Significance of Bridging Real-world Documents and NLP Technologies

Tadayoshi Hara Goran Topić
Yusuke Miyao Akiko Aizawa
National Institute of Informatics, Japan

Goal of Research

- Analyze any documents with NLP tools

2 Data Construction: A Dispute Corpus

We construct the first dispute detection corpus to date; it consists of dispute and non-dispute discussions from Wikipedia Talk pages.

Step 1: Get Talk Pages of Disputed Articles. Wikipedia articles are edited by different editors. If an article is observed to have disputes on its *talk page*, editors can assign dispute tags to the article to flag it for attention. In this research, we are interested in talk pages whose corresponding articles are labeled with the following tags: DISPUTED, TOTALLYDISPUTED, DISPUTED-SECTION, TOTALLYDISPUTED-SECTION, POV. The tags indicate that an article is disputed, or the neutrality of the article is disputed (POV).

We use the 2013-03-04 Wikipedia data dump, and extract talk pages for articles that are labeled with dispute tags by checking the revision history. This results in 19,071 talk pages.

Step 2: Get Discussions with Disputes. Dispute tags can also be added to *talk pages* themselves. Therefore, in addition to the tags mentioned above, we also consider the "Request for Comment" (RFC) tag on talk pages. According to Wikipedia[4], RFC is used to request outside opinions concerning the disputes.

3609 discussions are collected with dispute tags found in the revision history. We further classify dispute discussions into three subcategories: CONTROVERSY, REQUEST FOR COMMENT (RFC), and RESOLVED based on the tags found in discussions (see Table 1). The numbers of discussions for the three types are 42, 3484, and 105, respectively. Note that dispute tags only appear in a small number of articles and talk pages. There may exist other discussions with disputes.

Dispute Subcategory	Wikipedia Tags on Talk pages
Controversy	CONTROVERSIAL, TOTALLYDISPUTED, DISPUTED, CALM TALK, POV
Request for Comment	RFC
Resolved	Any tag from above + RESOLVED

Table 1: Subcategory for disputes with corresponding tags. Note that each

ACL 2014 paper (XHTML)

Goal of Research

- Analyze any documents with NLP tools

2 Data Construction: A Dispute Corpus

We construct the first dispute detection corpus to date; it consists of dispute and non-dispute discussions from Wikipedia Talk pages.

Step 1: Get Talk Pages of Disputed Articles. Wikipedia articles are edited by different editors. If an article is observed to have disputes on its *talk page*, editors can assign dispute tags to the article to flag it for attention. In this research, we are interested in talk pages whose corresponding articles are labeled with the following tags: DISPUTED, TOTALLYDISPUTED, DISPUTED-SECTION, TOTALLYDISPUTED-SECTION, POV. The tags indicate that an article is disputed, or the neutrality of the article is disputed (POV).

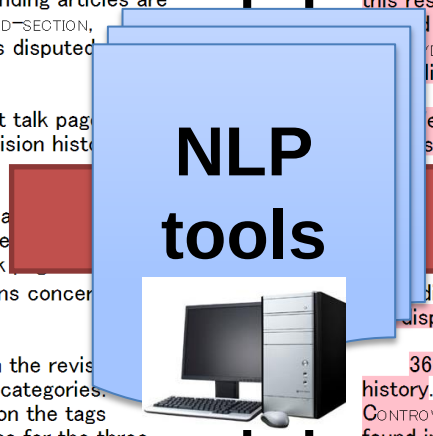
We use the 2013-03-04 Wikipedia data dump, and extract talk pages for articles that are labeled with dispute tags by checking the revision history. This results in 19,071 talk pages.

Step 2: Get Discussions with Disputes. Dispute tags can also be added to *talk pages* themselves. Therefore, in addition to the tags mentioned above, we also consider the "Request for Comment" (RFC) tag on talk pages. According to Wikipedia[4], RFC is used to request outside opinions concerning the disputes.

3609 discussions are collected with dispute tags found in the revision history. We further classify dispute discussions into three subcategories: CONTROVERSY, REQUEST FOR COMMENT (RFC), and RESOLVED based on the tags found in discussions (see Table 1). The numbers of discussions for the three types are 42, 3484, and 105, respectively. Note that dispute tags only appear in a small number of articles and talk pages. There may exist other discussions with disputes.

Dispute Subcategory	Wikipedia Tags on Talk pages
Controversy	CONTROVERSIAL, TOTALLYDISPUTED, DISPUTED, CALM TALK, POV
Request for Comment	RFC
Resolved	Any tag from above + RESOLVED

Table 1: Subcategory for disputes with corresponding tags. Note that each



ACL 2014 paper
(XHTML)

2 Data Construction: A Dispute Corpus

We construct the first dispute detection corpus to date; it consists of dispute and non-dispute discussions from Wikipedia Talk pages.

Step 1: Get Talk Pages of Disputed Articles. Wikipedia articles are edited by different editors. If an article is observed to have disputes on its talk page, editors can assign dispute tags to the article to flag it for attention. In this research, we are interested in talk pages whose corresponding articles are labeled with the following tags: DISPUTED, TOTALLYDISPUTED, DISPUTED-SECTION, TOTALLYDISPUTED-SECTION, POV. The tags indicate that an article is disputed, or the neutrality of the article is disputed (POV).

We use the 2013-03-04 Wikipedia data dump, and extract talk pages for articles that are labeled with dispute tags by checking the revision history. This results in 19,071 talk pages.

Step 2: Get Discussions with Disputes. Dispute tags can also be added to talk pages themselves. Therefore, in addition to the tags mentioned above, we also consider the "Request for Comment" (RFC) tag on talk pages. According to Wikipedia[4], RFC is used to request outside opinions concerning the disputes.

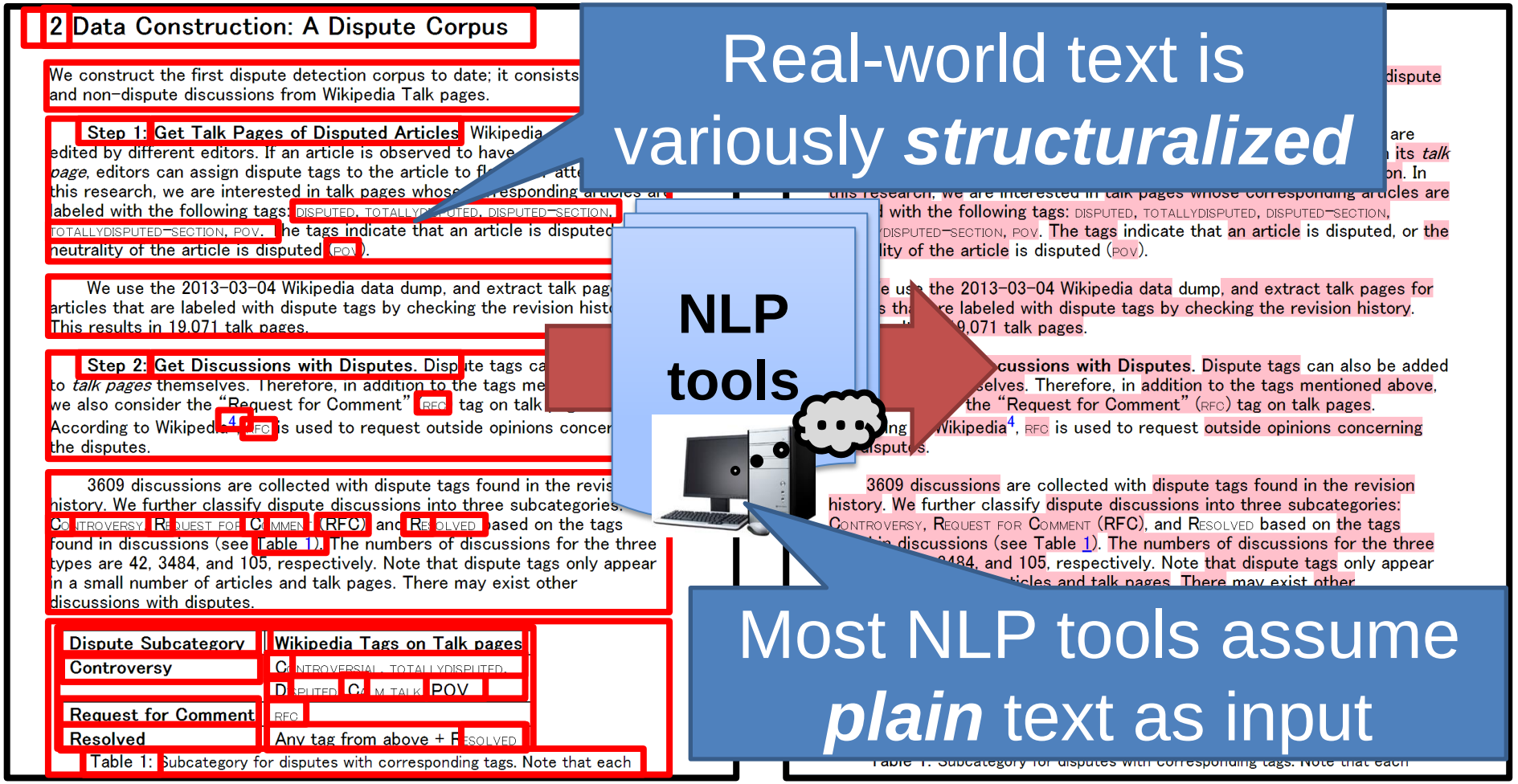
3609 discussions are collected with dispute tags found in the revision history. We further classify dispute discussions into three subcategories: CONTROVERSY, REQUEST FOR COMMENT (RFC), and RESOLVED based on the tags found in discussions (see Table 1). The numbers of discussions for the three types are 42, 3484, and 105, respectively. Note that dispute tags only appear in a small number of articles and talk pages. There may exist other

Dispute Subcategory	Wikipedia Tags on Talk pages
Controversy	CONTROVERSIAL, TOTALLYDISPUTED, DISPUTED, CALM TALK, POV
Request for Comment	RFC
Resolved	Any tag from above + RESOLVED

Table 1: Subcategory for disputes with corresponding tags. Note that each

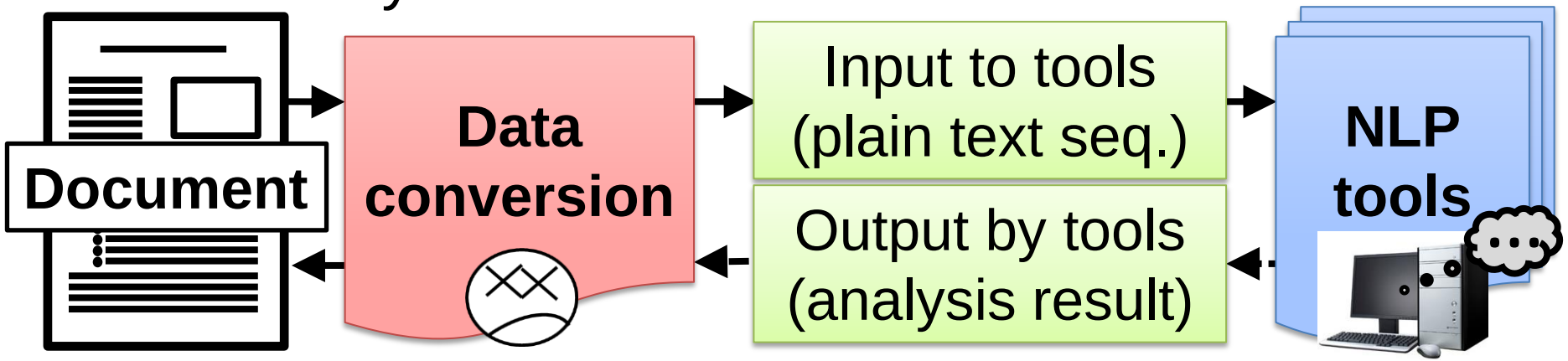
Goal of Research

- Analyze any documents with NLP tools



Goal of Research

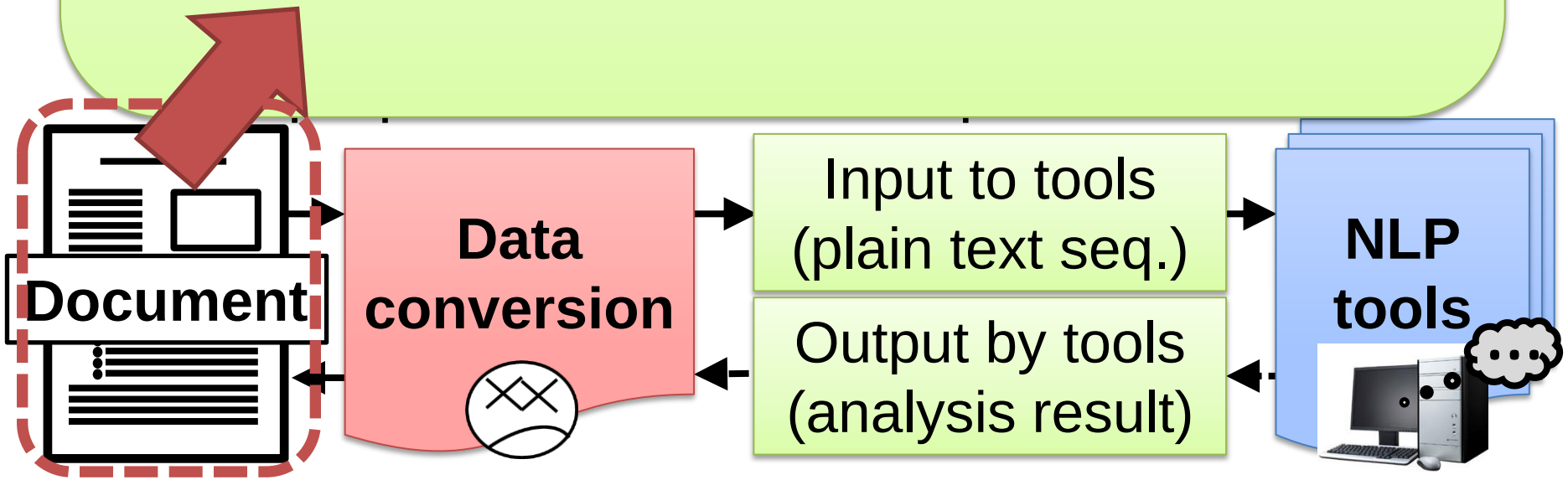
- Analyze any documents with NLP tools
  - → Real-world text is variously ***structuralized***
  - → Most NLP tools assume ***plain*** text as input
- Data conversion is required (up to users)
  - – Programming for every target is bothersome
  - – Rudely converted text can confuse NLP tools



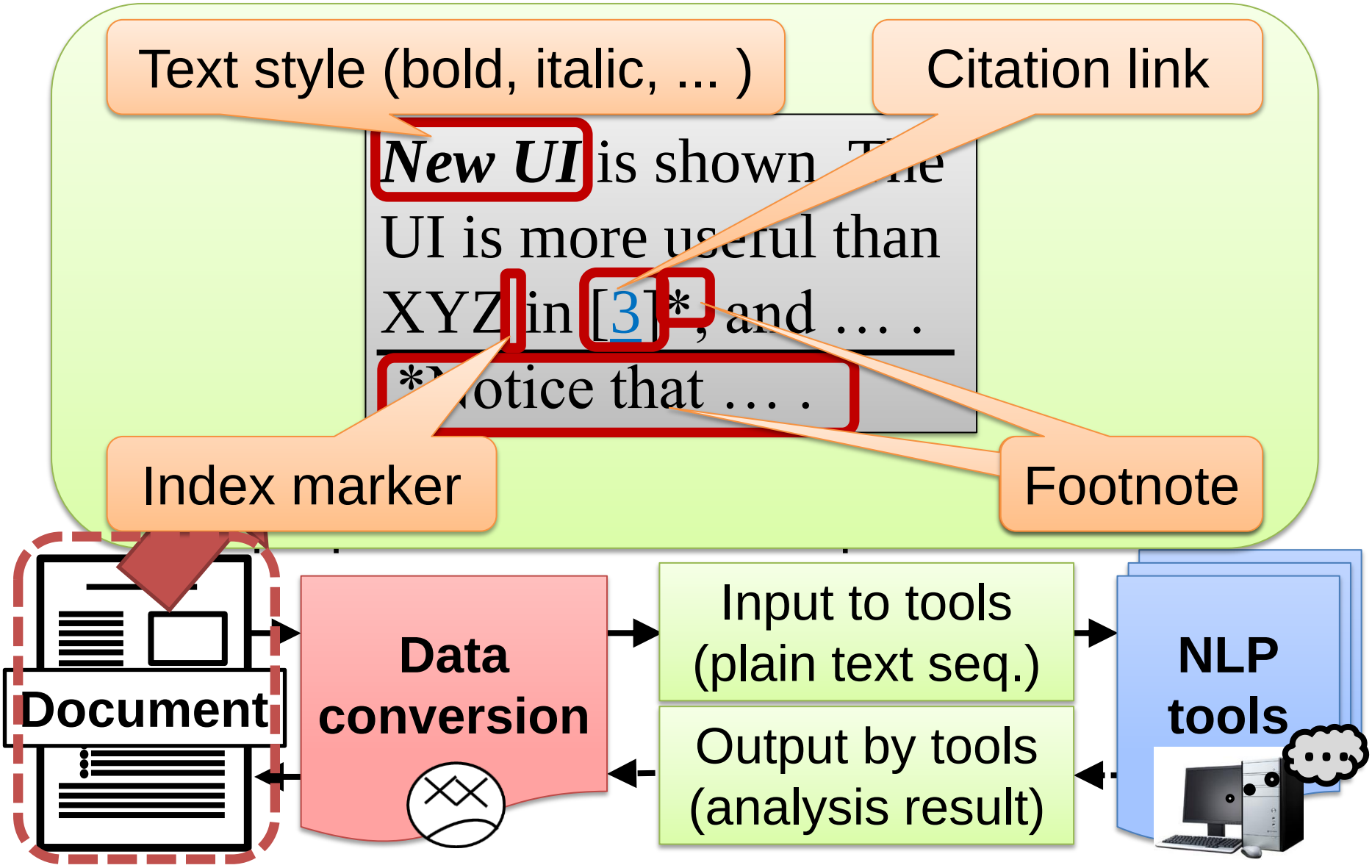
Goal of Research

New UI is shown. The UI is more useful than XYZ in [3]*, and … .

*Notice that … .



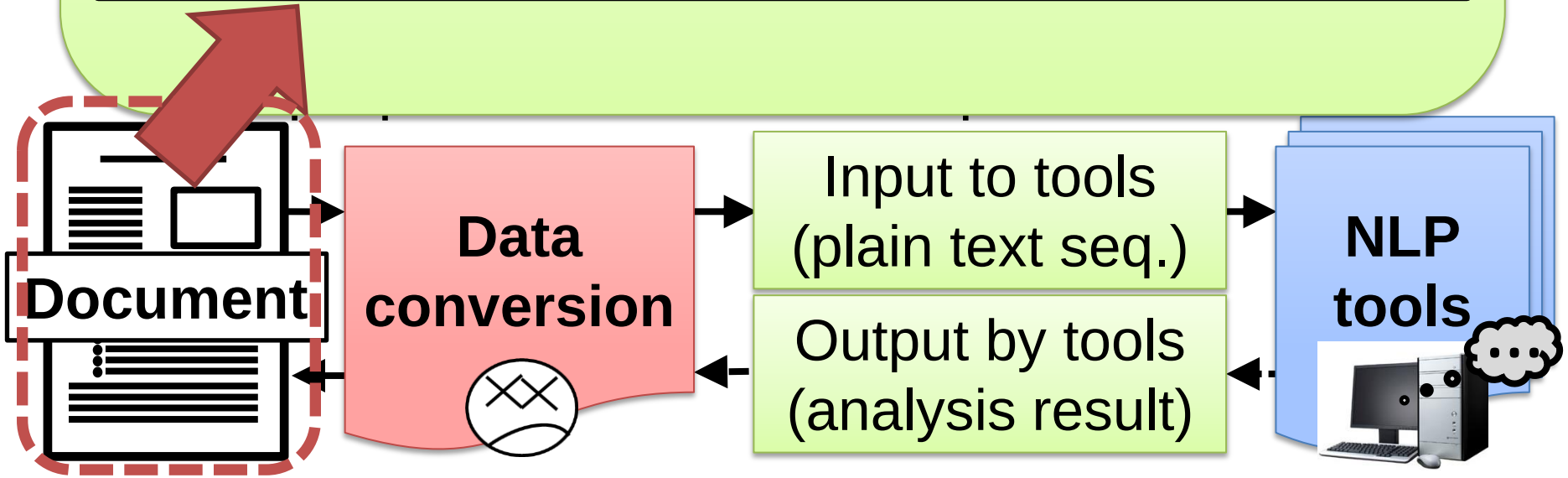
Goal of Research



Goal of Research

Tagged text

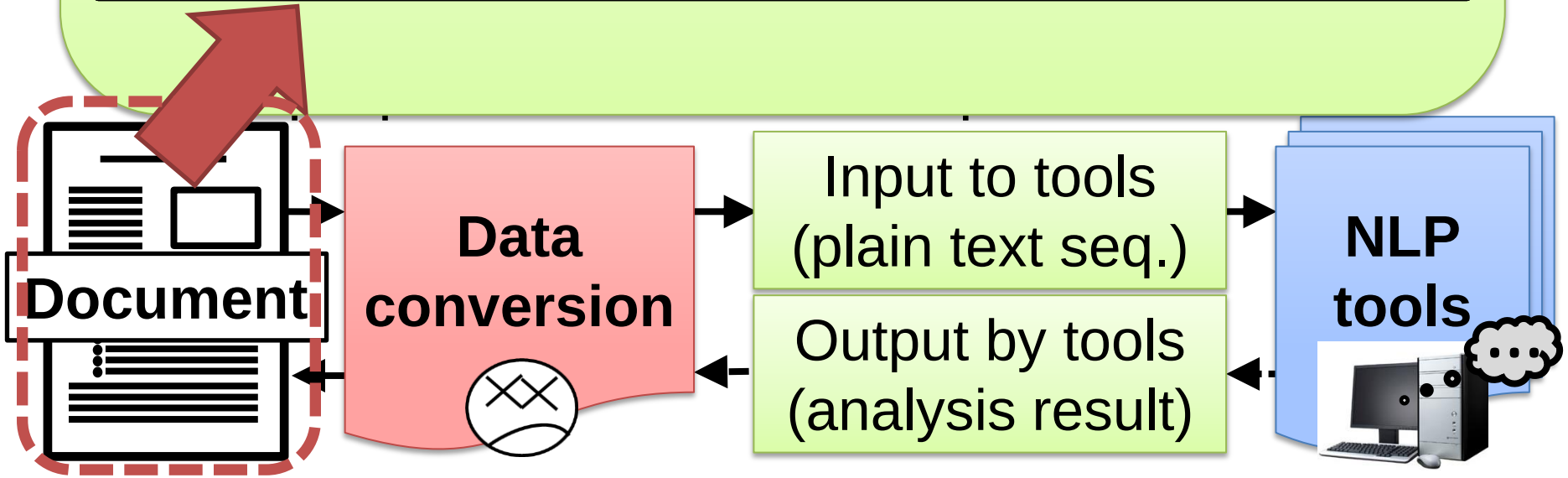
<text>New UI**</text>** is shown. The UI is more useful than XYZ**<index>**##**</index>** in **<cite>**[…]**</cite>** **<note>**Notice that … .**</note>**, and …



Goal of Research

Plain (?) text

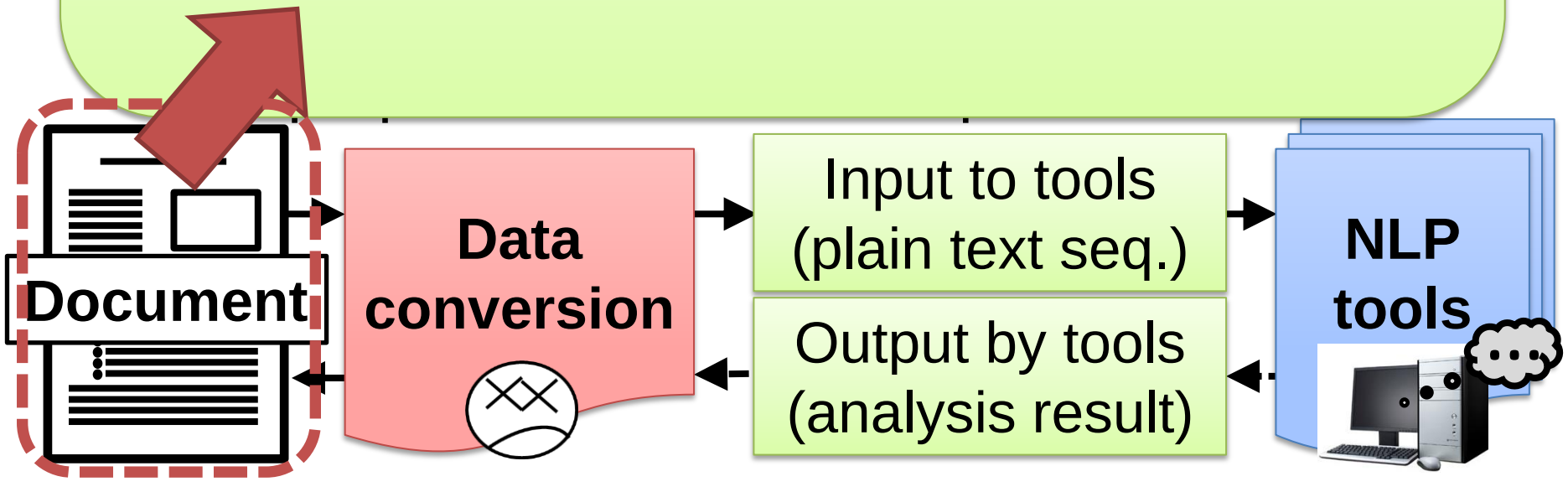
New UI is shown. The UI is more useful than XYZ ## in […] Notice that … . , and …



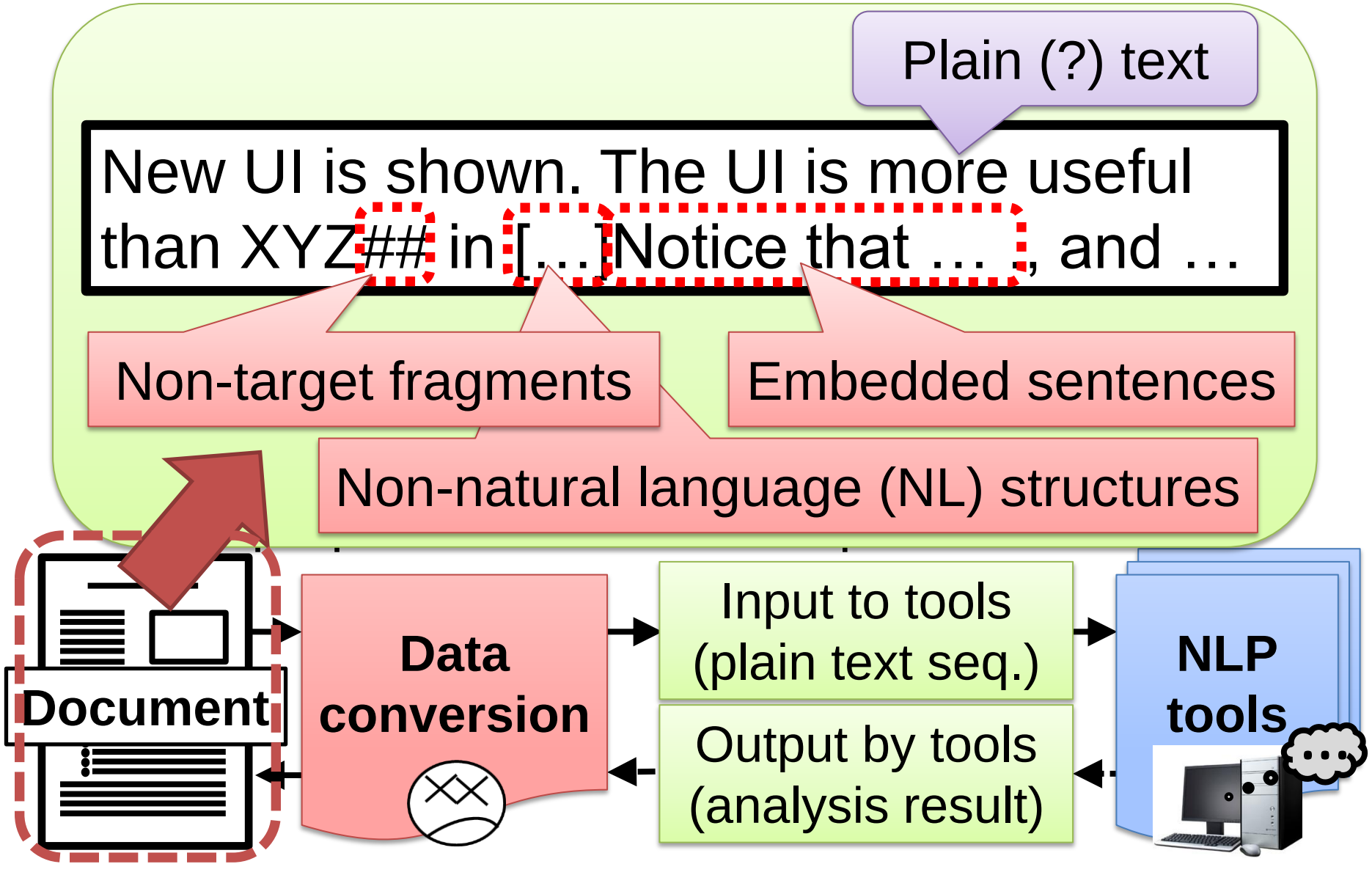
Goal of Research

Plain (?) text

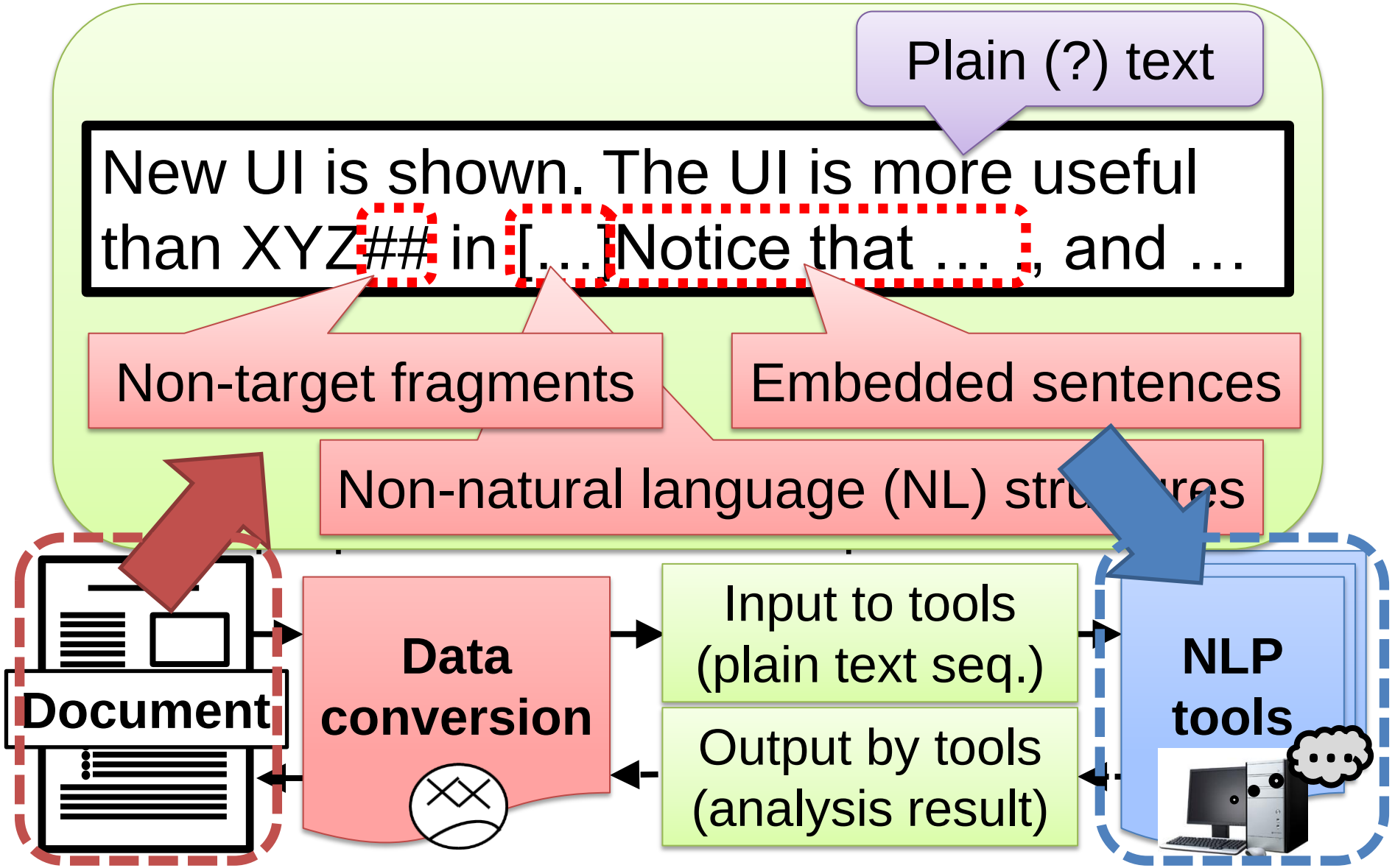
New UI is shown. The UI is more useful than XYZ## in […]Notice that … ., and …



Goal of Research



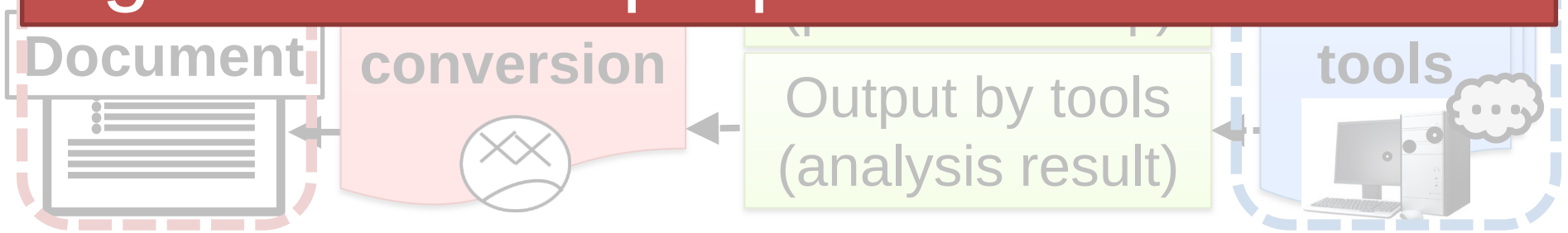
Goal of Research



Goal of Research

- Proper data conversion should be important task for applying NLP tools to real-world documents
  → Not extensively tackled so far

Promote discussion by demonstrating significance of proper data conversion



Outline

- **Related work** & **Our objective**
- **Our framework**
  - Extracting plain text sequences from XML-tagged text based on manual tag classification
- **Experimental results**
  - Extracting plain text sequences from documents
  - Applying parsers to obtained sequences
- **Discussion**
  - Significance of bridging real-world documents and NLP technologies

Related Work on Unified Methodology for Data Conversion

- (Not extensively tackled so far)
- Some NLP tools provide conversion scripts (e.g. parser with POS-tagger*[1,2])

  → Even the scripts assume plain-text input
- Some frameworks enable us to apply various NLP tools to various documents (e.g. UIMA*[3,4,5,6]/ GATE*[7])

  → Tools should be incorporated beforehand

---

[1] C&C (Clark et al., 2007),[2] Enju (Ninomiya et al., 2007),*[3] Ferruci et al.(2006)
[4] RASP4UIMA (Andersen et al., 2008),[5] U-compare (Kano et al., 2011)
[6] Kachako (Kano, 2012),[7] Cunningham et al.(2013)

Objective & Approach

Objective:

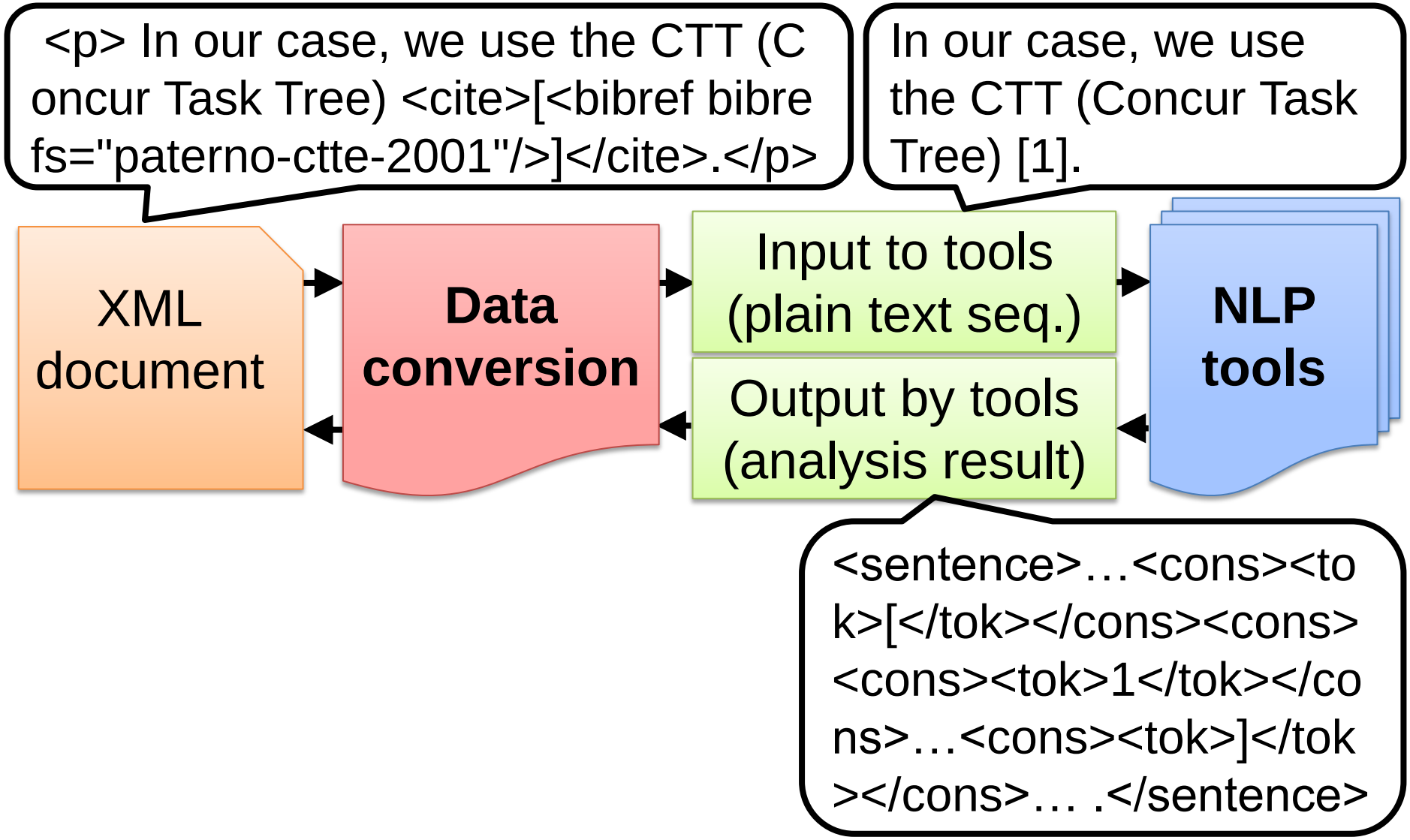
Show significance of proper data conversion

Approach:

1. Focus on XML documents
   – XML-tags provide structures beyond plain text
2. Propose framework for applying NLP tools to documents without modifying the tools

→ Exemplify impact through experiments

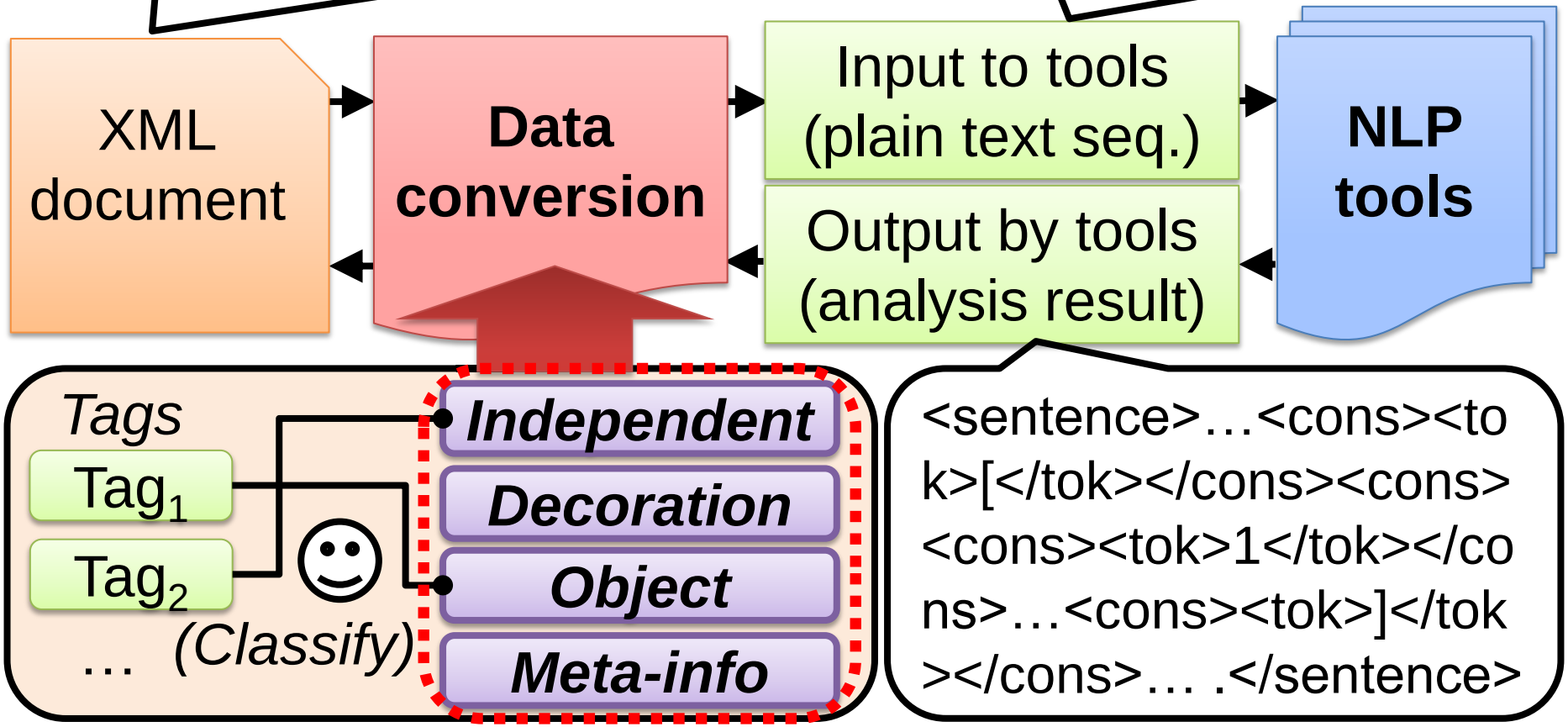
Overview of Our Framework



Overview of Our Framework

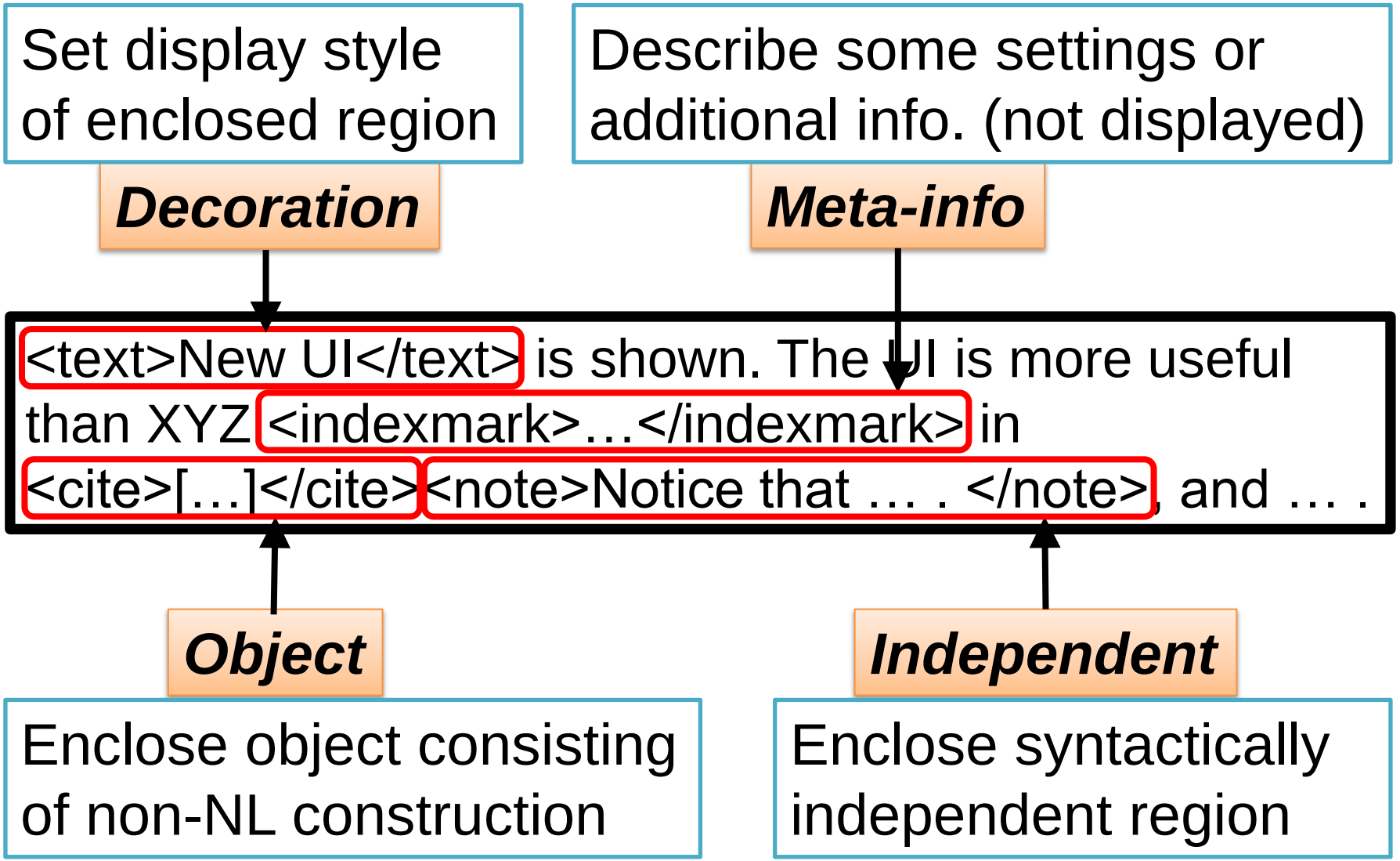
 In our case, we use the CTT (C oncur Task Tree) <cite>[<bibref bibre fs="paterno-ctte-2001"/>]</cite>.

In our case, we use the CTT (Concur Task Tree) [1].

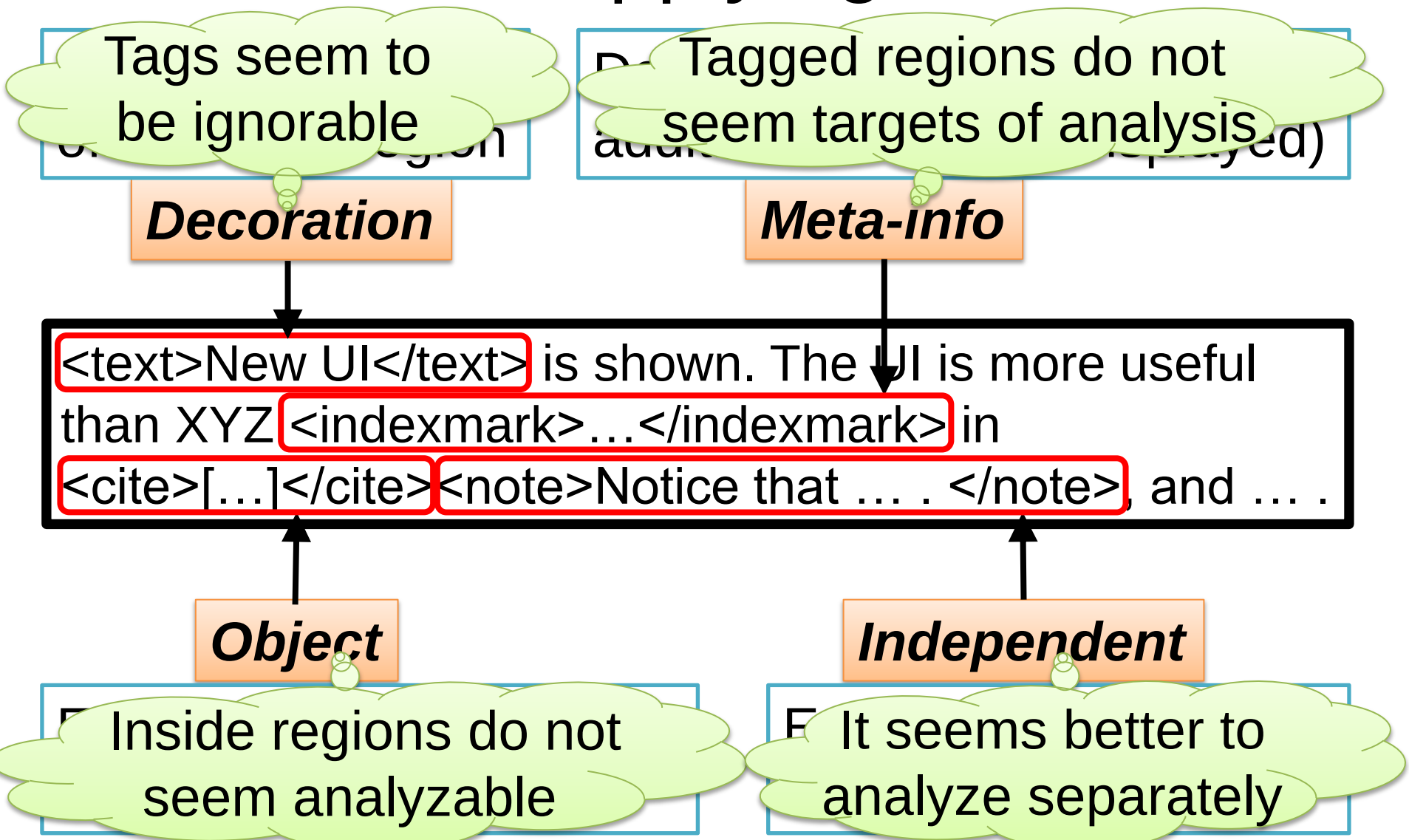


Classify into 4 functional types → auto-conversion

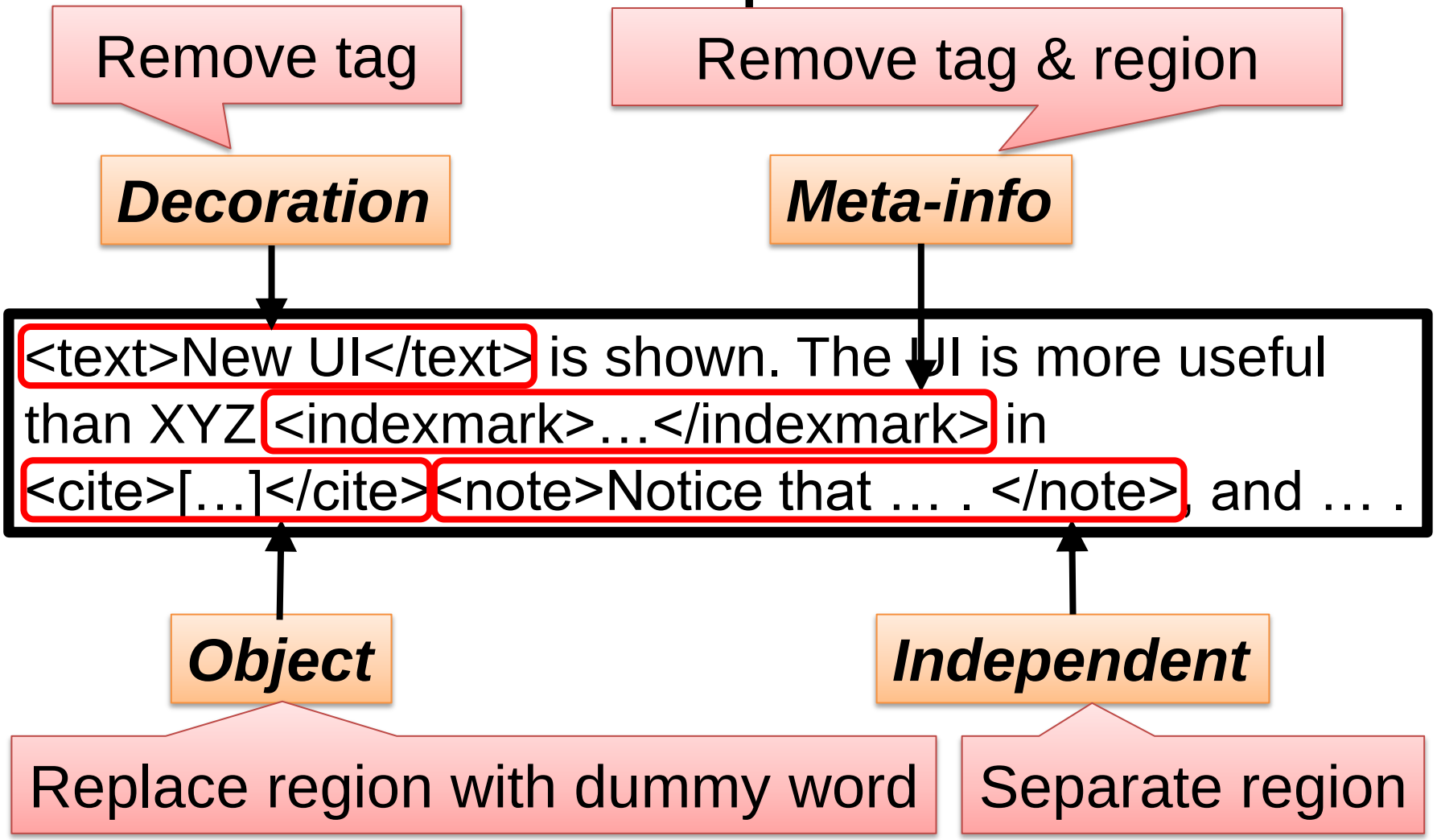
Four Types of Textual Functions



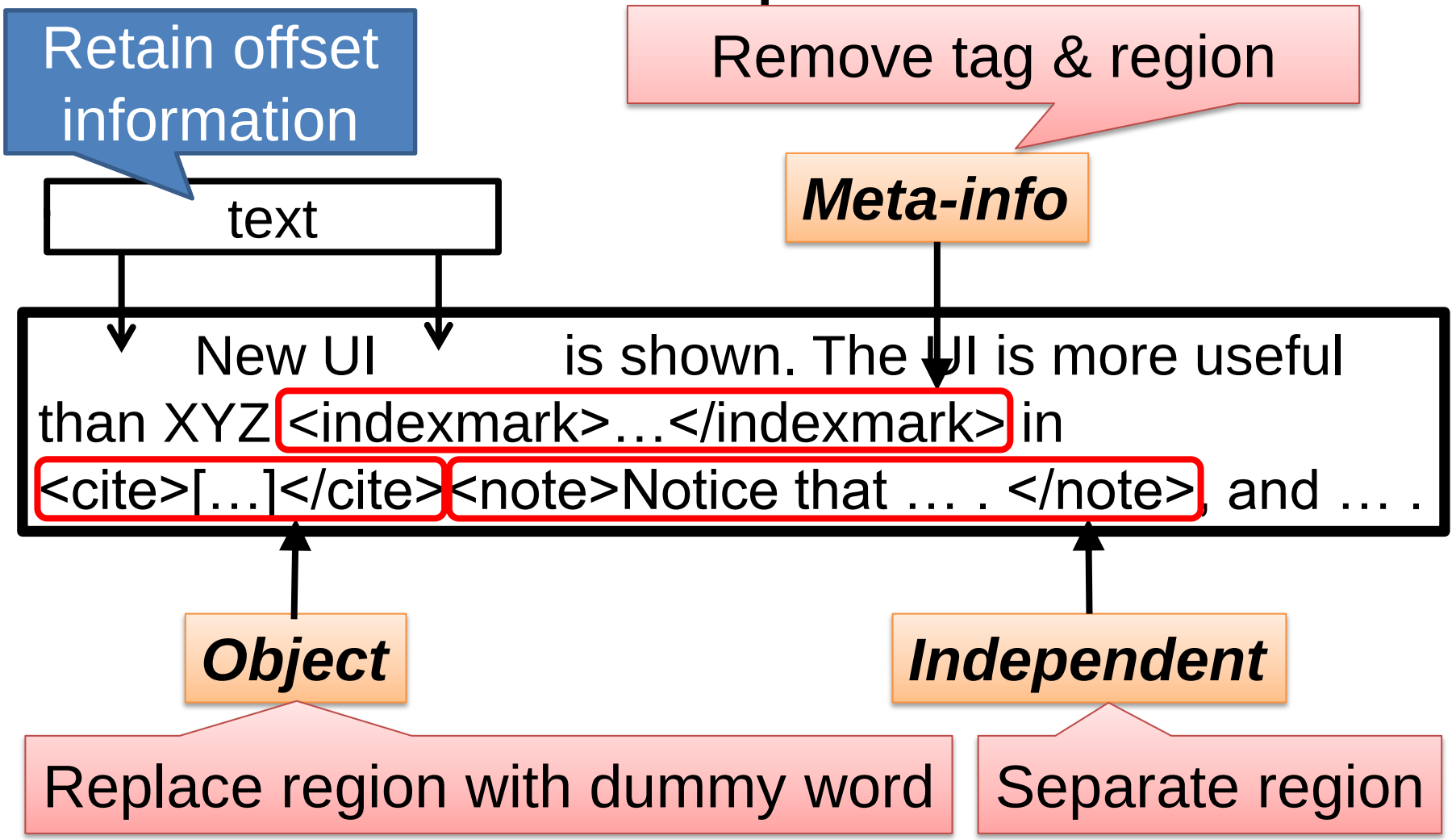
Intuition for Applying NLP Tools



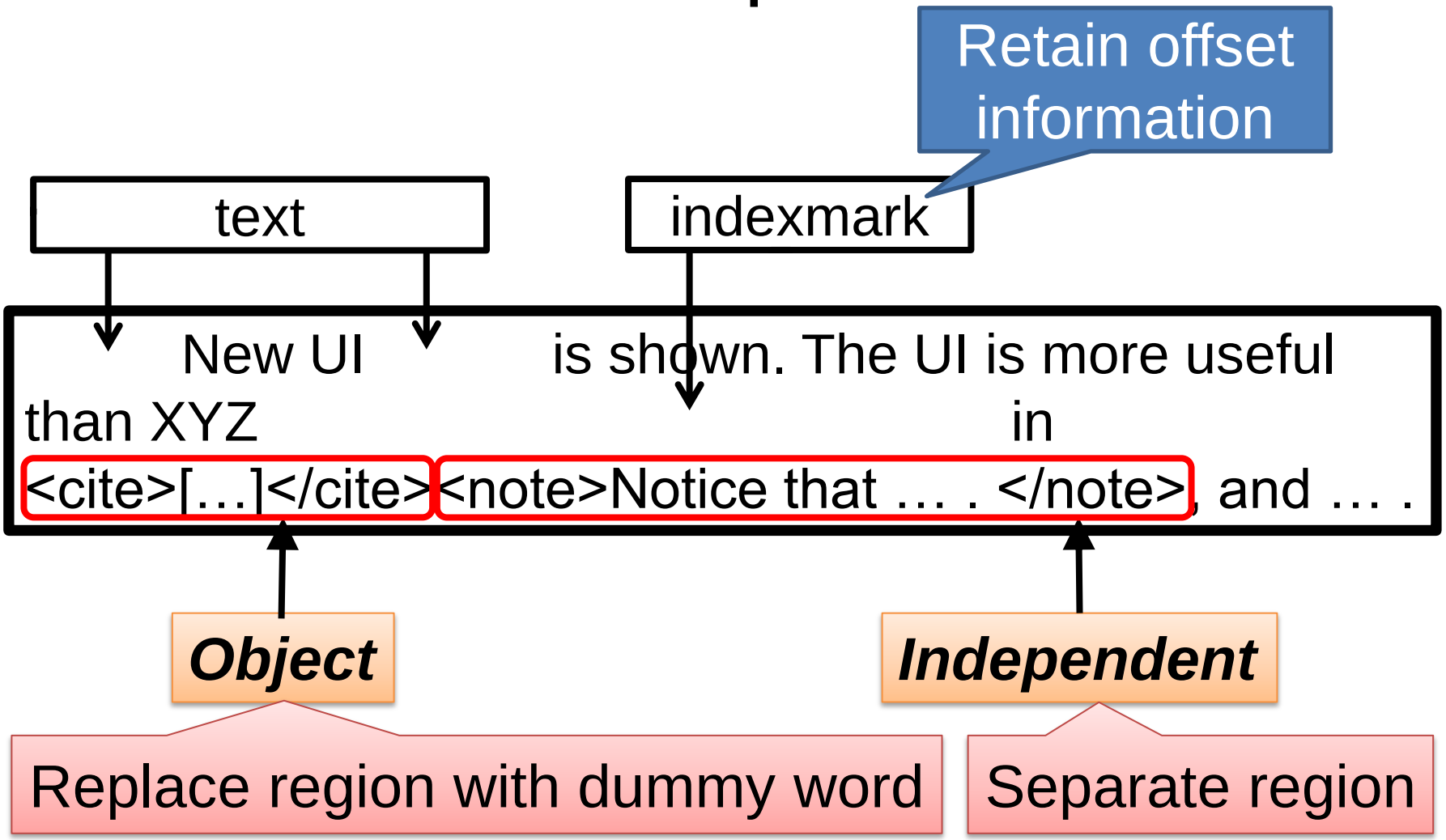
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



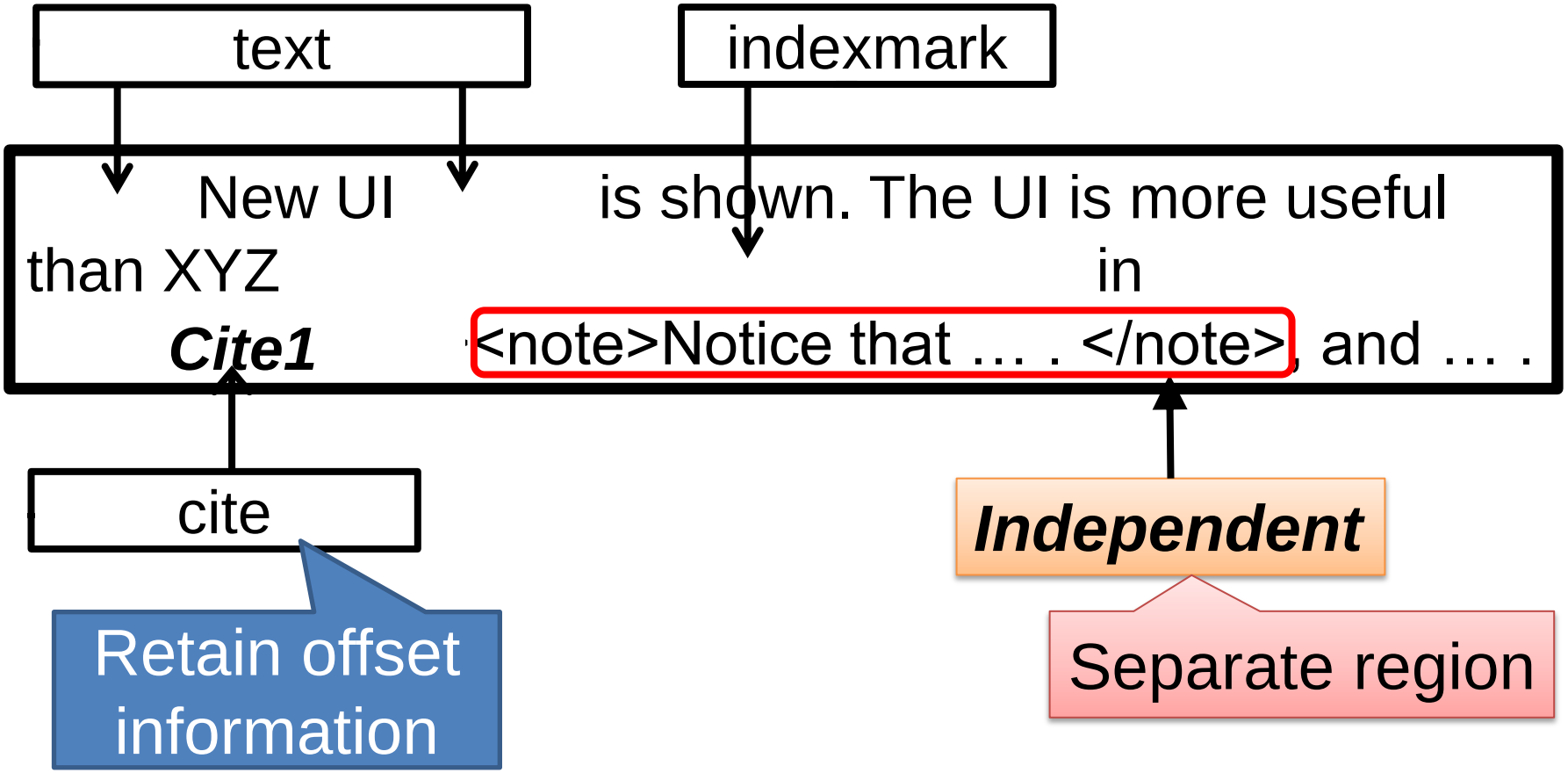
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



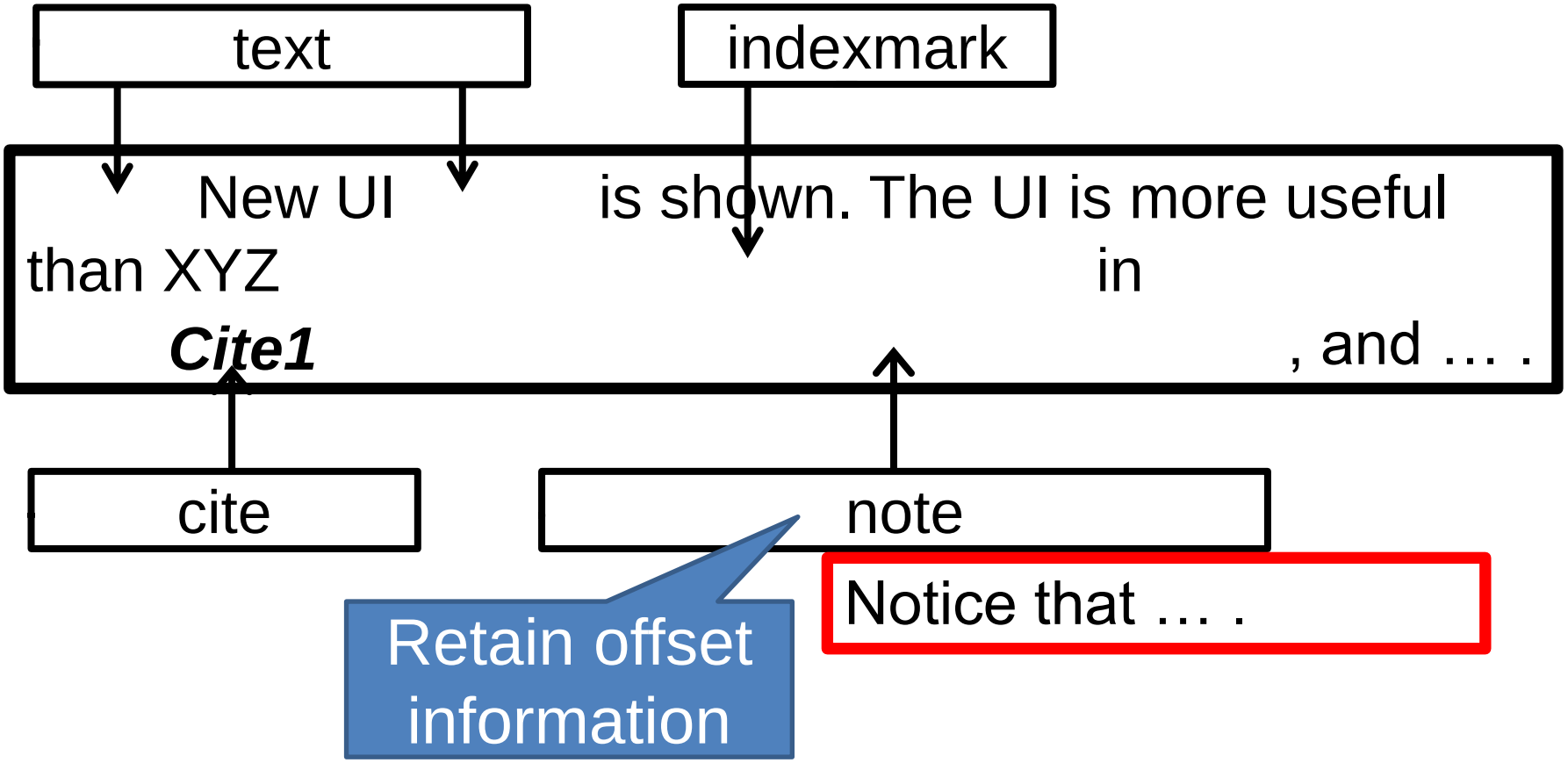
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



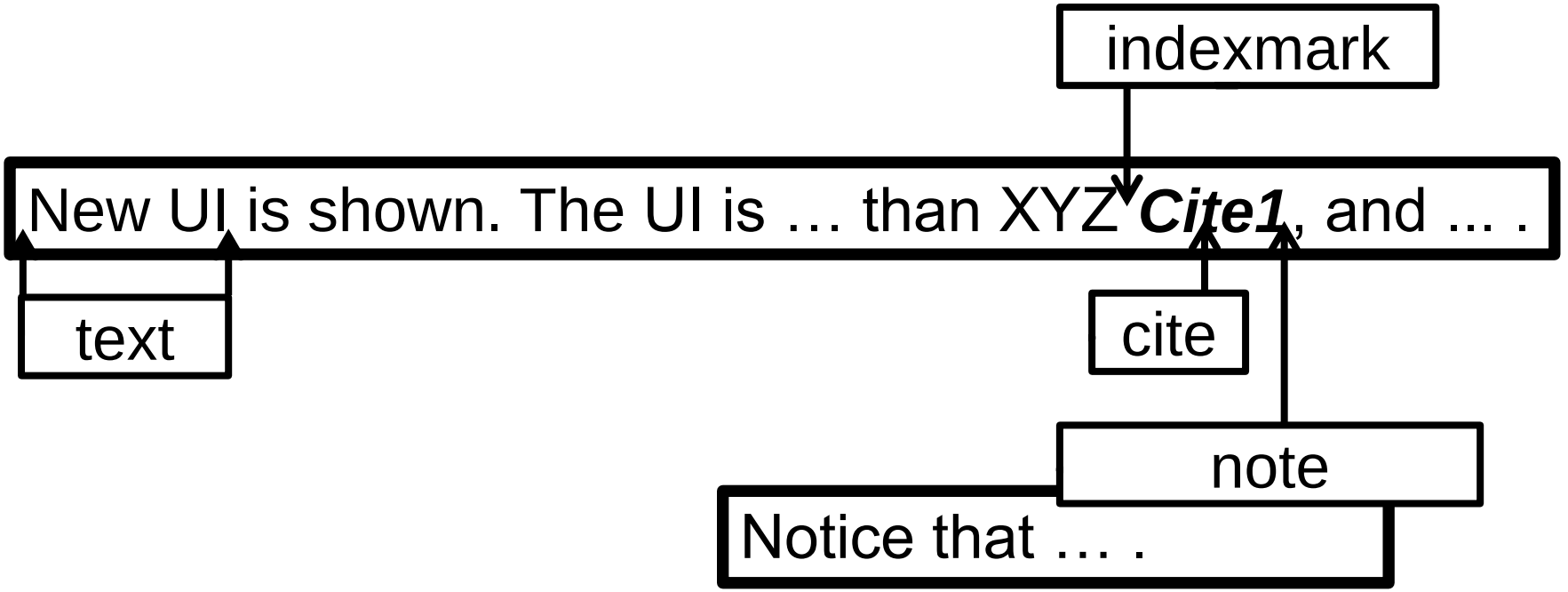
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



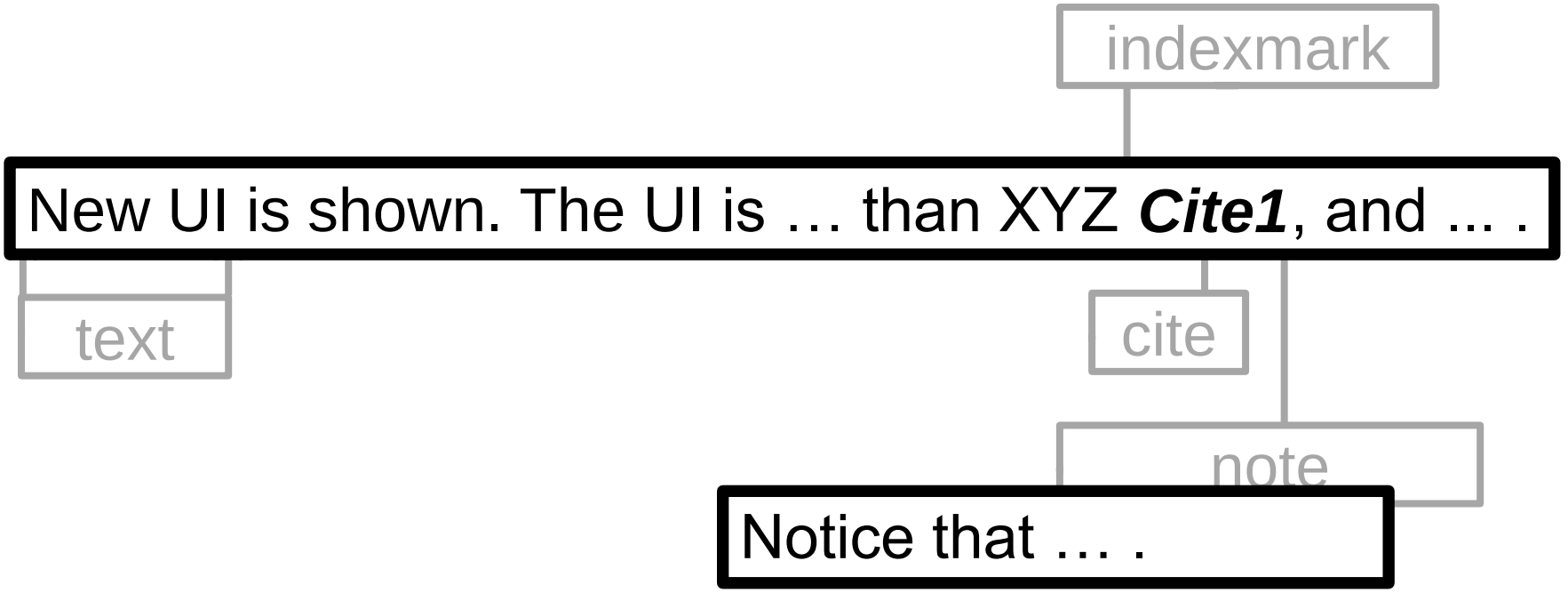
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



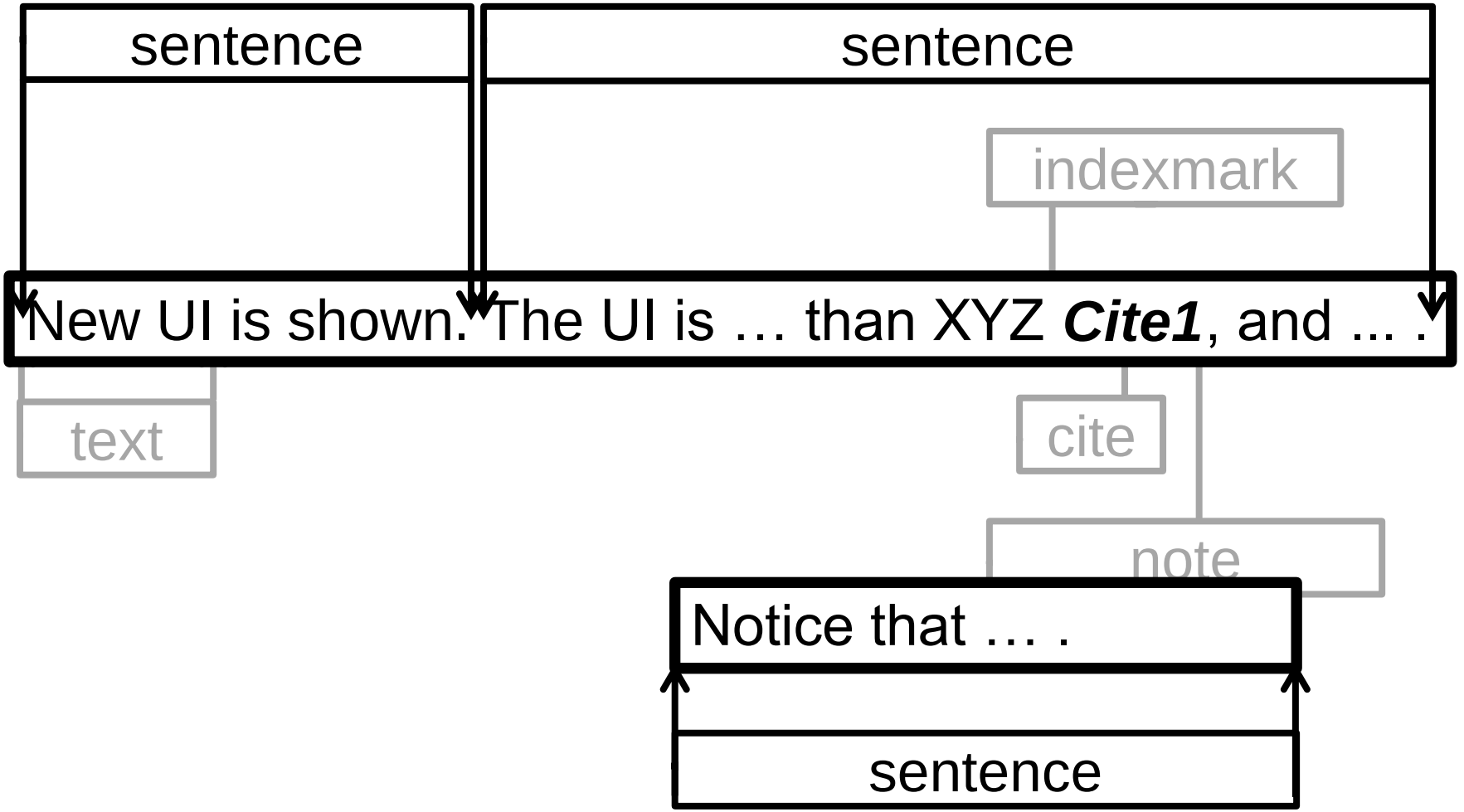
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



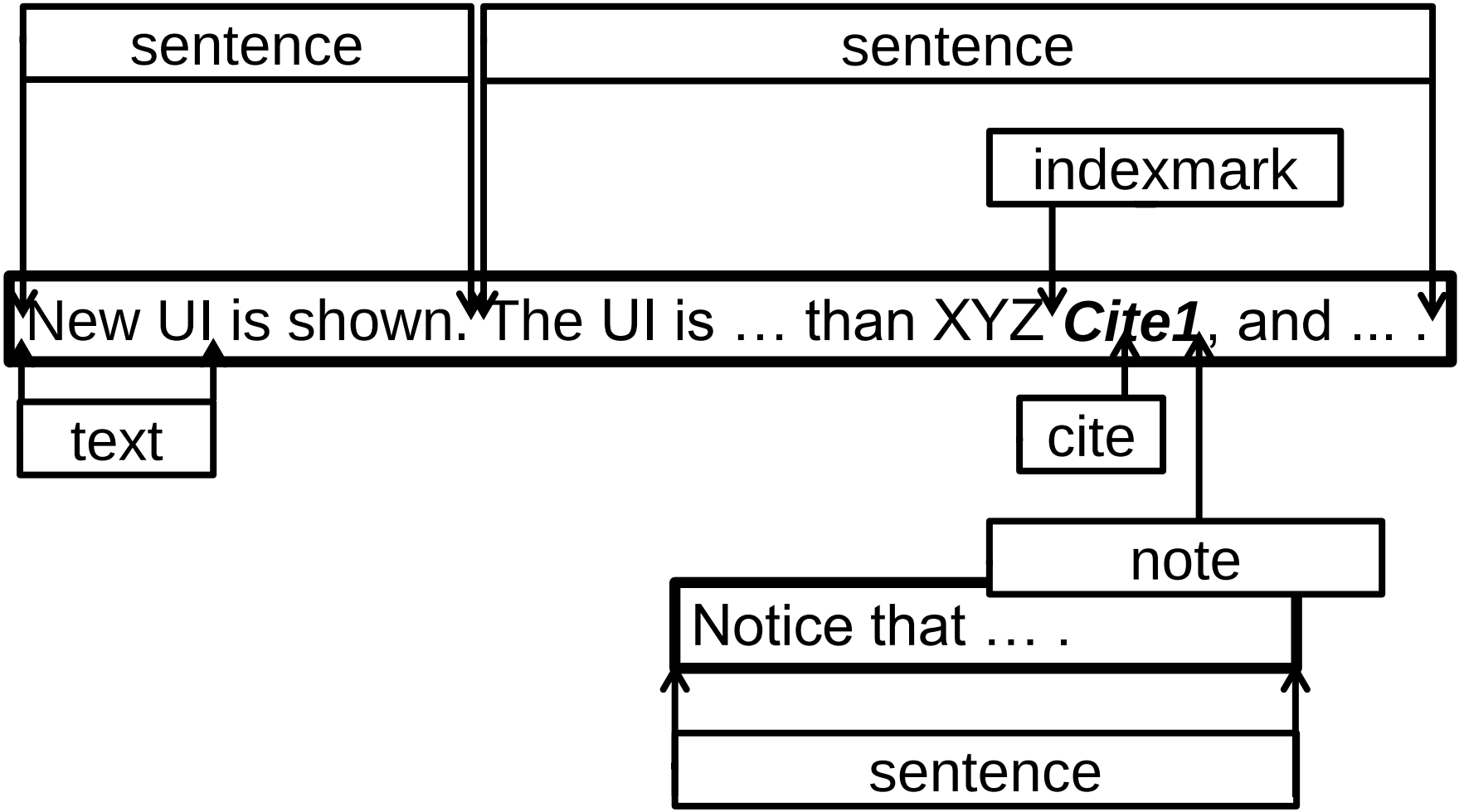
Strategies for Conversion (1/3): Conversion into Input for NLP Tools



Strategies for Conversion (2/3): Apply NLP Tools (Sentence Splitter)



Strategies for Conversion (3/3): Recover Original Tags

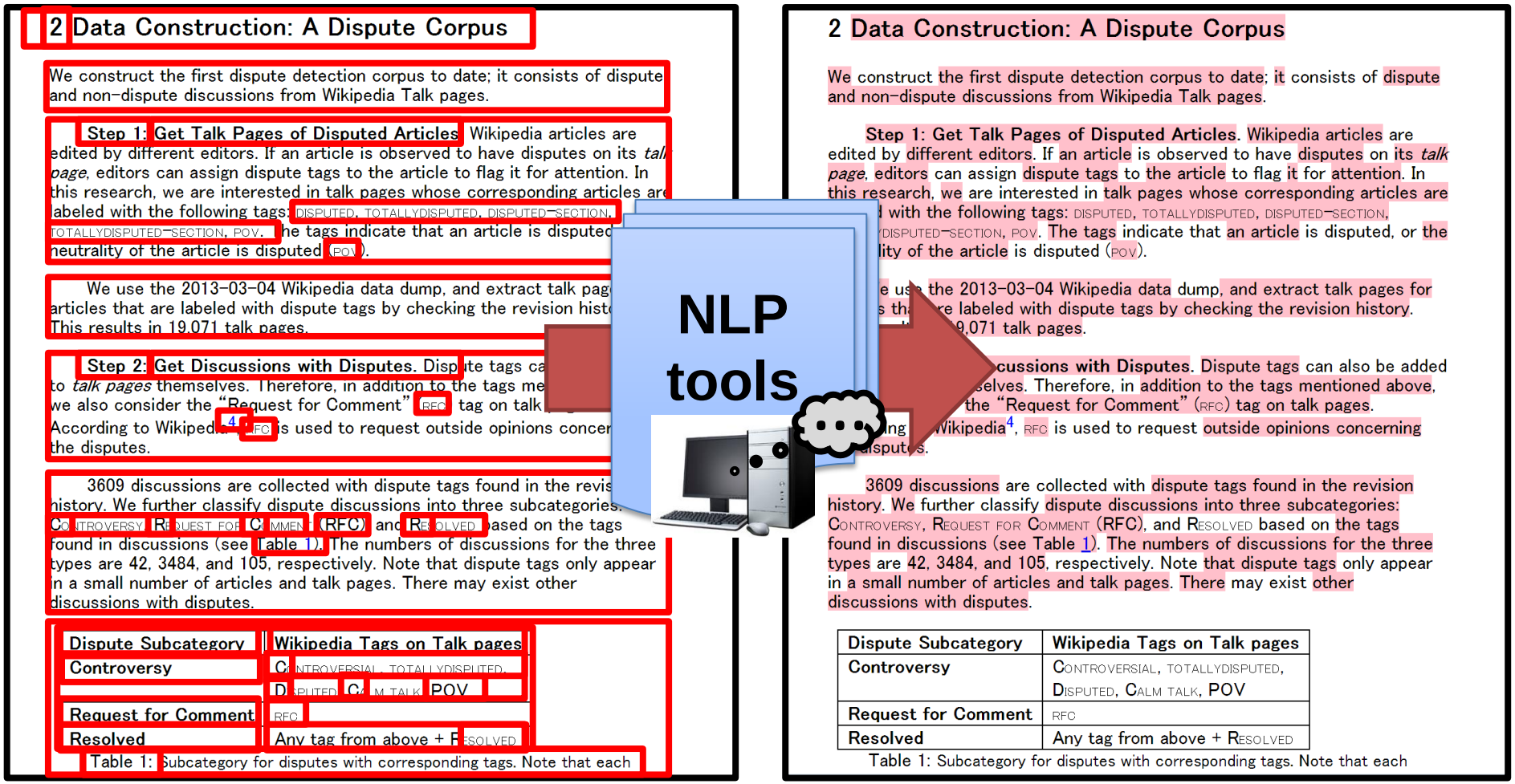


Strategies for Conversion (3/3): Recover Original Tags

<sentence><text>New UI</text> is shown.**</sentence>**
<sentence>The UI is more useful than XYZ<indexmark>…</indexmark> in <cite>[…]</cite><note>**<sentence>**
Notice that … . **</sentence>**</note>, and … .**</sentence>**

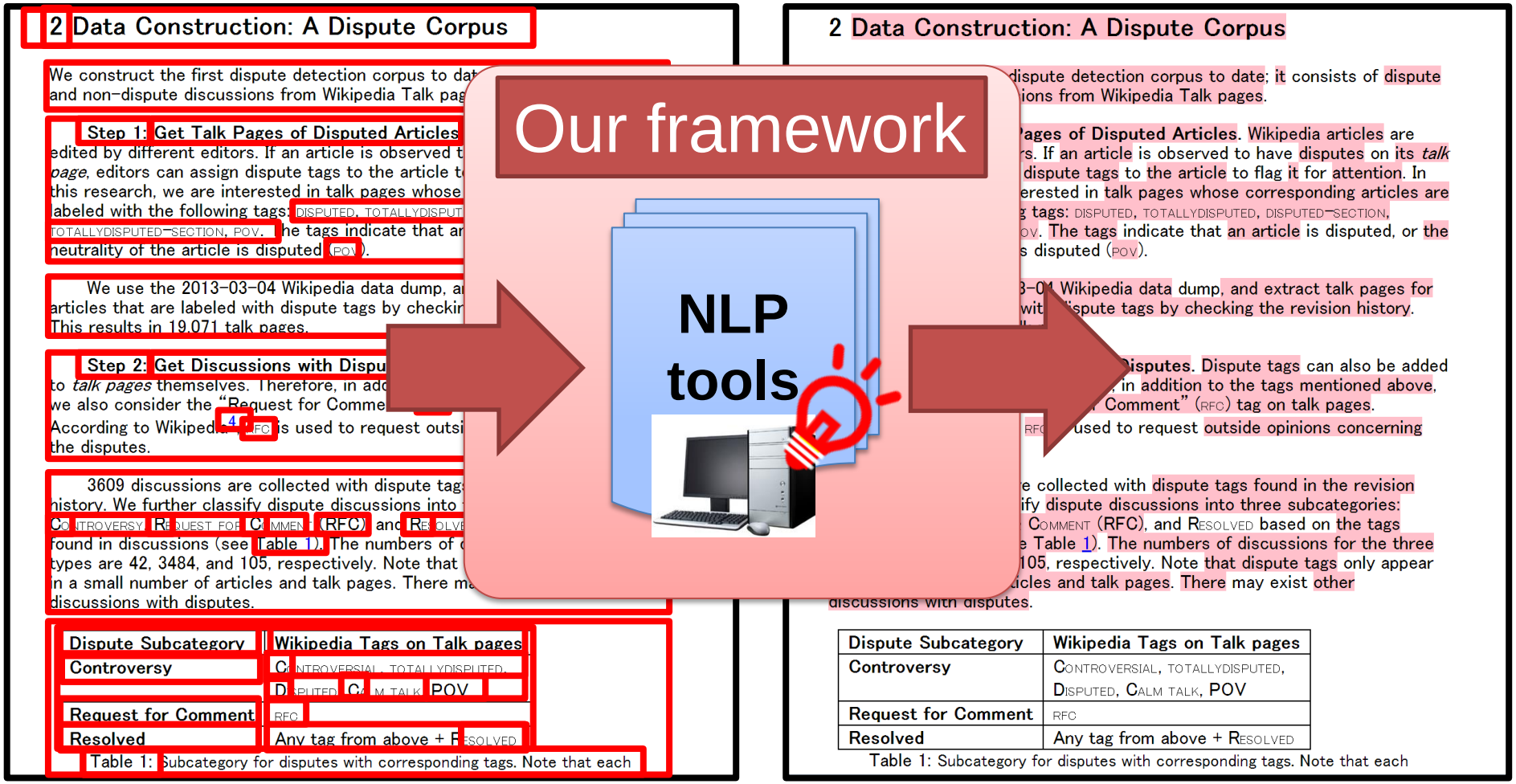
Goal of Research (Revisit)

- Analyze any (*XML*) doc. with NLP tools



Goal of Research (Revisit)

- Analyze any (*XML*) doc. with NLP tools



Summary of Tag Classification & Conversion Strategies

Types	Criteria	Conversion strategies
Independent	Enclose syntactically independent region	Separate (A) from (B) → NLP to (A) & (B) → recover (A) to (B)
Decoration	Set display style of region	Remove (A') from (B) → NLP → recover (A) to (B)
Object	Minimal object unit as text constituent	Replace (A) with (C) → NLP → recover (C) to (A)
Meta-info	Describe settings/ additional info.	Remove (A) from (B) → NLP → recover (A) to (B)

(A): tag&tagged region (A'): tag (B): original text
(C): dummy word

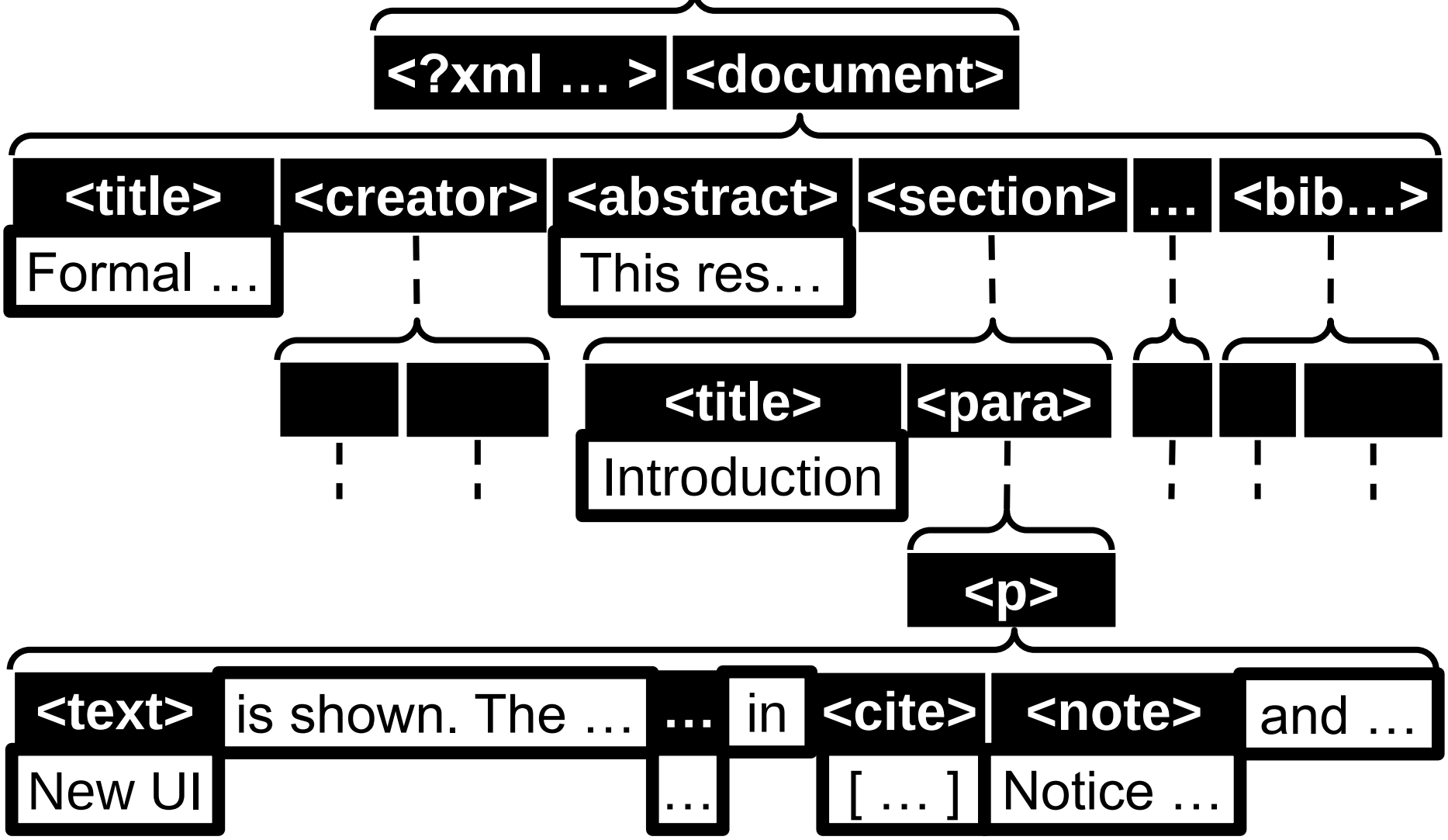
Efficiency of Tag Classification

XML document

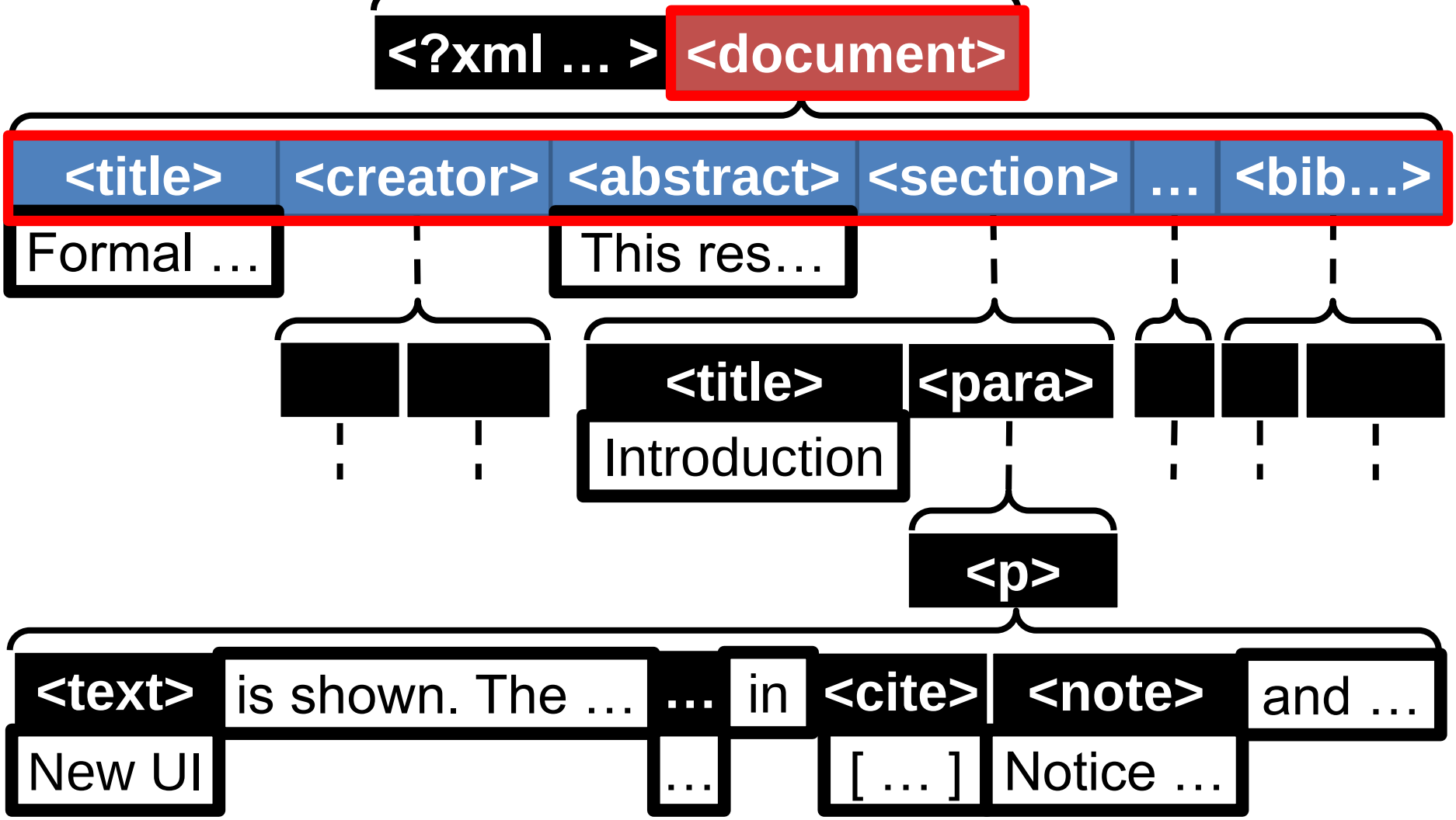
```
<document …>
   <title>Formal approaches … </title>
   <creator> … </creator>
   <abstract><p>This research … </p></abstract>
   <section><title>Introduction</title>
      <para><p><text>New UI</text> is shown. The
UI is more useful than XYZ<indexmark> …
</indexmark> in <cite>[ … ]</cite><note>Notice
that … </note> and … .</p></para>
   </section>
   <section> … </section>
   <bibliography> … </bibliography>
</document>
```

Classifying via naïve observation is inefficient

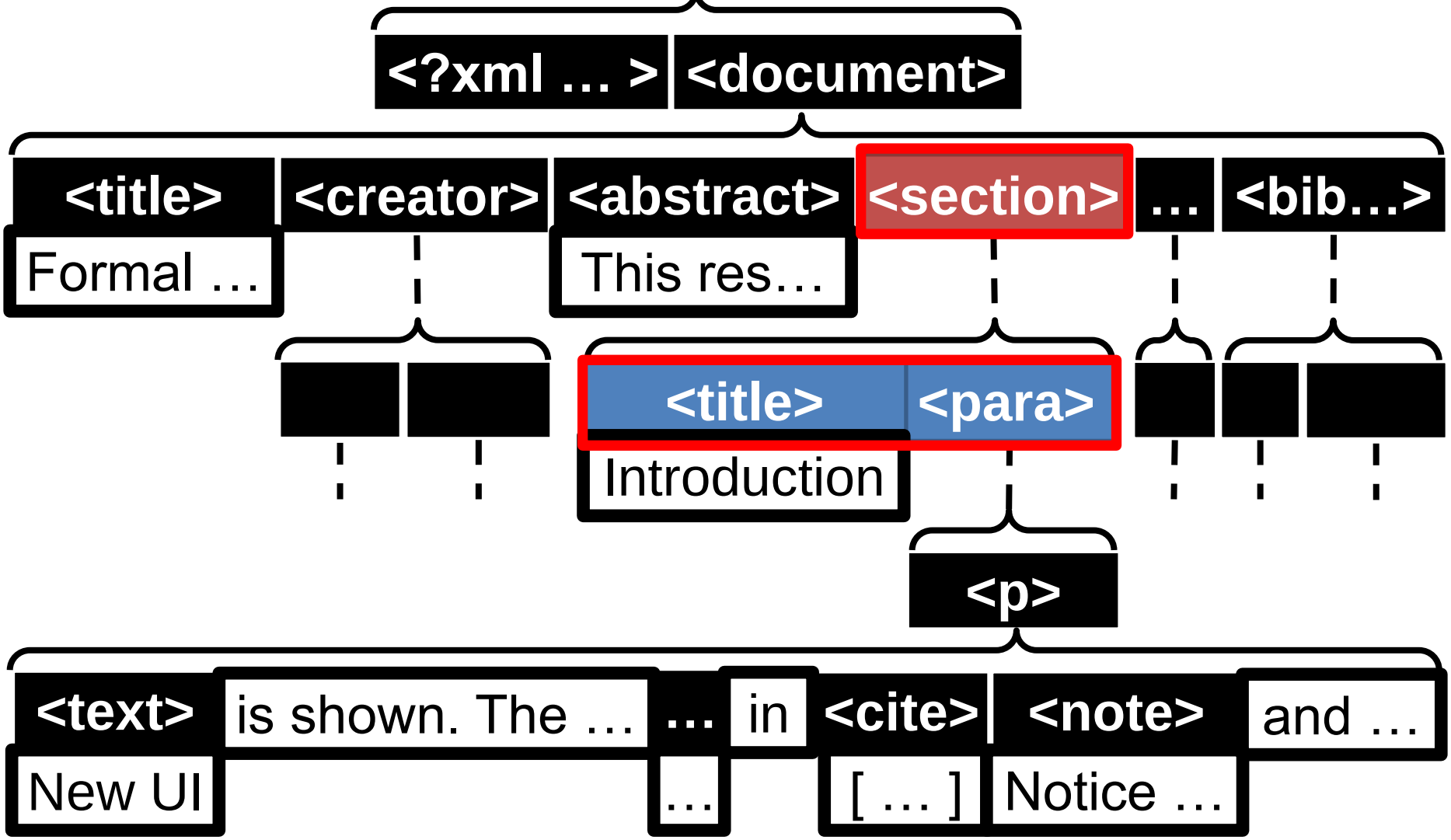
Multi-layered Structure of XML Document:



Multi-layered Structure of XML Document:

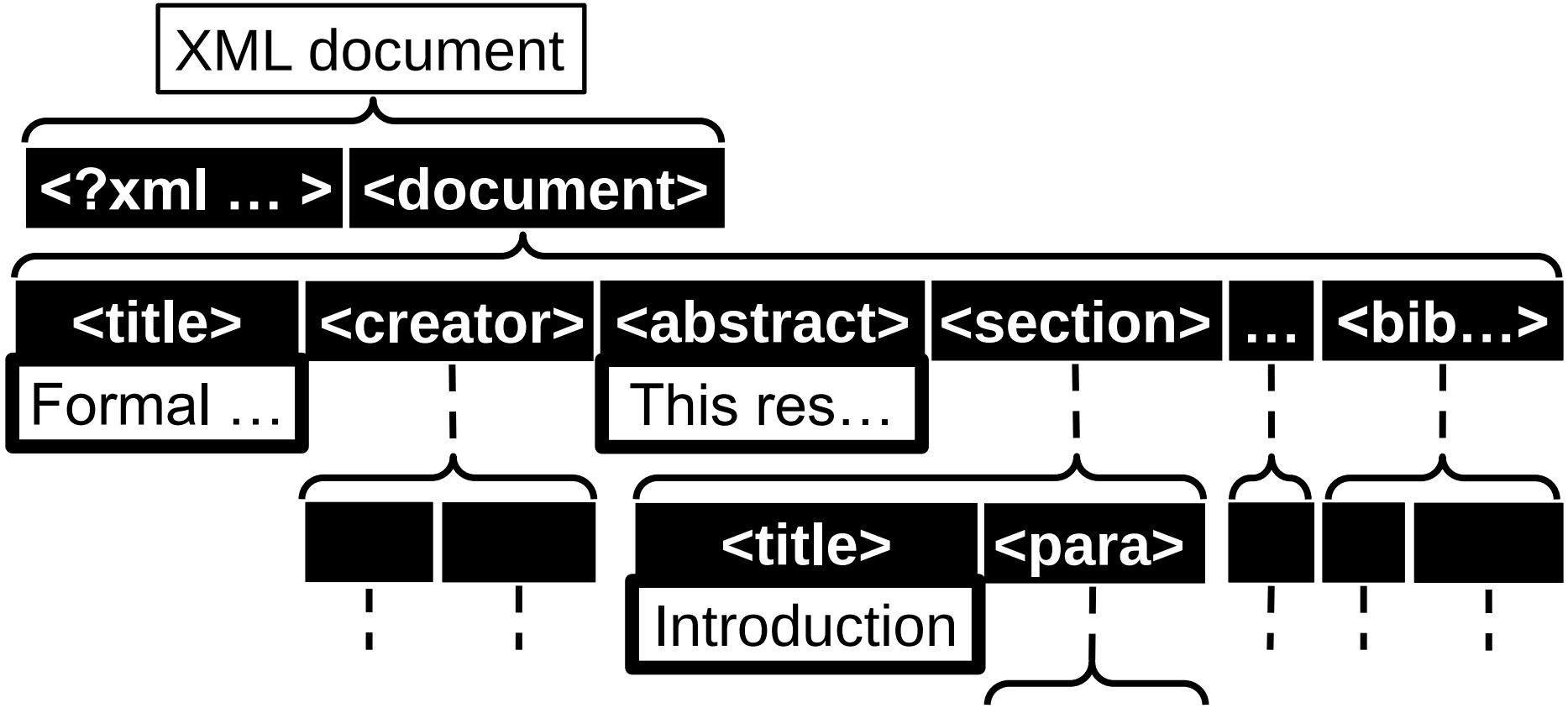


Multi-layered Structure of XML Document:



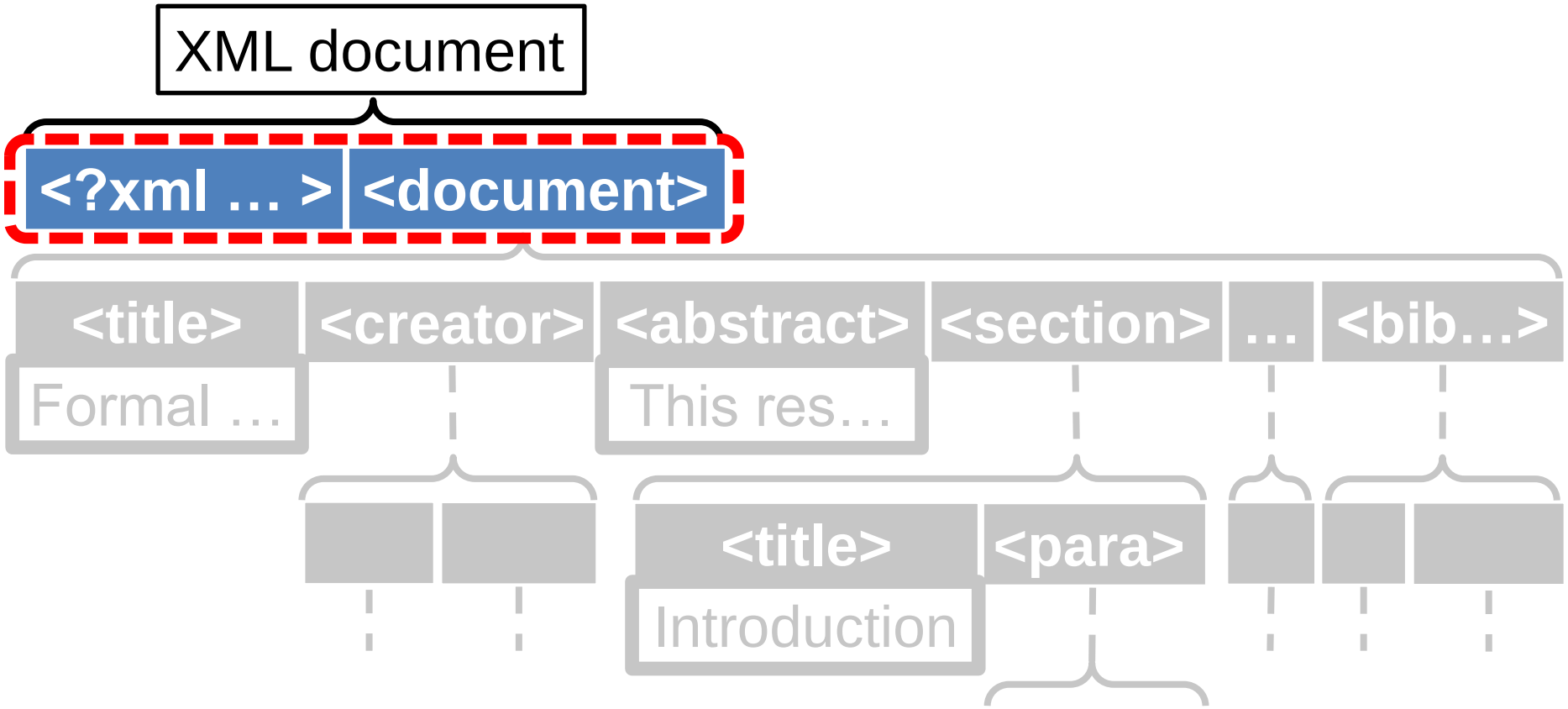
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



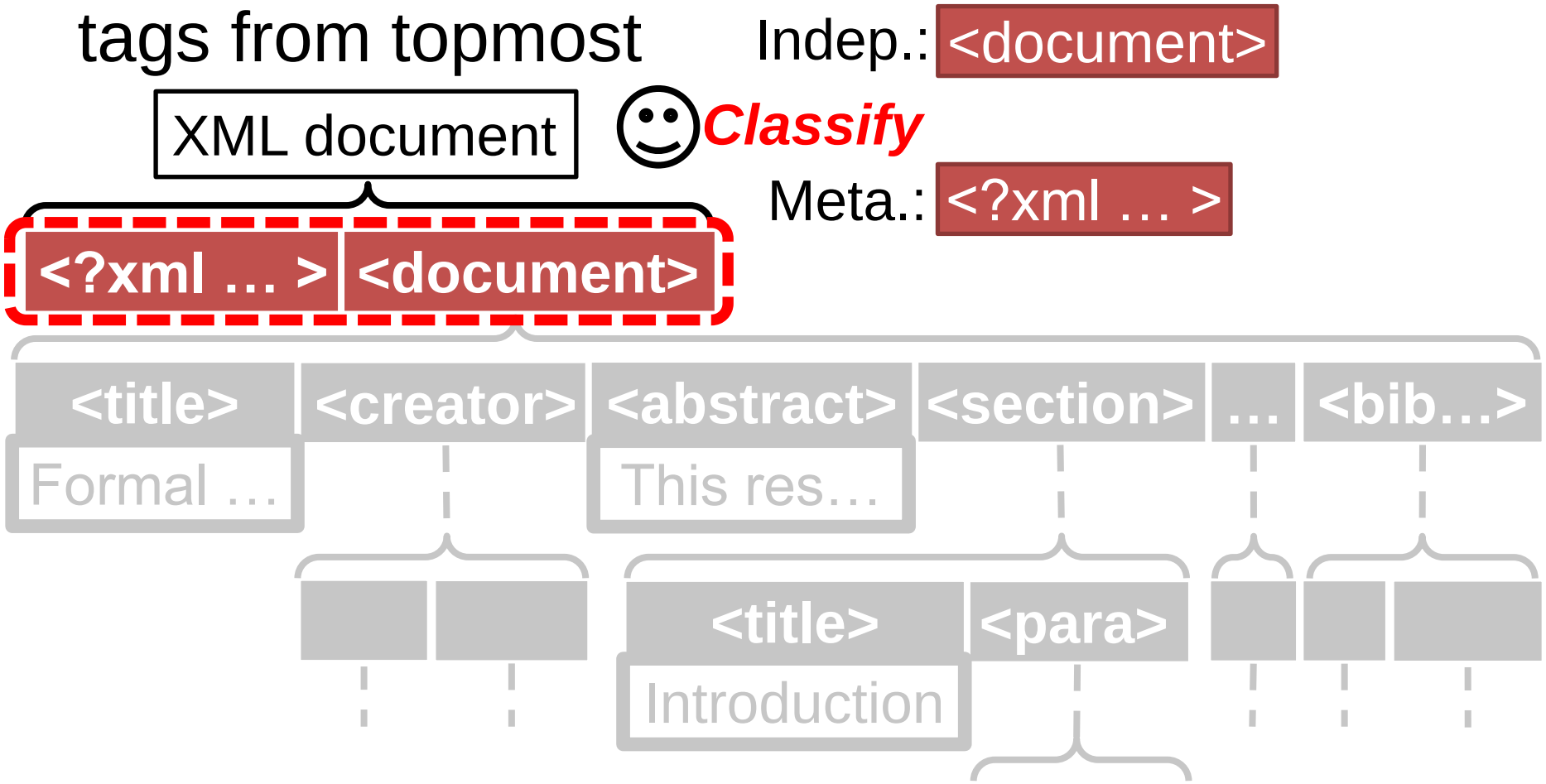
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



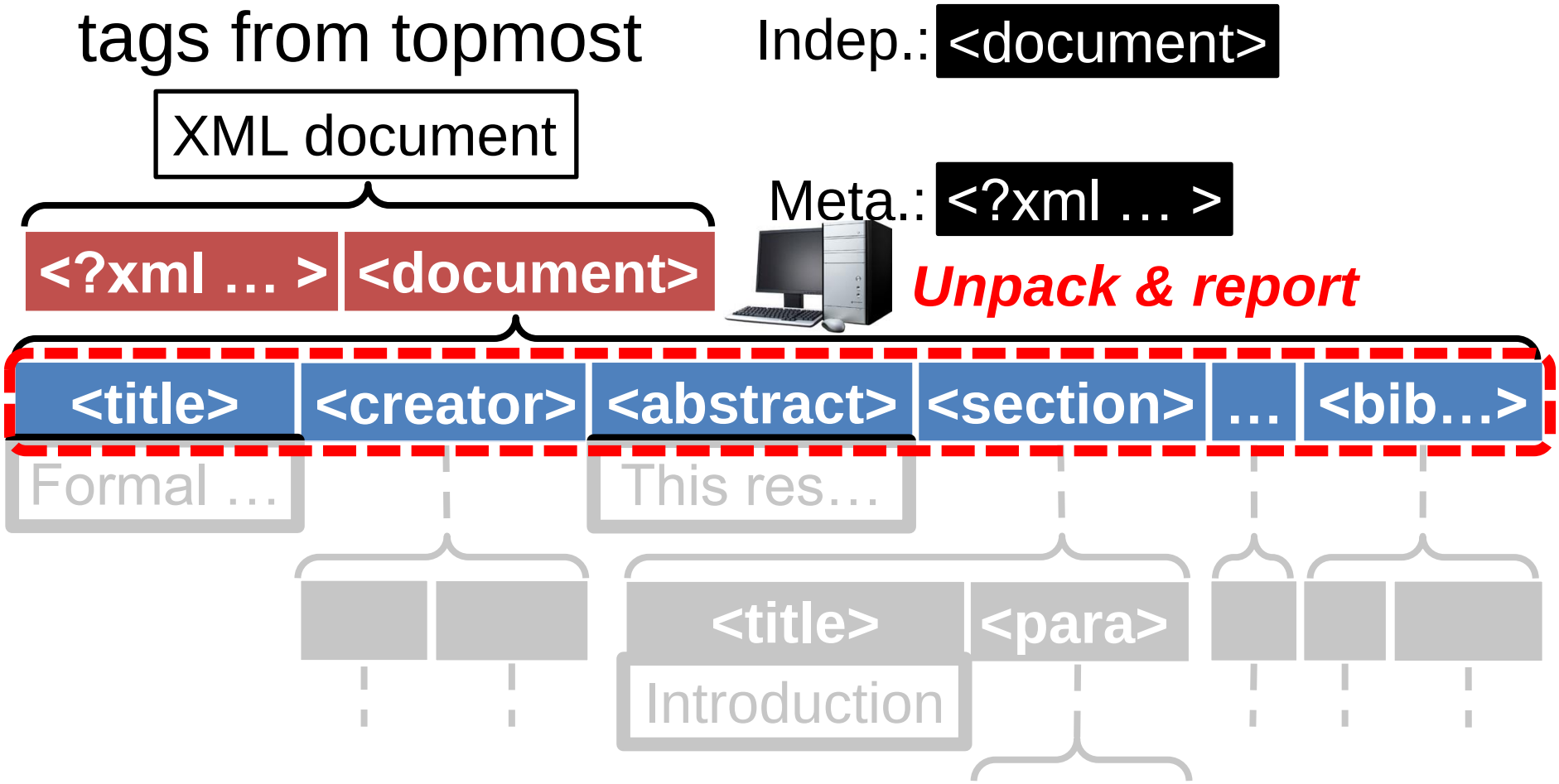
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



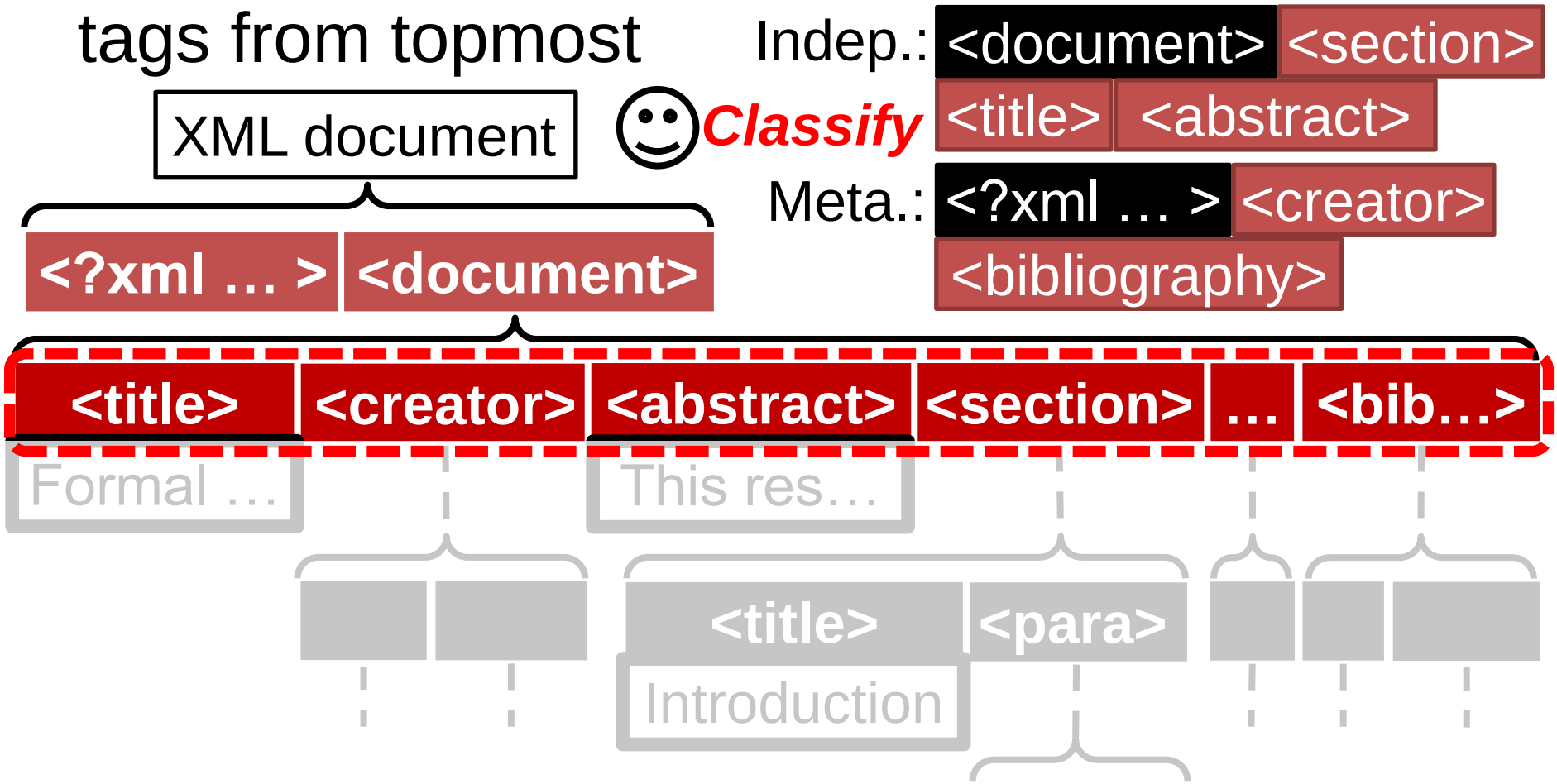
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



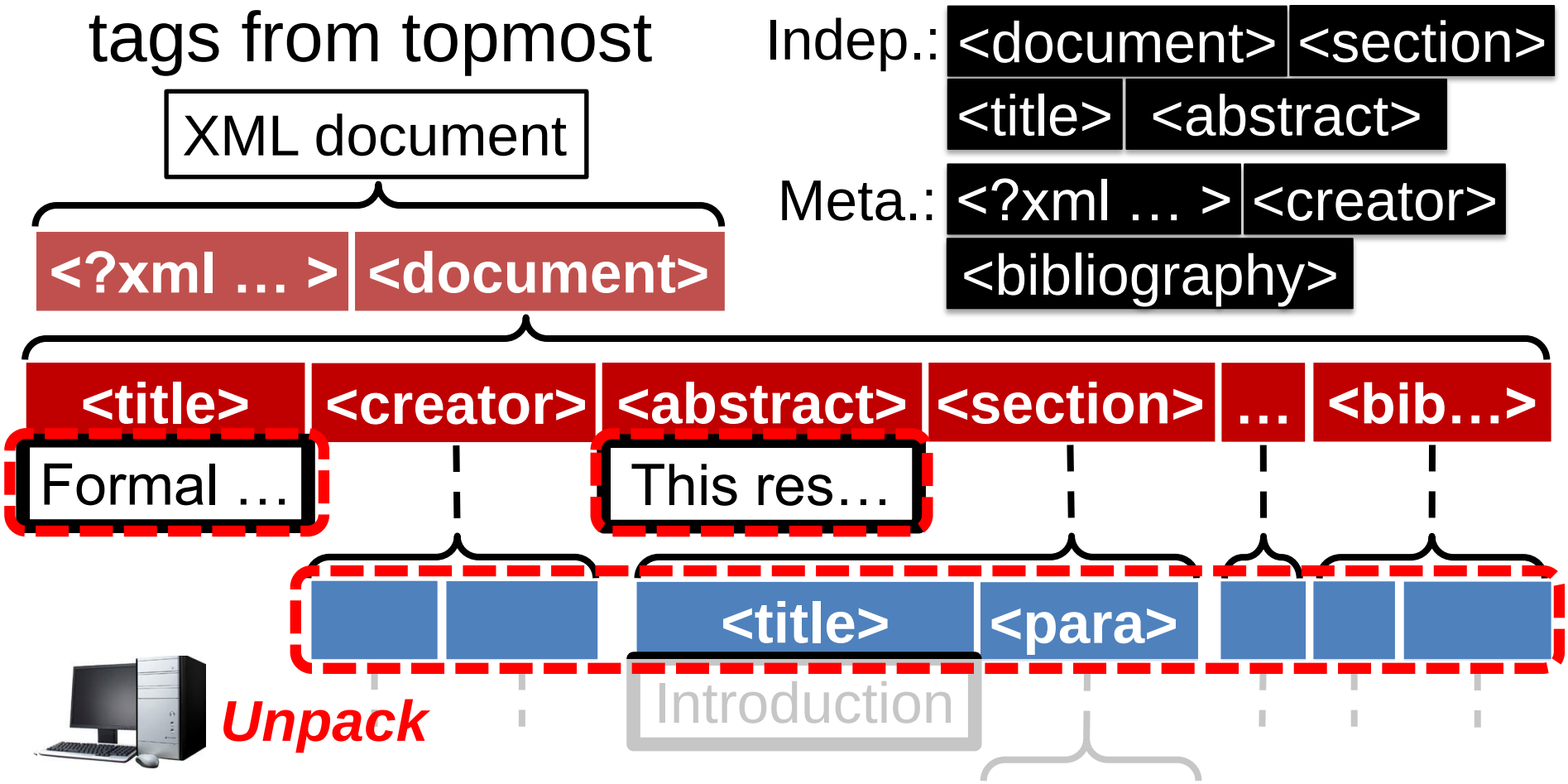
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



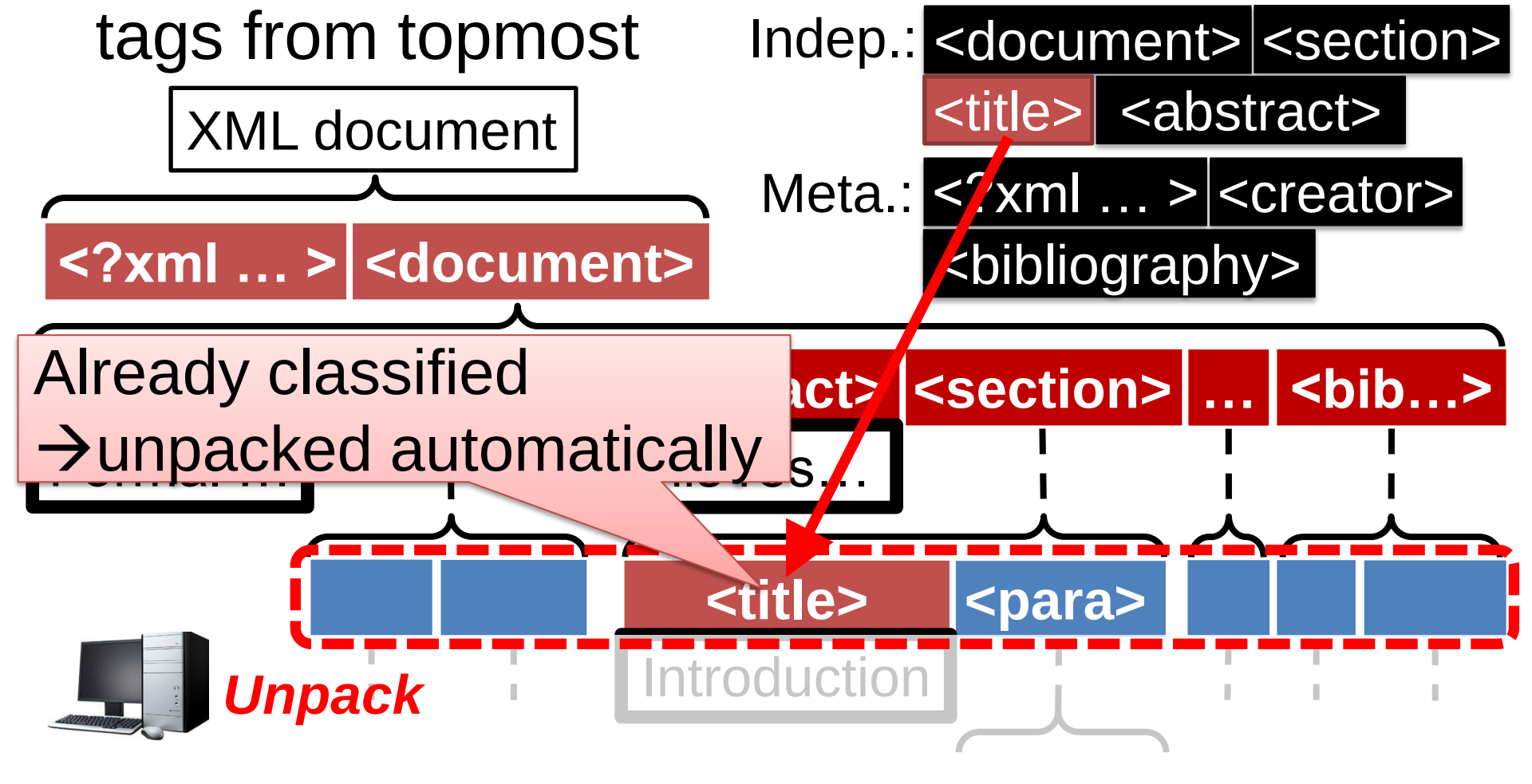
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



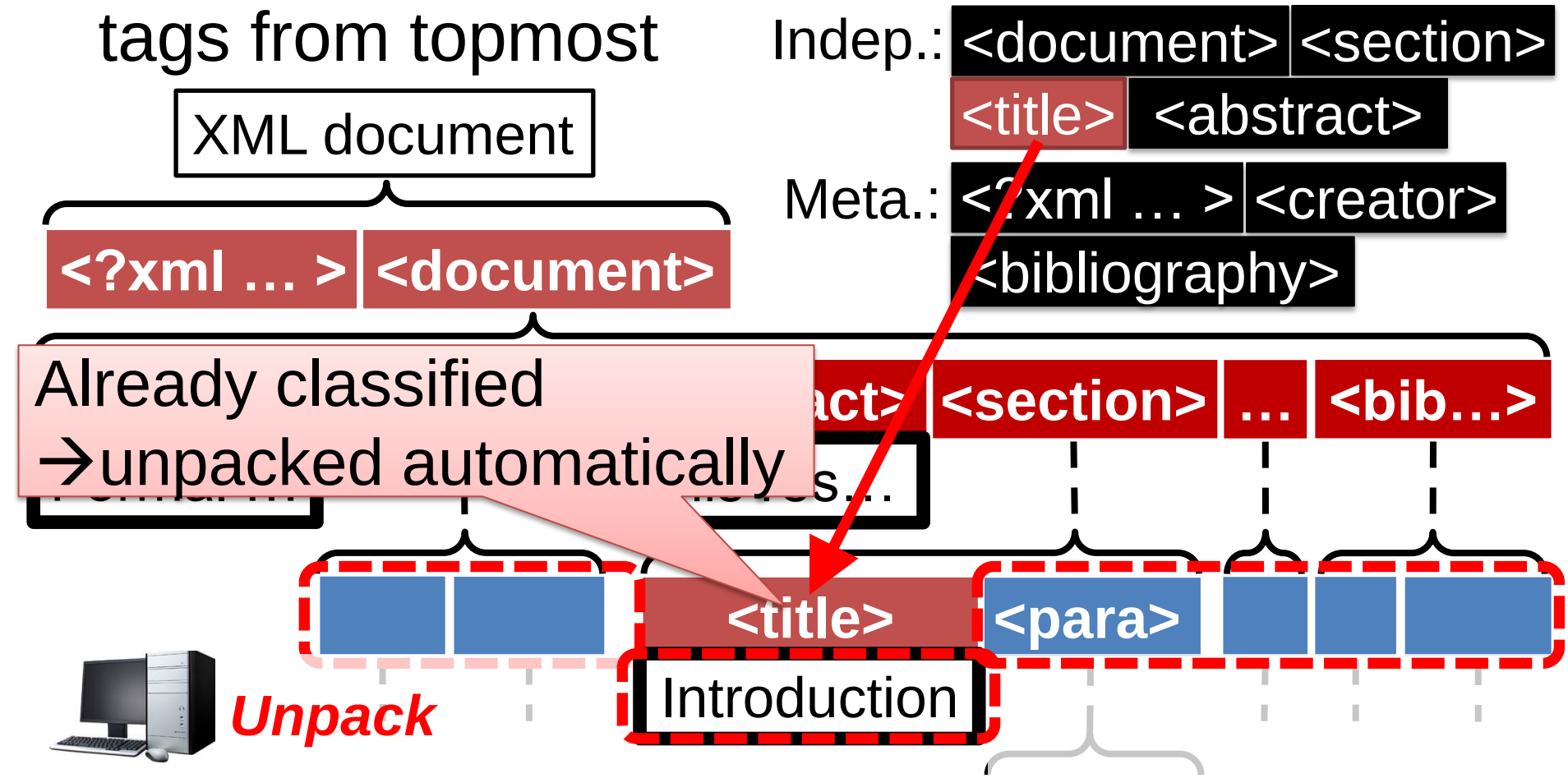
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



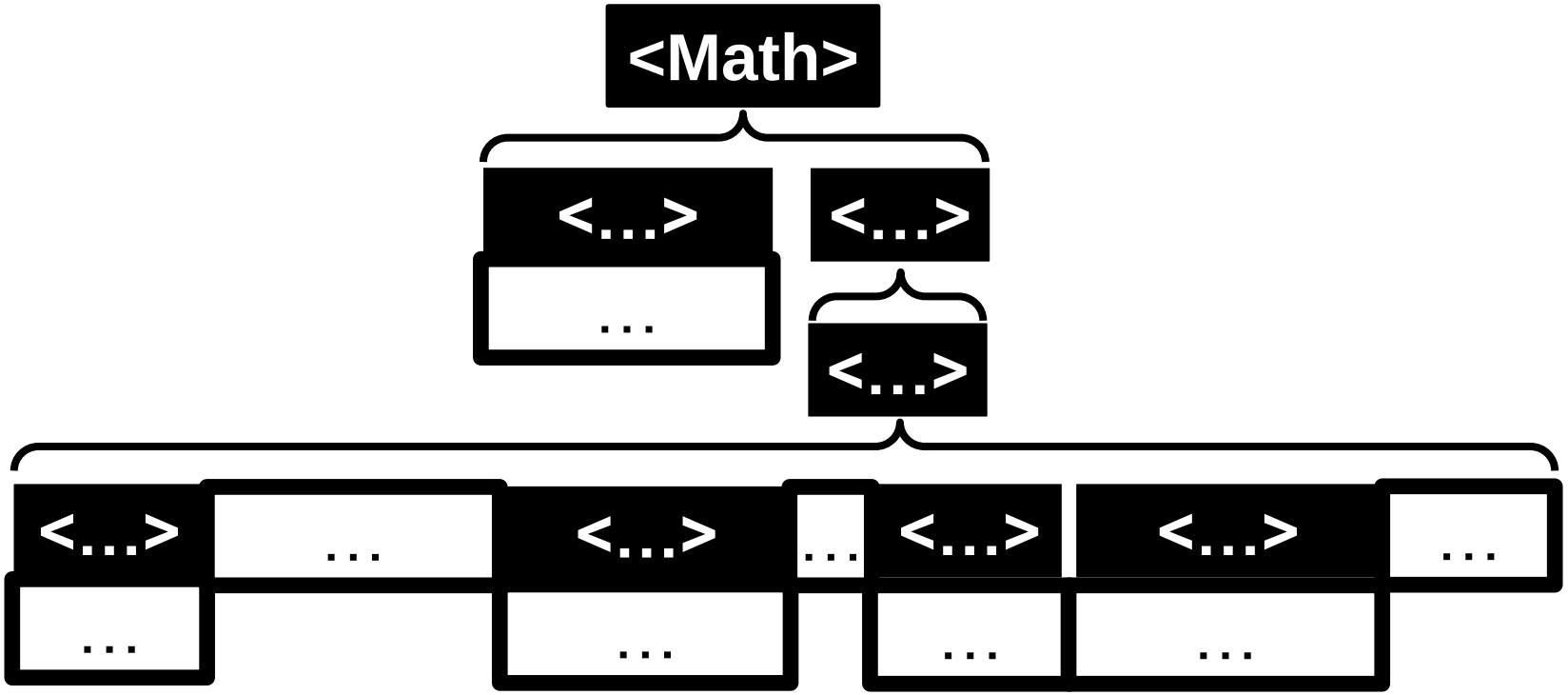
Procedure of Tag Classification

- Unpack regions enclosed by already-classified tags from topmost



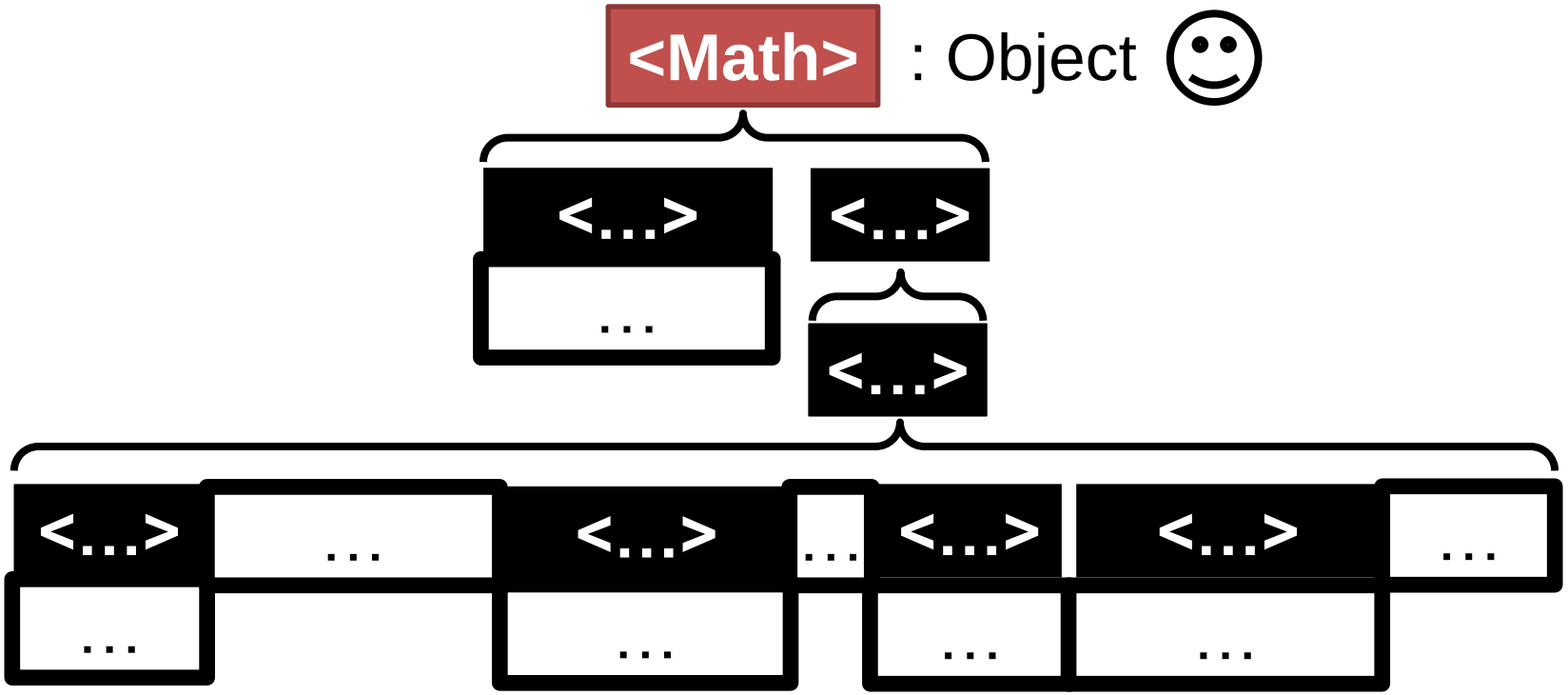
Efficiency Brought by Procedure

- Regions enclosed by Meta-info/Object tags are not unpacked → **User labor is saved**



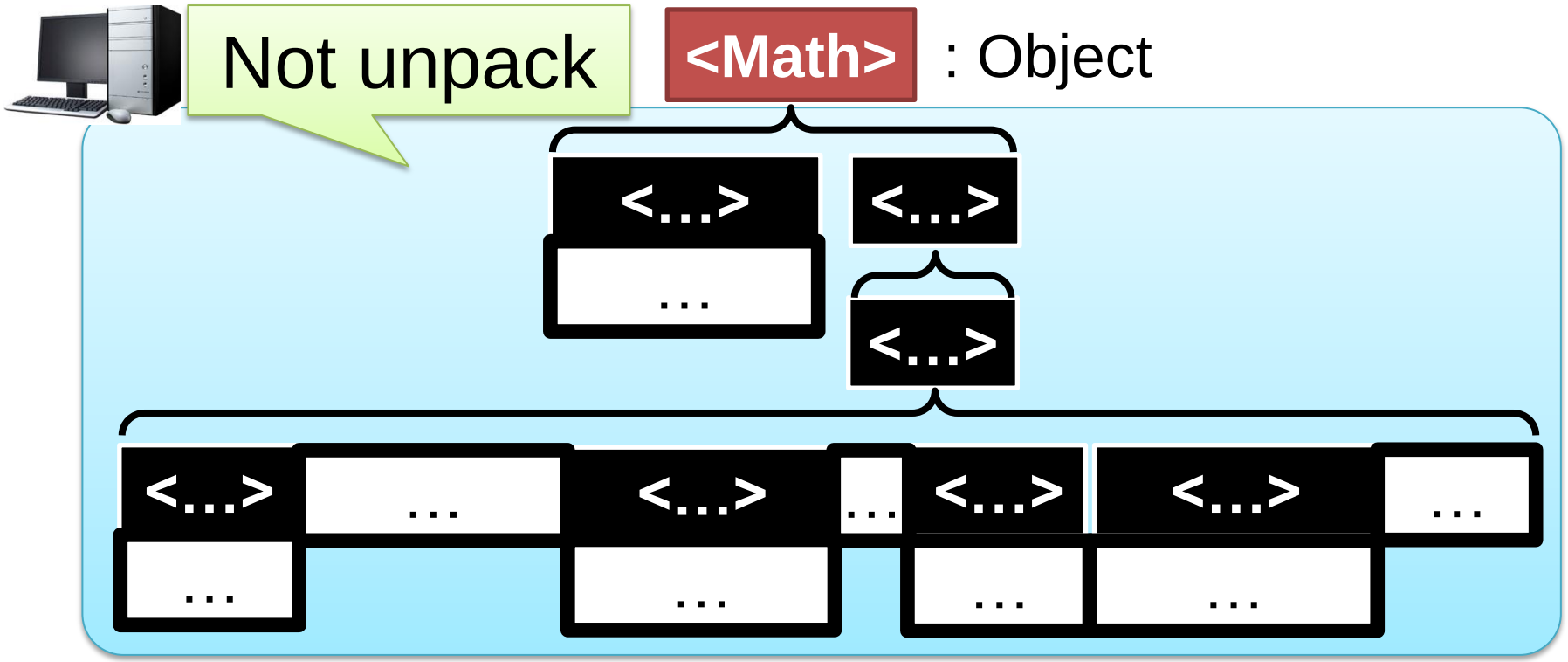
Efficiency Brought by Procedure

- Regions enclosed by Meta-info/Object tags are not unpacked → **User labor is saved**



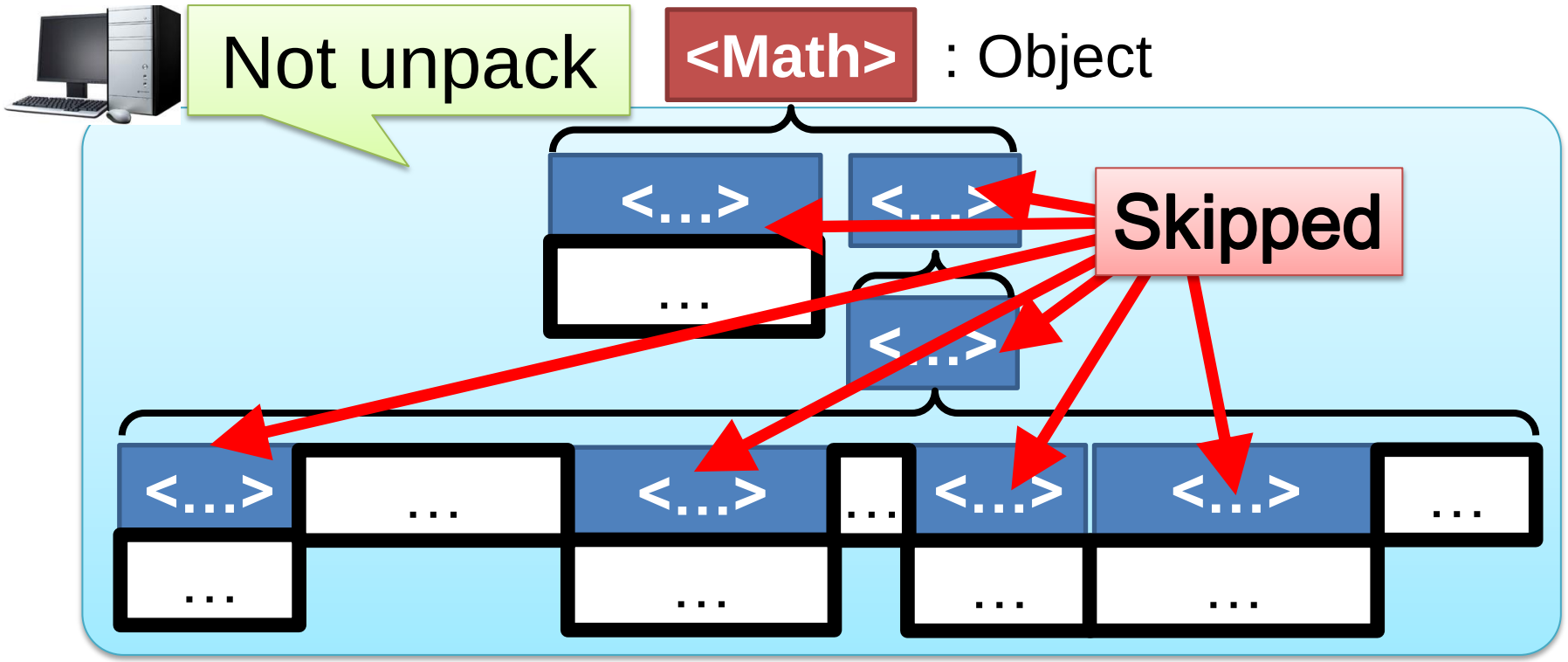
Efficiency Brought by Procedure

- Regions enclosed by Meta-info/Object tags are not unpacked → **User labor is saved**

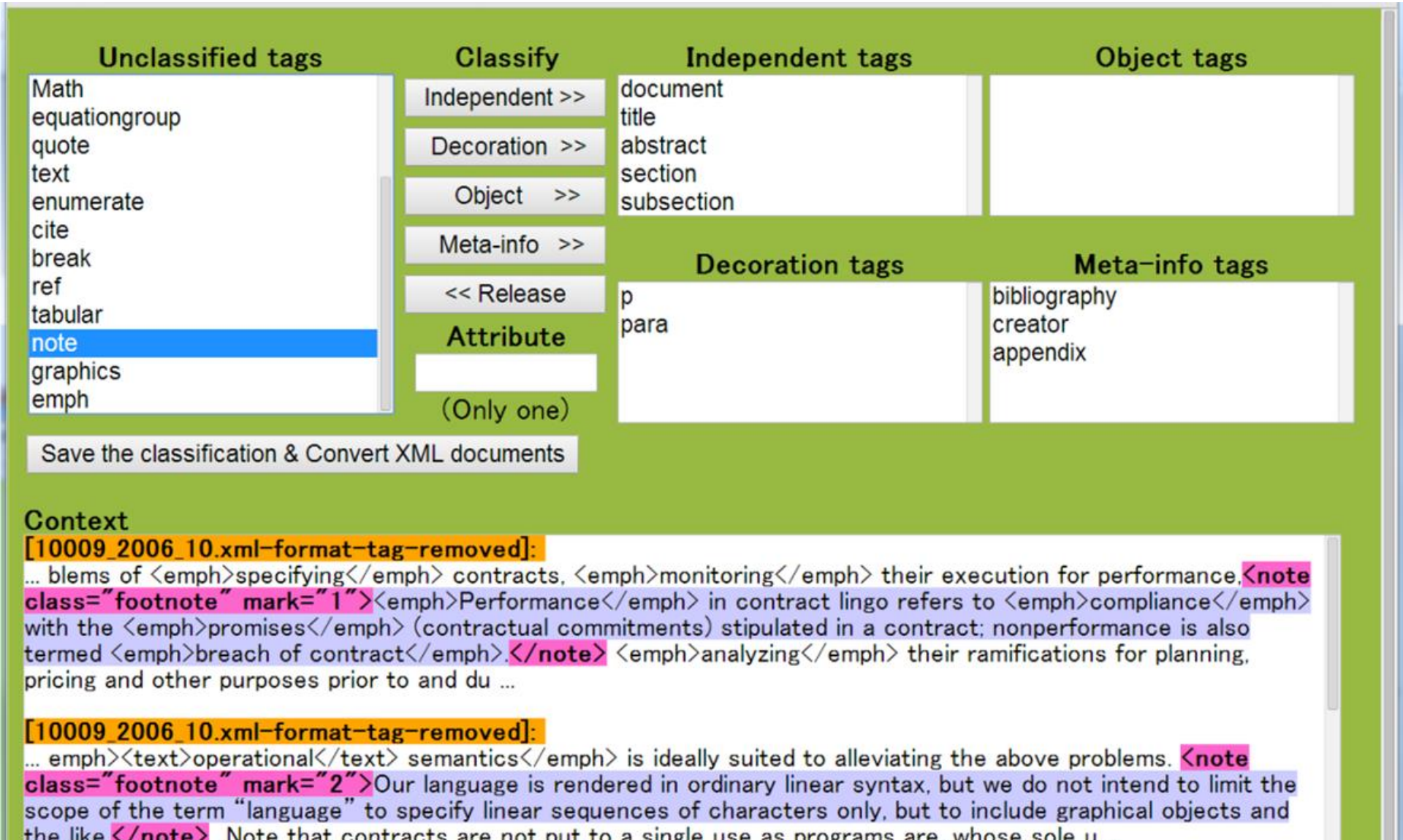


Efficiency Brought by Procedure

- Regions enclosed by Meta-info/Object tags are not unpacked → **User labor is saved**



GUI-based Tool for Classification & Conversion Procedure



Experiments

- Extract plain text sequences from several types of documents
  - Examine efficiency of tag classification
- Apply NLP tools to obtained sequences
  - Compare performance with naïvely obtained sequences
  - Discuss impact of proper extraction of plain text

Experimental Settings (1/3): Target Documents

Article type	Domain	Format	# used
PubMed Central (PMC)*[1]	Scientific paper	XML	1,000
arXiv.org*[2]		XHTML	300
ACL 2014*[3]		XHTML	67
Wikipedia*[4] entries	Web page	HTML†	300

(† XML-like: generated via intermediate XML files)

---

*[1] http://www.ncbi.nlm.nih.gov/pmc/tools/ftp/ *[2] http://arxiv.org/
*[3] http://anthology.aclweb.org/ *[4] http://www.wikipedia.org/

Experimental Settings (2/3): NLP Tools (Two Types of Parsers)

- Enju parser *1 (+ Genia Sentence Splitter*2)
  - Deep syntactic/semantic analysis
  - Memory overflow of search space = failure
- Stanford parser *3
  - Phrase structure & dependency analysis
  - Failure in long sentences terminated whole process → limit sentence length to 50 words
  - (>50 word sentences were skipped → failure)

---

*1 Enju (Ninomiya et al., 2007)

*2 GeniaSS (http://www.nactem.ac.uk/y-matsu/geniass/)

*3 Stanford parser (de Marneffe et al., 2006)

Experimental Settings (3/3): Comparison and Evaluation

- Three types of tag treatment compared:
  - **Remove**: simply remove all tags
  - **O/M**: Object/Meta-info → our framework
    Decoration / Independent → remove
  - **I/D/O/M**: process all tags using our framework
- Performance measured by:
  - # sentences detected by parser
  - Total parsing time
  - # (% of) sentences which could not be parsed

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	D	O	M	Total	
PMC (1,000)	**1,357 k (421)**	32k (12)	62k (9)	48k (9)	34k (56)	**177k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	6k (15*)	47k (12*)	60k (8*)	8k (17*)	**121k (52*)**	**4 k**
ACL (67)	**131 k (66)**	3k (24*)	14 k (29*)	5k (15*)	2k (19*)	**24k (87*)**	**2 k**
Wiki. (300)	**224 k (60)**	3k (12*)	11 k (8*)	1k (28*)	11k (67*)	**28k (115*)**	**2 k**

(**I**: Independent, **D**: Decoration, **O**: Object, **M**: Meta-info)

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	D	O	M	Total	
PMC (1,000)	**1,3[illegible] k (421)**	[illegible] (12)	[illegible] (3)	[illegible] (3)	[illegible] (50)	**[illegible] k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	6k (15*)	47k (12*)	60k (8*)	8k (17*)	**121k (52*)**	**4 k**
ACL (67)	**131 k (66)**	3k (24*)	14 k (29*)	5k (15*)	2k (19*)	**24k (87*)**	**2 k**
Wiki. (300)	**224 k (60)**	3k (12*)	11 k (8*)	1k (28*)	11k (67*)	**28k (115*)**	**2 k**

Classify only 20% of total tag types

(**I**: Independent, **D**: Decoration, **O**: Object, **M**: Meta-info)

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	**D**	**O**	**M**	**Total**	
PMC (1,000)	**1,3[illegible] k (421)**	[illegible] (12)	[illegible] (3)	[illegible] (3)	[illegible] (50)	**177k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	6k (15*)	47k (12*)	60k (8*)	8k (17*)	**121k (52*)**	**4 k**
ACL (67)							
Wiki. (300)							

(**I**: Independent, **D**: Decoration, **O**: Object, **M**: Meta-info)

Classify only 20% of total tag types

* XHTML, HTML:
Tag names represent more abstract information
→ Use combination of tag name & its selected attributes as a single tag

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	D	O	M	Total	
PMC (1,000)	**1,3[illegible] k (421)**	[illegible] (12)	[illegible] (9)	[illegible] (9)	[illegible] (56)	**177k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	[illegible] (15*)	[illegible] (12*)	[illegible] (8*)	[illegible] (17*)	**121k (52*)**	**4 k**
ACL (67)	**131 k**	[illegible]	[illegible]	[illegible]	[illegible]	**24k**	**2 k**
Wiki. (300)	**224 k (60)**	[illegible] (12*)	[illegible] (8*)	[illegible] (28*)	[illegible] (67*)	**28k (115*)**	**2 k**

Classify only 20% of total tag types

Focus on less than 20% of total occurrences

(**I**: Independent, **D**: Decoration, **O**: Object, **M**: Meta-info)

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	P	O	M	Total	
PMC (1,000)	**1,357 k (421)**	32k (12)	62k (9)	43k (9)	34k (56)	**171k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	9k (15*)	47k (12*)	33k (8*)	32k (17*)	**121k (52*)**	**4 k**
ACL (67)	**131 k**	3k	14 k	5k	2k	**24k**	**2 k**
Wiki. (300)	**224 k**	3k	11 k	1k	11k	**28k**	**2 k**

(**I**: Indepe -info)

Classify only 20% of total tag types

Focus on less than 20% of total occurrences

Tags within regions enclosed by Object/ Meta-info tags were not considered

Classified Tags and Obtained Sequences for Each Type of Article

Article type (#)	# tags (# types)	# classified tags (# types)					# seq.
		I	D	O	M	Total	
PMC (1,000)	**1,357 k (421)**	32k (12)	62k (9)	48k (9)	34k (56)	**177k (85)**	**26 k**
arXiv (300)	**1,969 k (210)**	6k (15*)	47k (12*)	60k (8*)	8k (17*)	**121k (52*)**	**4 k**
ACL	**131 k**	3k	14 k	5k	2k	**24k**	**2 k**
							2 k

Observe sequences for randomly-selected articles

→ They consisted of *valid sentences*:

- Could be directly input into NLP tools
- Thoroughly covered content of original articles

nfo)

Impact of Tag Treatment on Performance of Enju Parser

Art. (#)	Treatment	# sentences	Time (s)	# failure (%)
PMC (1,000)	**Remove**	159,327	209,783	4,721 (2.96)
	O/M	112,285	135,752	810 (0.72)
	I/D/O/M	126,215	132,250	699 (0.55)
arXiv (300)	**Remove**	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	411 (1.00)
	I/D/O/M	43,208	87,952	348 (0.81)
ACL (67)	**Remove**	19,571	15,142	115 (0.59)
	O/M	9,819	9,481	63 (0.64)
	I/D/O/M	11,136	8,482	39 (0.35)
Wiki. (300)	**Remove**	10,561	14,704	1,161(10.99)
	O/M	5,026	6,743	67 (1.33)
	I/D/O/M	6,893	6,058	61 (0.88)

Impact of Tag Treatment on Performance of Stanford Parser

Art. (#)	Treatment	# sentences	Time (s)	# failure (%)
PMC (1,000)	**Remove**	170,999	58,865	18,621(10.89)
	O/M	126,176	50,741	11,881 (9.42)
	I/D/O/M	139,805	63,295	11,338 (8.11)
arXiv (300)	**Remove**	75,672	27,970	10,590(13.99)
	O/M	48,666	24,630	5,457(11.21)
	I/D/O/M	50,504	26,360	5,345(10.58)
ACL (67)	**Remove**	17,166	5,047	1,095 (6.38)
	O/M	11,182	4,157	616 (5.51)
	I/D/O/M	12,402	4,871	587 (4.73)
Wiki. (300)	**Remove**	14,883	3,114	1,651(11.09)
	O/M	6,173	2,248	282 (4.57)
	I/D/O/M	8,049	2,451	258 (3.21)

Impact of Tag Treatment on Performance of Enju Parser

Art. (#)	Treatment	# sentences	Time (s)	# failure (%)
PMC (1,000)	Remove	159,327	209,783	4,721 (2.96)
	O/M	112,285	135,752	810 (0.72)
	I/D/O/M	126,215	132,250	699 (0.55)
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	411 (1.00)
	I/D/O/M	43,208	87,952	348 (0.81)
ACL (67)	Remove	19,571	15,142	115 (0.59)
	O/M	9,819	9,481	63 (0.64)
	I/D/O/M	11,136	8,482	39 (0.35)
Wiki. (300)	Remove	10,561	14,704	1,161(10.99)
	O/M	5,026	6,743	67 (1.33)
	I/D/O/M	6,893	6,058	61 (0.88)

Impact of Tag Treatment on Performance of

Parsing failure: ~10%↓
→ Much higher coverage

Art. (#)	Treatment	# sentenc		
PMC (1,000)	Remove	159,327	209,783	4,721 (2.96)
	O/M	112,285	135,752	**810 (0.72)**
	I/D/O/M	126,215	132,250	699 (0.55)
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	**411 (1.00)**
	I/D/O/M	43,208	87,952	348 (0.81)
ACL (67)	Remove	19,571	15,142	115 **(0.59)**
	O/M	9,819	9,481	**63 (0.64)**
	I/D/O/M	11,136	8,482	39 (0.35)
Wiki. (300)	Remove	10,561	14,704	1,161(10.99)
	O/M	5,026	6,743	**67 (1.33)**
	I/D/O/M	6,893	6,058	61 (0.88)

Impact of Tag Treatment on

Parsing time: 18 ~ 54%↓
→ Drastic speed-up

Parsing failure: ~10%↓
→ Much higher coverage

Ar				
PMC (1,000)	Remove	159,327	209,783	4,721 (2.96)
	O/M	112,285	**135,752**	**810 (0.72)**
	I/D/O/M	126,215	132,250	699 (0.55)
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	**89,200**	**411 (1.00)**
	I/D/O/M	43,208	87,952	348 (0.81)
ACL (67)	Remove	19,571	15,142	115 **(0.59)**
	O/M	9,819	**9,481**	**63 (0.64)**
	I/D/O/M	11,136	8,482	39 (0.35)
Wiki. (300)	Remove	10,561	14,704	1,161(10.99)
	O/M	5,026	**6,743**	**67 (1.33)**
	I/D/O/M	6,893	6,058	61 (0.88)

Impact of Tag Treatment on Performance of Stanford Parser

Art. (#)	Treatment	# sentences	Time (s)	# failure (%)
PMC (1,000)	Remove	170,999	58,865	18,621(10.89)
	O/M	126,176	50,741	11,881 (9.42)
	I/D/O/M	139,805	63,295	11,338 (8.11)
arXiv (300)	Remove	75,672	27,970	10,590(13.99)
	O/M	48,666	24,630	5,457(11.21)
	I/D/O/M	50,504	26,360	5,345(10.58)
ACL (67)	Remove	17,166	5,047	1,095 (6.38)
	O/M	11,182	4,157	616 (5.51)
	I/D/O/M	12,402	4,871	587 (4.73)
Wiki. (300)	Remove	14,883	3,114	1,651(11.09)
	O/M	6,173	2,248	282 (4.57)
	I/D/O/M	8,049	2,451	258 (3.21)

Impact of Tag Treatment on

Parsing time: 12 ~ 28%↓
→ Drastic speed-up

Parsing failure: 1 ~ 7%↓
→ Much higher coverage

Ar				
PMC (1,000)	Remove	170,099	58,865	18,621(10.89)
	O/M	126,176	**50,741**	**11,881 (9.42)**
	I/D/O/M	139,805	63,295	11,338 (8.11)
arXiv (300)	Remove	75,672	27,970	10,590(13.99)
	O/M	48,666	**24,630**	**5,457(11.21)**
	I/D/O/M	50,504	26,360	5,345(10.58)
ACL (67)	Remove	17,166	5,047	1,095 (6.38)
	O/M	11,182	**4,157**	**616 (5.51)**
	I/D/O/M	12,402	4,871	587 (4.73)
Wiki. (300)	Remove	14,883	3,114	1,651(11.09)
	O/M	6,173	**2,248**	**282 (4.57)**
	I/D/O/M	8,049	2,451	258 (3.21)

Non-NL(Object)/-target(Meta-info) parts were excluded

Only target word sequences were input

→ Drastic speed-up

→ Much higher coverage

Performance of Stanford Parser

Ar				
PMC (1,000)	Remove	170,099	58,865	18,621(10.89)
	O/M	126,176	**50,741**	**11,881 (9.42)**
	I/D/O/M	139,805	63,295	11,338 (8.11)
arXiv (300)	Remove	75,672	27,970	10,590(13.99)
	O/M	48,666	**24,630**	**5,457(11.21)**
	I/D/O/M	50,504	26,360	5,345(10.58)
ACL (67)	Remove	17,166	5,047	1,095 (6.38)
	O/M	11,182	**4,157**	**616 (5.51)**
	I/D/O/M	12,402	4,871	587 (4.73)
Wiki. (300)	Remove	14,883	3,114	1,651(11.09)
	O/M	6,173	**2,248**	**282 (4.57)**
	I/D/O/M	8,049	2,451	258 (3.21)

Impact of Tag Treatment on Performance of Enju Parser

Art. (#)	Treatment	# sentences	Time (s)	# failure (%)
PMC (1,000)	Remove	159,327	209,783	4,721 (2.96)
	O/M	112,285	135,752	810 (0.72)
	I/D/O/M	126,215	132,250	699 (0.55)
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	411 (1.00)
	I/D/O/M	43,208	87,952	348 (0.81)
ACL (67)	Remove	19,571	15,142	115 (0.59)
	O/M	9,819	9,481	63 (0.64)
	I/D/O/M	11,136	8,482	39 (0.35)
Wiki. (300)	Remove	10,561	14,704	1,161(10.99)
	O/M	5,026	6,743	67 (1.33)
	I/D/O/M	6,893	6,058	61 (0.88)

Impact of Tag Treatment on Performance of Enju Parser

Parsing time: 1~11%↓

Parsing failure: 0.2~0.5%↓

Ar	Treatment	# words/s	Time (s)	# failures (%)
PMC (1,000)	O/M	112,285	135,752	810 (0.72)
	I/D/O/M	126,215	**132,250**	**699 (0.55)**
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	411 (1.00)
	I/D/O/M	43,208	**87,952**	**348 (0.81)**
ACL (67)	Remove	19,571	15,142	115 (0.59)
	O/M	9,819	9,481	63 (0.64)
	I/D/O/M	11,136	**8,482**	**39 (0.35)**
Wiki. (300)	Remove	10,561	14,704	1,161(10.99)
	O/M	5,026	6,743	67 (1.33)
	I/D/O/M	6,893	**6,058**	**61 (0.88)**

Embedded sentences (Independent) were separated
→ **Correct & shorter sentences** were increased

Parser could perform better

Parsing time: 1~11%↓

Parsing failure: 0.2~0.5%↓

PMC (1,000)	O/M	112,285	135,752	810 (0.72)
	I/D/O/M	**126,215**	**132,250**	**699 (0.55)**
arXiv (300)	Remove	74,762	108,831	2,047 (2.74)
	O/M	41,265	89,200	411 (1.00)
	I/D/O/M	**43,208**	**87,952**	**348 (0.81)**
ACL (67)	Remove	19,571	15,142	115 (0.59)
	O/M	9,819	9,481	63 (0.64)
	I/D/O/M	**11,138**	**8,482**	**39 (0.35)**
		10,561	14,704	1,161(10.99)
		5,026	6,743	67 (1.33)
		6,893	**6,058**	**61 (0.88)**

Detected sentences:
5~37%↑

Impact of Tag Treatment on Performance of Stanford Parser

Parsing time: 7~25%↑

Parsing failure: 0.6~1.5%↓

PMC (1,000)	O/M	126,176	**50,741**	11,881 (9.42)
	I/D/O/M	139,805	63,295	**11,338 (8.11)**
arXiv (300)	Remove	75,672	27,970	10,590(13.99)
	O/M	48,666	**24,630**	5,457(11.21)
	I/D/O/M	50,504	26,360	**5,345(10.58)**
ACL (67)	Remove	17,166	5,047	1,095 (6.38)
	O/M	11,182	**4,157**	616 (5.51)
	I/D/O/M	12,402	4,871	**587 (4.73)**
Wiki. (300)	Remove	14,883	3,114	1,651(11.09)
	O/M	6,173	**2,248**	282 (4.57)
	I/D/O/M	8,049	2,451	**258 (3.21)**

Embedded sentences (Independent) were separated
→ **Correct & shorter sentences** were increased

Parsing time: 7~25%↑

Parsing failure: 0.6~1.5%↓

PMC (1,000)	O/M	126,176	**50,741**	11,881 (9.42)
	I/D/O/M	**139,805**	63,295	**11,338 (8.11)**
arXiv (300)	Remove	75,672	27,970	10,590(13.99)
	O/M	48,666	**24,630**	5,457(11.21)
	I/D/O/M	**50,504**	26,360	**5,345(10.58)**
ACL (67)	Remove	17,166	5,047	1,095 (6.38)
	O/M	11,182	**4,157**	616 (5.51)
	I/D/O/M	**12,402**	4,871	**587 (4.73)**
		14,883	3,114	1,651(11.09)
		6,173	**2,248**	282 (4.57)
		8,049	2,451	**258 (3.21)**

Detected sentences: 5~37%↑

Embedded sentences (Independent) were separated
→ **Correct & shorter sentences** were increased

Skipped (>50 word) sentences were decreased

Parsing time: 7~25%↑

Parsing failure: 0.6~1.5%↓

PMC (1,000)	O/M	126,176	**50,741**	11,881	(9.42)
	I/D/O/M	**139,805**	63,295	**11,338**	**(8.11)**
arXiv (300)	Remove	75,672	27,970	10,590	(13.99)
	O/M	48,666	**24,630**	5,457	(11.21)
	I/D/O/M	**50,504**	26,360	**5,345**	**(10.58)**
ACL (67)	Remove	17,166	5,047	1,095	(6.38)
	O/M	11,182	**4,157**	616	(5.51)
	I/D/O/M	**12,402**	4,871	**587**	**(4.73)**
		14,883	3,114	1,651	(11.09)
		6,173	**2,248**	282	(4.57)
		8,049	2,451	**258**	**(3.21)**

Detected sentences:
5~37%↑

Discussion: What Thorough & Efficient Document Processing Brings About

- Shallow analysis with simple approach (word count etc.) → removing tags will be enough
  - # words is not affected by embedded sentences
  - Some non-NL seq. canceled by many NL seq.
- Detailed/precise analysis (discourse analysis/translation/grammar extraction/etc.)
  - Even subtle utterance cannot be overlooked
  - Seq. other than body text should be excluded

  = Presumed condition in most NLP challenges

Significance of Bridging Real-world Documents and NLP Technologies

- True goal of NLP challenges = analyze any real-world(, richly formatted) documents
- Proper framework enables conventional NLP tools to process real-world text without significant loss of performance

"Adequately bridging target real-world documents and NLP technologies"
= Crucial task for utilizing full benefit brought by NLP in ubiquitous application

Summary

- We proposed framework for data conversion between XML-tagged text and I/O of NLP
  - – According to classification of tag functions
- We succeeded in obtaining plain text seq. from target doc. by classifying 20% of tags

  → Much more thorough & efficient parsing of target doc. than with naively tag-removed text

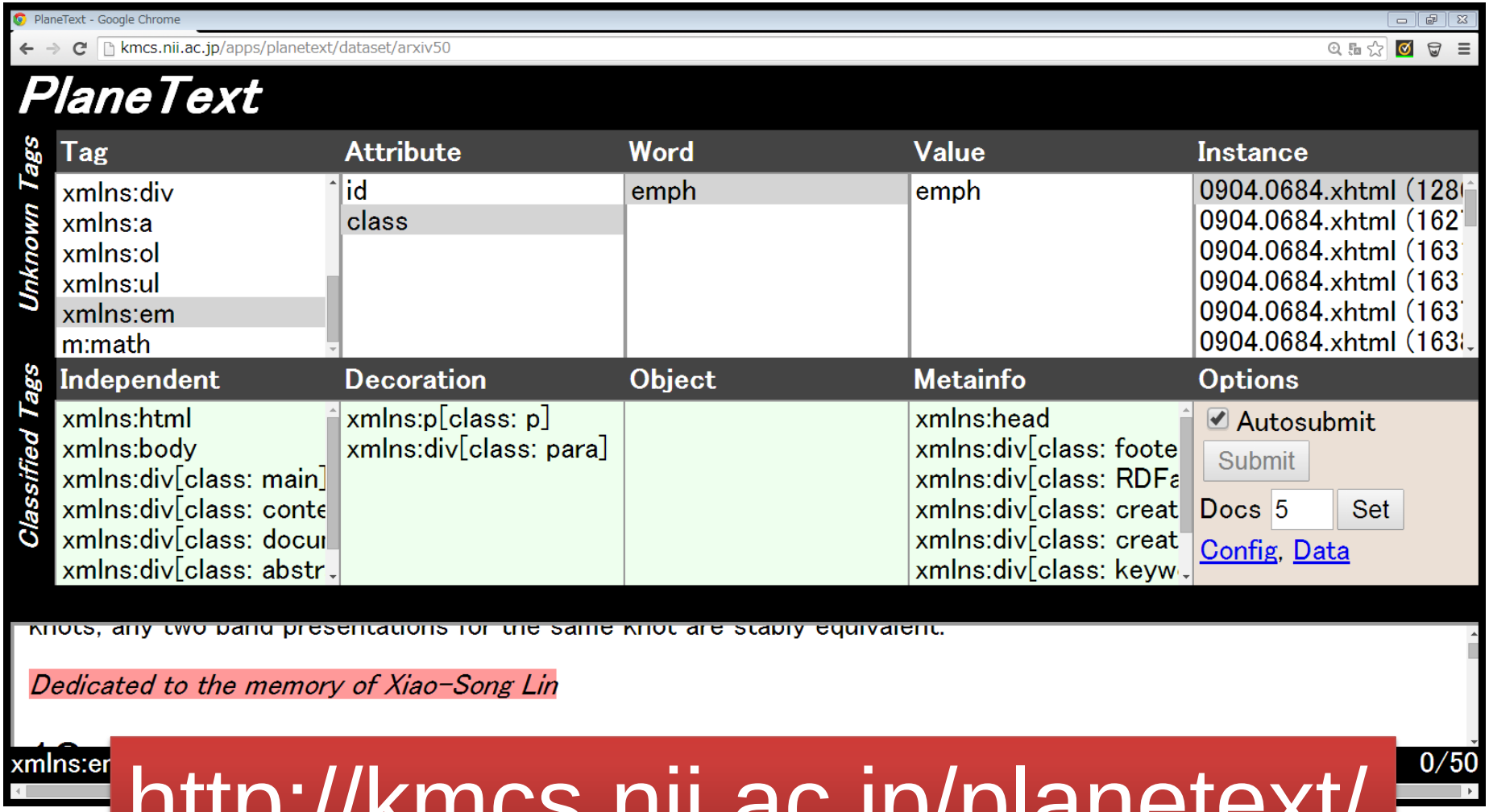
→ Emphasize significance of bridging real-world documents and NLP technologies

Future Work

- Release tool for converting XML doc. into plain text seq. utilizing our framework

  → Share further discussion on applying NLP tools to various real-world documents
- Treat tag & tagged regions more flexibly
  - Treatment of textual parts in Object regions
- Apply NLP to various formats of documents
  - OCR data / presentation slides / etc.

Downloadable Package Almost Ready (will be Released in September)



http://kmcs.nii.ac.jp/planetext/

Thank you !