from real-world documents; however, even these scripts assume specific formats for the documents. For example, deep syntactic parsers such as the C&C Parser (Clark and Curran, 2007) and Enju (Ninomiya et al., 2007) assume POS-tagged sentences as input, and therefore the distributed packages for the parsers[1] contain POS-taggers together with the parsers. The POS-taggers assume plain text sentences as their input.

As the work most relevant to our study, UIMA (Ferruci et al., 2006) deals with various annotations in an integrated framework. However, in this work, the authors merely proposed the framework and did

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

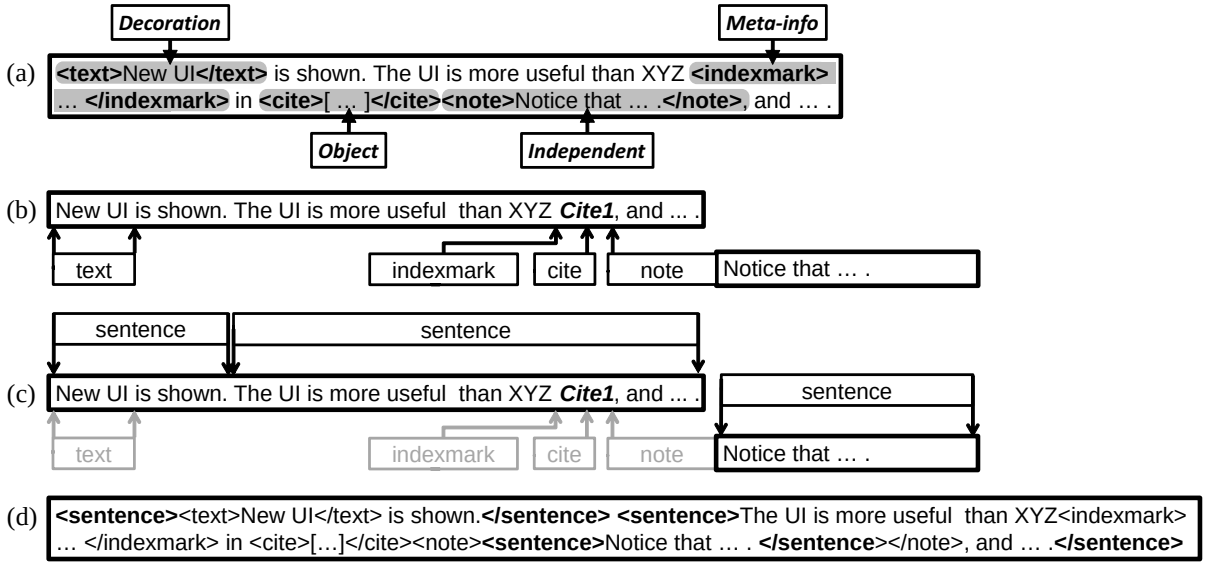


Figure 2: Example of executing our strategy

not explain how the given text can be used in a target annotation process such as parsing. Some projects based on the UIMA framework, such as RASP4UIMA (Andersen et al., 2008), U-compare (Kano et al., 2011), and Kachako (Kano, 2012)[2], have developed systems where the connections between various documents and various tools are already established. Users, however, can utilize only the text and tool pairs that have already been integrated into the systems. GATE (Cunningham et al., 2013) is based on a similar concept to UIMA; it supports XML documents as its input, while the framework also requires integration of tools into the systems.

In our framework, although availability of XML documents is assumed, a user can apply NLP tools to the documents without modifying the tools; instead, this is achieved by merely classifying the XML-tags in the documents into a small number of functional types.

3 Data Conversion Framework

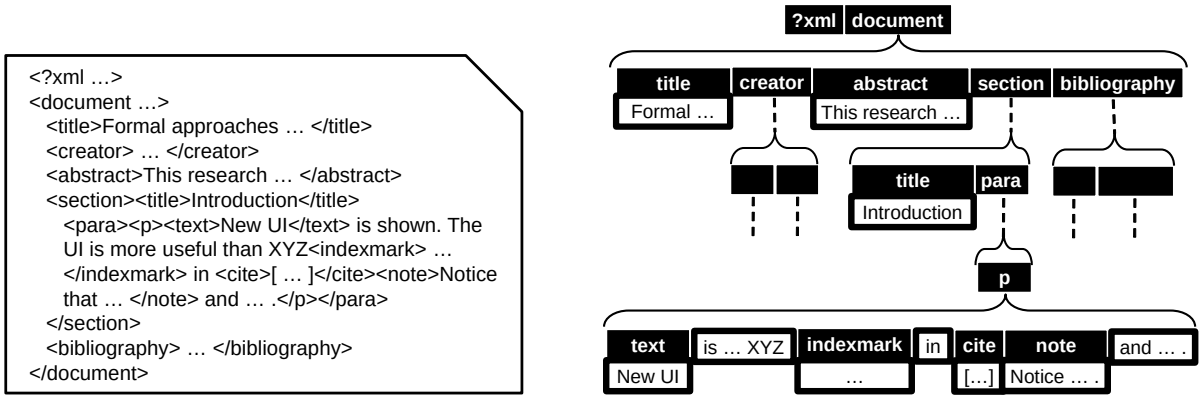
We designed a framework for data conversion between tagged text and the input/output formats of NLP tools based on the four types of textual functions of tags. First, we introduce the four tag types and the data conversion strategy for each. Then, we introduce the procedure for managing the entire data conversion process using the strategies.

3.1 Strategies for the Four Tag Types

The functions of the tags are classified into only four types, namely, Independent, Decoration, Object, and Meta-info, and for each of these types, a strategy for data conversion is described, as given in Table 1. This section explains the types and their strategies using a simple example where we attempt to apply a sentence splitter to the text given in Figure 2(a). The target text has four tags, "<*note*>", "<*text*>", "<*cite*>", and "<*indexmark*>", denoting, respectively, Independent, Decoration, Object, and Meta-info tags. We now describe each of the types.

Regions enclosed by Independent tags contain syntactically independent text, such as *titles*, *sections*, and so on. In some cases, a region of this type is inserted into the middle of another sentence, like the "<*note*>" tags in the example, which represent footnote text. The data conversion strategy for text containing these tags is to split the enclosed region into multiple subregions and apply the NLP tools separately to each subregion.

[2][U-compare]: http://u-compare.org/ / [Kachako]: http://kachako.org/kano/



(a) XML document (b) Structure of XML document

Figure 3: Example XML document

Decoration tags, on the other hand, do not necessarily guarantee the independence of the enclosed text regions, and are utilized mainly for embossing the regions visually, such as *changing the font or color of the text* ("*<text>*" in the example), *paragraphing sections*[3], and so on. The data conversion strategy for text containing these tags is to remove the tags before inputting the text into the NLP tools, and then to recover the tags afterwards[4].

Regions enclosed by Object tags contain special descriptions for representing objects treated as single syntactic components in the surrounding text. The regions do not consist of natural language text, and therefore cannot be analyzed by NLP tools[5]. The data conversion strategy for text containing this tag is to replace the enclosed region with some proper character sequence before inputting the text into NLP tools, and then to recover the replaced region afterwards.

Regions enclosed by Meta-info tags are not targets of NLP tools, mainly because the regions are not displayed, but utilized for other purposes, such as creating index pages (like this "*<indexmark>*")[6]. The data conversion strategy for text containing these tags is to delete the tagged region before inputting the text into NLP tools, and then to recover the region afterwards.

According to the above strategies, which are summarized in Table 1, conversion of the example text in Figure 2(a) is carried out as follows. In the first step, tags are removed from the text, whilst retaining their offsets in the resulting tag-less sequence shown in (b). "Cite1" in the sequence is a plain word utilized to replace the "*<cite>*" tag region. For the "*<note>*" tag, we recursively apply our strategies to its inner region, with two plain text regions "New UI ..." and "Notice that ..." consequently input into the sentence splitter. Thereafter, sentence boundary information is returned as shown in (c), and finally, using the retained offset information of the tags, the obtained analysis and original tag information are integrated to produce the XML-tagged sequence shown in (d).

3.2 Procedure for Efficient Tag Classification and Data Conversion

In actual XML documents as shown in Figure 3(a), a number of tags are introduced and tagged regions are multi-layered as illustrated in Figure 3(b) (where black and white boxes represent, respectively, XML tags and plain text regions, and regions enclosed by tags are placed below the tags in the order they appear.). We implemented a complete data conversion procedure for efficiently classifying tags in text documents into the four types and simultaneously obtaining plain text sequences from such documents,

[3]In some types of scientific articles, one sentence can be split into two *paragraph* regions. It depends on the target text whether a *paragraph* tag is classified as Independent or Decoration.

[4]The tags may imply that the enclosed regions constitute chunks of text, which may be suitable for use in NLP tools.

[5]In some cases, natural language text is used for parts of the descriptions, for example, *itemization* or *tables* in scientific articles. How the inner textual parts are generally associated with the surrounding text would be discussed in our future work. For the treatment of list structures, we can learn more from Aït-Mokhtar et al. (2003).

[6]If the tagged region contains analyzable text, it depends on the user policy whether NLP tools should be applied to the region, that is, whether to classify the tag as Independent.

```
@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;
}
```

Figure 4: Pseudo-code algorithm for data conversion from XML text to plain text for use in NLP tools

as given by the pseudo-code algorithm in Figure 4. In the remainder of this section, we explain how the algorithm works.

Our data conversion procedure applies the strategies for the four types of tags recursively from the top-level tags to the lower tags. The $@independent$, $@decoration$, $@object$ and $@meta\_info$ lists contain, respectively, Independent, Decoration, Object, and Meta-info tags, which have already been classified by the user. When applied to a target document, the algorithm uses the four lists and strategies given in the previous section in its first attempt at converting the document into plain text sequences, storing unknown (and therefore unprocessed) tags, if any, in $@unknown$. After the entire document has been processed for the first time, the user classifies any reported unknown tags. This process is repeated until no further unknown tags are encountered.

In the first iteration of processing the document in Figure 3(a) the algorithm is applied to the target document with the four tag lists empty. In the function "*data_convert*", top-level tags in the document, "*<?xml>*" and "*<document>*", are detected as yet-to-be classified tags and added to $@unknown$. The tags and their enclosed regions in the target document are replaced with unique plain text such as "UN1" and "UN2", and the input text thus becomes a sequence consisting of only plain words like "UN1 UN2". The algorithm then adds the sequence to $@plain\_text\_sequences$ and terminates. The user then classifies the reported yet-to-be classified tags in $@unknown$ into the four tag lists, and the algorithm

Article type	# articles	# total tags (# types)	# classified tags (# types)					#obtained seq.
			I	D	O	M	Total	
PMC	1,000	1,357,229(421)	32,109(12)	62,414(8)	48205(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
ACL	67	130,861(66*)	3,240(24)	14,064(29)	4,589(15)	2,304(19)	24,197(87)	2,293
Wiki.	300	223,514(60*)	3,530(12)	11,197(8)	1,470(28)	11,360(67)	27,557(115)	2,286

(**ArX.**: arXiv.org, **Wiki.**: Wikipedia, I: Independent, D: Decoration, O: Object, M: Meta-info)

Table 2: Classified tags and obtained sequences for each type of article

Article type	Treated tag classes	Parsing with Enju parser				Parsing with Stanford parser			
		* # sentences	** Time (s)	Avg. (**/*)	# failures (rate)	* # sentences	** Time (s)	Avg. (**/*)	# failures (rate)
PMC	**None**	159,327	209,783	1.32	4,721 (2.96%)	170,999	58,865	0.39	18,621 (10.89%)
	O/M	112,285	135,752	1.21	810 (0.72%)	126,176	50,741	0.44	11,881 (9.42%)
	All	126,215	132,250	1.05	699 (0.55%)	139,805	63,295	0.49	11,338 (8.11%)
ArX.	**None**	74,762	108,831	1.46	2,047 (2.74%)	75,672	27,970	0.43	10,590 (13.99%)
	O/M	41,265	89,200	2.16	411 (1.00%)	48,666	24,630	0.57	5,457 (11.21%)
	All	43,208	87,952	2.04	348 (0.81%)	50,504	26,360	0.58	5,345 (10.58%)
ACL	**None**	19,571	15,142	0.77	115 (0.59%)	17,166	5,047	0.29	1,095 (6.38%)
	O/M	9,819	9,481	0.97	63 (0.64%)	11,182	4,157	0.37	616 (5.51%)
	All	11,136	8,482	0.76	39 (0.35%)	12,402	4,871	0.39	587 (4.73%)
Wiki.	**None**	10,561	14,704	1.39	1,161 (10.99%)	14,883	3,114	0.24	1,651 (11.09%)
	O/M	5,026	6,743	1.34	67 (1.33%)	6,173	2,248	0.38	282 (4.57%)
	All	6,893	6,058	0.88	61 (0.88%)	8,049	2,451	0.31	258 (3.21%)

(**ArX.**: arXiv, **Wiki.**: Wikipedia, O/M: Object and Meta-info)

Table 3: Impact on parsing performance of plain text sequences extracted using classified tags

starts its second iteration[7].

In the case of Independent/Decoration tags, the algorithm splits the regions enclosed by the tags/removes only the tags from the target text, and recursively processes the obtained text sequence(s) according to our strategies. In the splitting/removal operation, the algorithm stores in $@recovery\_info$, the locations (offsets) in the obtained text where the tags should be inserted in order to recover the tags and textual structures after applying the NLP tools. In the case of Object/Meta-info tags, regions enclosed by these tags are replaced with unique plain text/omitted from the target text, which means that the inner regions are not unpacked and processed (with relevant information about the replacement/omitting process also stored in $@recovery\_info$). This avoids unnecessary classification tasks for tags that are utilized only in the regions, and therefore minimizes user effort.

When no further unknown tags are reported, sufficient tag classification has been done to obtain plain text sequences for input into NLP tools, with the sequences already stored in $@plain\_text\_sequences$. After applying NLP tools to the obtained sequences, $@recovery\_info$ is used to integrate the annotated output from the tools into the original XML document by merging the offset information[8], and consequently to recover the structure of the original document.

4 Experiments

We investigated whether the algorithm introduced in Section 3.2 is robustly applicable to different types of XML documents and whether the obtained text sequences are adequate for input into NLP tools. The results of this investigation highlight the significance of bridging real-world text and NLP technologies.

4.1 Target Documents

Our algorithm was applied to four types of XML documents: three types of scientific articles, examples of which were, respectively, downloaded from PubMed Central (PMC) (http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/), arXiv.org (http://arxiv.org/) and ACL Anthology

[7]The user can delay the classification for some tags to later iterations.

[8]When crossover of tag regions occur, the region in the annotated output is divided into subregions at the crossover point.

(http://anthology.aclweb.org/)[9], and a web page type, examples of which were downloaded from Wikipedia (http://www.wikipedia.org/). The articles obtained from PMC were originally given in an XML format, while those from arXiv.org and ACL Anthology were given in XHTML (based on XML), and those from Wikipedia were given in HTML, with the HTML articles generated via intermediate XML files. These four types of articles were therefore more or less based on valid XML (or XML-like) formats. For our experiments, we randomly selected 1,000 PMC articles, randomly selected 300 arXiv.org articles, collected 67 (31 long and 36 short) ACL 2014 conference papers without any conversion errors (see Footnote 9), and randomly downloaded 300 Wikipedia articles.

Each of the documents contained a variety of textual parts; we decided to apply the NLP tools to the titles of the articles and sections, abstracts, and body text of the main sections in the scientific articles, and to the titles of the articles, body text headings, and the actual body text of the Wikipedia articles. According to these policies, we classified the tags appearing in all articles of each type.

4.2 Efficiency of Tag Classification

Table 2 summarizes the classified tags and obtained sequences for each type of document. The second to ninth columns give the numbers of utilized articles, tags (in tokens and types) in the documents, each type of tag actually classified and processed, and obtained text sequences, respectively[10]. Using simple regular-expression matching, we found no remaining tagged regions in the obtained sequences. From this we concluded that our framework at least succeeded in converting XML-tagged text into plain text.

For the PMC articles, we obtained plain text sequences by classifying only a fifth or less of the total number of tag types, that is, focusing on less than 15% of the total tag occurrences in the documents (comparing the third and eighth columns). This is because the tags within the regions enclosed by Object and Meta-info tags were not considered by our procedure. For each of the arXiv.org, ACL and Wikipedia articles, a similar effect was implied by the fact that the number of classified tags was less than 20% of the total occurrences of all tags.

4.3 Adequacy of Obtained Sequences for Use in NLP Tools

We randomly selected several articles from each article type, and confirmed that the obtained text sequences consisted of valid *sentences*, which could be directly input into NLP tools and which thoroughly covered the content of the original articles. Then, to evaluate the impact of this adequacy in a more practical situation, we input the obtained sequences (listed in Table 2) into two typical parsers, namely, the Enju parser for deep syntactic/semantic analysis, and the Stanford parser (de Marneffe et al., 2006)[11] for phrase structure and dependency analysis[12]. Table 3 compares the parsing performance on three types of plain text sequences obtained by different strategies: simply removing all tags, processing Object and Meta-info tags using our framework and removing the remaining tags, and processing all the tags using our framework. For each combination of parser and article type, we give the number of detected sentences[13], the total parsing time, the average parsing time per sentence[14], and the number/ratio of sentences that could not be parsed[15].

For all article types, the parsers, especially the Enju parser, succeeded in processing the entire article with much higher coverage (see the fourth column for each parser) and in much less time (see the third

[9]The XHTML version of 178 ACL 2014 conference papers were available at ACL Anthology. Each of the XHTML files was generated by automatic conversion of the original article using LaTeXML (http://dlmf.nist.gov/LaTeXML/).

[10]For Wikipedia, arXiv.org and ACL articles, since HTML/XHTML tag names represent more abstract textual functions, the number of different tag types was much smaller than for PMC articles (see * in the table). To better capture the textual functions of the tagged regions, we used the combination of the tag name and its selected attributes as a single tag. The number of classified tags for Wikipedia, arXiv.org and ACL given in the table reflects this decision.

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

[12]The annotated output from the parsers was integrated into the original XML documents by merging the offset information, and the structures of the original documents were consequently recovered. The recovered structures were input to *xmllint*, a UNIX tool for parsing XML documents, and the tool succeeded in parsing the structures without detecting any error.

[13]For the Enju parser, we split each text sequence into sentences using GeniaSS [http://www.nactem.ac.uk/y-matsu/geniass/].

[14]For the Enju parser, the time spent parsing failed sentences was also considered.

[15]For the Stanford parser, the maximum sentence length was limited to 50 words using the option settings because several sentences caused parsing failures, even after increasing the memory size from 150 MB to 2 GB, which terminated the whole process.

column for each parser) using the text sequences obtained by treating some (Object and Meta-info) or all tags with our framework than with those sequences obtained by merely removing the tags. This is mainly because the text sequences obtained by merely removing the tags contained some embedded inserted sentences (specified by Independent tags), bare expressions consisting of non natural language (non-NL) principles (specified by Object tags), and sequences not directly related to the displayed text (specified by Meta-info tags), which confused the parsers. In particular, treating Object and Meta-info tags drastically improved parsing performance, since non-NL tokens were excluded from the analysis.

Compared with treating Object/Meta-info tags, treating all tags, that is, additionally treating Independent tags and removing the remaining tags as Decoration tags, increased the number of detected sentences. This is because Independent tags provide solid information for separating text sequences into shorter sequences and thus prompting the splitting of sequences into shorter sentences, which decreased parsing failure by preventing a lack of search space for the Enju parser and by increasing target (≤ 50 word) sentences for the Stanford parser. Treating all tags increased the total time for the Stanford parser since a decrease in failed (> 50 word) sentences directly implied an increase in processing cost, whereas, for the Enju parser, the total time decreased since the shortened sentences drastically narrowed the required search space.

4.4 Significance of Bridging Real-world Documents and NLP Technologies

As demonstrated above, the parsers succeeded in processing the entire article with much higher coverage and in much less time with the text sequences obtained by our framework than with those sequences obtained by merely removing the tags. Then, what does such thorough and efficient processing bring about? If our target is shallow analysis of documents which can be achieved by simple approaches such as counting words, removing tags will suffice; embedded sentences do not affect word count, and non-NL sequences can be canceled by a large amount of valid sequences in the target documents.

Such shallow approaches, however, cannot satisfy the demands on more detailed or precise analysis of documents: discourse analysis, translation, grammar extraction, and so on. In order to be sensitive to subtle signs from the documents, information uttered even in small parts of text cannot be overlooked, under the condition that sequences other than body text are excluded.

This process of plain text extraction is a well-established procedure in NLP research; in order to concentrate on precise analysis of natural language phenomena, datasets have been arranged in the format of plain text sequences, and, using those datasets, plenty of remarkable achievements have been reported in various NLP tasks while brand-new tasks have been found and tackled.

But what is the ultimate goal of these challenges? Is it to just parse carefully arranged datasets? We all know this to be just a stepping stone to the real goal: to parse real-world, richly-formatted documents. As we demonstrated, if supported by a proper framework, conventional NLP tools already have the ability to process real-world text without significant loss of performance. Adequately bridging target real-world documents and NLP technologies is thus a crucial task for taking advantage of full benefit brought by NLP technologies in ubiquitous application of NLP.

5 Conclusion

We proposed a framework for data conversion between XML-tagged text and input/output formats of NLP tools. In our framework, once each tag utilized in the XML-tagged text has been classified as one of the four types of textual functions, the conversion is automatically done according to the classification. In the experiments, we applied our framework to several types of documents, and succeeded in obtaining plain text sequences from these documents by classifying only a fifth of the total number of tag types in the documents. We also observed that with the obtained sequences, the target documents were much more thoroughly and efficiently processed by parsers than with naively tag-removed text. These results emphasize the significance of bridging real-world documents and NLP technologies.

We are now ready for public release of a tool for conversion of XML documents into plain text sequences utilizing our framework. We would like to share further discussion on applying NLP tools to various real-world documents for increased benefits from NLP.

Acknowledgements

This research was partially supported by “Data Centric Science: Research Commons” at the Research Organization of Information and Systems (ROIS), Japan.

References

Salah Aït-Mokhtar, Veronika Lux, and Éva Bánik. 2003. Linguistic parsing of lists in structured documents. In *Proceedings of Language Technology and the Semantic Web: 3rd Workshop on NLP and XML (NLPXML-2003)*, Budapest, Hungary, April.

Ø. Andersen, J. Nioche, E.J. Briscoe, and J. Carroll. 2008. The BNC parsed with RASP4UIMA. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC 2008)*, pages 865–869, Marrakech, Morocco, May.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. 2013. Getting more out of biomedical documents with GATE’s full lifecycle open source text analytics. *PLoS Comput Biol*, 9(2).

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC 2006)*, pages 449–454, Genoa, Italy, May.

David Ferruci, Adam Lally, Daniel Gruhl, Edward Epstein, Marshall Schor, J. William Murdock, Andy Frenkiel, Eric W. Brown, Thomas Hampp, Yurdaer Doganata, Christopher Welty, Lisa Amini, Galina Kofman, Lev Kozakov, and Yosi Mass. 2006. Towards an interoperability standard for text and multi-modal analytics. Technical Report RC24122, IBM Research Report.

Yoshinobu Kano, Makoto Miwa, Kevin Cohen, Larry Hunter, Sophia Ananiadou, and Jun’ichi Tsujii. 2011. U-Compare: a modular NLP workflow construction and evaluation system. *IBM Journal of Research and Development*, 55(3):11:1–11:10.

Yoshinobu Kano. 2012. Kachako: a hybrid-cloud unstructured information platform for full automation of service composition, scalable deployment and evaluation. In *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, November.

Takashi Ninomiya, Takuya Matsuzaki, Yusuke Miyao, and Jun’ichi Tsujii. 2007. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. In *Proceedings of the 10th International Conference on Parsing Technologies (IWPT’07)*, Prague, Czech Republic, June.