An Annotation System for Enhancing Quality of Natural Language Processing

Hideo Watanabe*, Katashi Nagao**, Michael C. McCord*** and Arendse Bernth***

* IBM Research,
Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato,
Kanagawa 242-8502, Japan
hiwat@jp.ibm.com

** Dept. of Information Engineering
Nagoya University
Furo-cho, Chikusa-ku,
Nagoya 464-8603, Japan
nagao@nuie.nagoya-u.ac.jp

*** IBM T. J. Watson
Research Center
Route 134, Yorktown Heights,
NY 10598, USA
mcmccord@us.ibm.com,
arendse@us.ibm.com

Abstract

Natural language processing (NLP) programs are confronted with various difficulties in processing HTML and XML documents, and have the potential to produce better results if linguistic information is annotated in the source texts. We have therefore developed the *Linguistic Annotation Language* (or LAL), which is an XML-compliant tag set for assisting natural language processing programs, and NLP tools such as parsers and machine translation programs which can accept LAL-annotated input. In addition, we have developed a LAL-annotation editor which allows users to annotate documents graphically without seeing tags. Further, we have conducted an experiment to check the translation quality improvement by using LAL annotation.

1 Introduction

Recently there has been increasing interest in applying natural language processing (NLP) systems, such as keyword extraction, automatic text summarization, and machine translation, to Internet documents. However, there are various obstacles that make it difficult for them to produce good results. It is true that NLP technologies are not perfect, but some of the difficulties result from problems in HTML. Further, in general, if linguistic information is added to source texts, it greatly helps NLP programs to produce better results. In what follows, we would like to show some examples related to machine translation.

In general, it is very helpful for machine translation programs to know boundaries on many levels (such as sentence, phrases, and words) and to know word-to-word dependency relations. For instance, in the following example, since "St." has two possible meanings, "street" and "saint," it is difficult to determine whether the following example consists of one or two sentences.

> I went to Newark St. Paul lived there two years ago.

As another example, the following sentence has two interpretations; one interpretation is that what he likes is people and the other interpretation is that what he likes is accommodating.

> He likes accommodating people.

If there are tags indicating the direct-object modifier of the word "like," then the correct interpretation is possible. NLP may be able to resolve these ambiguities eventually by using advanced context processing techniques, but current NLP technology generally needs a hint from the author for these sorts of ambiguities.

Further, there are issues in HTML/XML. When MT systems are applied to Web pages, most of the errors are generated by the linguistic incompleteness of MT technology, but some are generated by problems in HTML and XML tag usage. For instance, writers often use `<br>` tag to sentence termination. Sometimes writers intend that a `<br>` tag should terminate the sentence (even without terminating punctuation such as a period), and in other cases writers intend `<br>` only as a formatting device. In the HTML `<table>` shown in Figure 1, the writer intends each line of a cell to express one linguistic unit. The MT program cannot tell whether each line is a unit for translation, or, instead, the two lines form one unit. In this example, some MT programs would try to produce a translation of a unit "NetVista Models ThinkPad News."

As shown in the above examples, NLP applications do not achieve their full potential, on account of problems unrelated to the essential NLP processes. If tags expressing linguistic information

```
<table><tr><td>
<a href="...">NetVista Models</a><br>
<a href="...">ThinkPad News</a><br>
</td></tr></table>
```

Figure 1: An example of using ⟨br⟩ tags in a table

are inserted into source documents, they help NLP programs recognize document and linguistic structures properly, allowing the programs to produce much better results. At the same time, it is true that NLP technologies are incomplete, but their deficiencies can sometimes be circumvented through the use of such tags. Therefore, this paper proposes a set of tags for helping NLP programs, called *Linguistic Annotation Language* (or LAL).

2 Linguistic Annotation Language

LAL is an XML-compliant tag set and its XML namespace prefix is `lal`.

The LAL tag set is designed to be as simple as possible for the following reasons: (1) A simple tag set is easier for developers to check manually. (2) An easy-to-use annotation tool is mandatory for this annotation scheme. Simplicity is important for making an easy-to-use annotation tool, since if we use a feature-rich tag set, the user must check many annotation items.

2.1 Basic Tags

The sentence tag `s` is used to delimit a sentence.

```
<lal:s>This is the first sentence.</lal:s>
<lal:s>This is the second sentence.</lal:s>
```

The attribute `type="hdr"` means that the sentence is a title or header.

The word tag `w` is used to delimit a word. It can have attributes for additional information such as base form (`lex`), part-of-speech (`pos`), features (`ftrs`), and sense (`sense`) of a word. The values of these attributes are language-dependent, and are not described in this paper because of space limitations. The following example illustrates some of these tags and attributes.

```
<lal:s>
<lal:w lex="this" pos="det">This</lal:w>
<lal:w lex="be" pos="verb" ftr="sg,3rd">
   is</lal:w>
<lal:w lex="a" pos="det">a</lal:w>
<lal:w lex="pen" pos="noun" ftr="sg,count">
   pen</lal:w>
</lal:s>
```

The dependency (or word-to-word modification) relationship can be expressed by using the `id` and `mod` attributes of a word tag; that is, a word can have the ID value of its modifiee in a `mod` attribute. The ID value of a `mod` attribute must be an ID value of a word tag or a segment tag. For instance, the following example contains attributes showing that the word "with" modifies the word "saw," meaning that "she" has a telescope.

```
She <lal:w id="w1" lex="see" pos="v"
sense="see1">saw</lal:w> a man
<lal:w mod="w1">with</lal:w>
a telescope.
```

The phrase (or segment) tag `seg` is used to specify a phrase scope on any level. In addition, you can specify the syntactic category for a phrase by using an optional attribute `cat`. The following example specifies the scope of a noun phrase "a man ... a telescope," and it is a noun phrase. This also implies that the prepositional phrase "with a telescope" modifies the noun phrase "a man."

```
She saw <lal:seg cat="np">a man with a
telescope</lal:seg>.
```

The attribute `para="yes"` means that the segment is a coordinated segment. The following example shows that the word "software" and the word "hardware" are coordinated.

```
This company deals with <lal:seg cat="np"
para="yes">software and hardware</lal:seg>
for networking.
```

The `ref` attribute has the ID value of the referent of the current word. This can be used to specify a pronoun referent, for instance:

```
<lal:s>He bought <lal:seg id="w1">a
new car</lal:seg> yesterday.</lal:s>
<lal:s>She was very surprised to
learn that <lal:w ref="w1">it</lal:w>
was very expensive.</lal:s>
```

2.2 Expressing Multiple Parses

As mentioned earlier, since natural language contains ambiguities, it is useful for LAL annotation to have a mechanism for expressing syntactic ambiguities.

We have introduced a parse identifier (or PID) in attribute values for distinguishing parses. An attribute value which may be changed according to parses can be allowed to be expressed as space-separated multiple values, each of which consists of a PID prefix followed by a colon and an attribute value.

```
<lal:s>
<lal:w id="1" mod="2">He</lal:w>
<lal:w id="2" mod="0">likes</lal:w>
<lal:w id="3" mod="p1:2 p2:4">
    accommodating</lal:w>
<lal:w id="4" mod="p1:3 p2:2">people
</lal:w>.</lal:s>
```

This example shows that there are two interpretations whose PIDs are `p1` and `p2`, and that the `p1` interpretation is "He likes people" and `p2` is "He likes accommodating."

3 LAL-Aware NLP Programs

We have modified certain NLP systems to be LAL-aware. ESG [5, 6] is an English parsing system developed by the IBM Watson Research Center, and updated to accept and generate LAL-annotated English. We have also developed a Japanese parsing system with LAL output functionality. These LAL-aware versions of parsers are used as a back-end process to show users the system's default interpretation for a given sentence in the LAL-annotation editor described below.

Further, the English to German, French, Spanish, Italian and Portuguese translation engines [6, 7] and English to Japanese translation engine [9] are modified to accept LAL-annotated English HTML input.[1]

4 The LAL-Annotation Editor

Since inserting tags into documents manually is not generally an easy task for end users, it is important to provide an easy-to-use GUI-based editing environment. In developing such an environment, we took into consideration the following points: (1) Users should not have to see any tags. (2) Users should not have to see internal representations expressing linguistic information. (3) Users should be able to view and modify linguistic information such as feature values, but only if they want to.

Considering these points, we have found that most of the errors made by NLP programs result from their failure to recognize the phrasal structures of sentences. Therefore, we have decided to show only a structural view of a sentence in the initial screen; other information is shown only if the user requests it.

The important issue here is how to represent the syntactic structure of a sentence to the user. NLP programs normally deal with a linguistic structure by means of a syntactic tree, but such a structure is not necessarily easy for end users to understand. For instance, Figure 2 shows the dependency structure of the English sentence "IBM announced a new computer system for children with voice function." This dependency structure is difficult for end users, partly because a dependency tree does not keep the surface word order, so that it is difficult to map it to the original sentence quickly.[2] Therefore, an important property for the linguistic structural view is that users can easily reconstruct the original surface sentence string.

The next important issue is how easily a user can understand the overall linguistic structure. If a user is, at first, presented with detailed linguistic structure at the word level, then it is difficult to grasp the important linguistic skeleton of a sentence. Therefore, another necessary property is to give users a view in which the overall sentence structure is easily recognized.

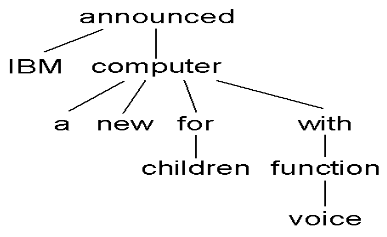


Figure 2: An example of tree structure of an English sentence

With these requirements in mind, we have developed a GUI tool called the LAL Editor. To satisfy the last requirement, this editor has two presentation modes: the *reduced* presentation view and the *expanded* presentation view. In the reduced presentation view, a main verb and its modifiers are basic units for presenting dependencies, and they are located on different lines, keeping the surface order. Figure 3 shows an example of this reduced presentation view. In this view, since dependencies that are obvious for native speakers (e.g. "a" and "computer") are not displayed explicitly, the user can concentrate on dependencies between key

[1]In addition, Watanabe [11] reported on an algorithm for accelerating CFG-parsing by using LAL tag information, and it is implemented in the above English-to-Japanese translation engine.

[2]You must perform an inorder tree walk to reconstruct a surface sentence string.

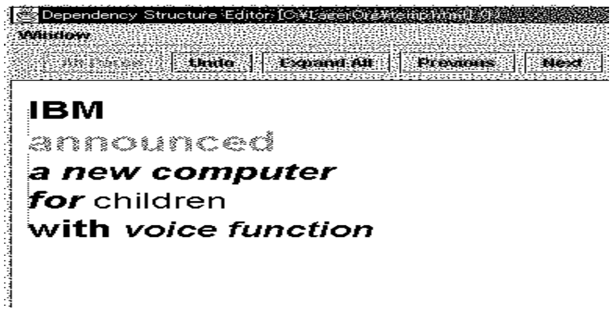


Figure 3: Screen Images of LAL Editor - Reduced View

units (or phrases). If the user finds any dependency errors in the reduced view, he or she can enter the expanded view mode in which all words are basic units for presenting dependencies. Figure 4 (a, b) shows examples of this expanded view. In these views, to satisfy the former requirement, dependencies between basic units are expressed by using indentation. Therefore you can easily reconstruct the surface sentence string by just looking at words from top to bottom and from left to right, and easily know dependencies of words by looking at words located in the same column. For details of the algorithm, see [12].

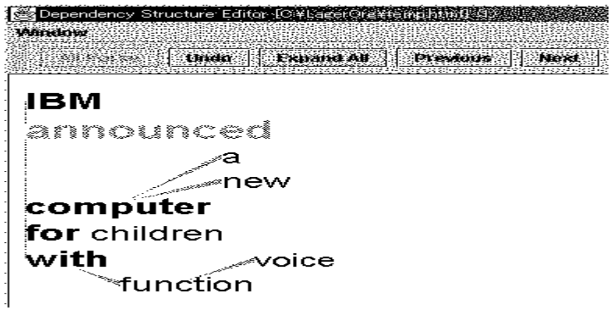
In Figure 3, you can easily grasp the overall structure. In this case, since the dependencies between "for" and "announced," and "with" and "announced" are wrong, the user can change the mode to the expanded view (as shown in Figure 4 (a)). In this view, the user can change dependencies by dragging a modifier to the correct modifiee using a mouse. The corrected dependency structure is shown in Figure 4 (b).

In addition, the LAL Editor has the capability of testing translation by using LAL annotation. Figure 5 shows a window in which the top pane shows the input sentence, the second pane shows the LAL-annotation of the input, the third pane shows the translation result using the LAL annotation, and the fourth pane shows the default translation without using the LAL annotation. The user can easily check whether the current annotation can improve translations.

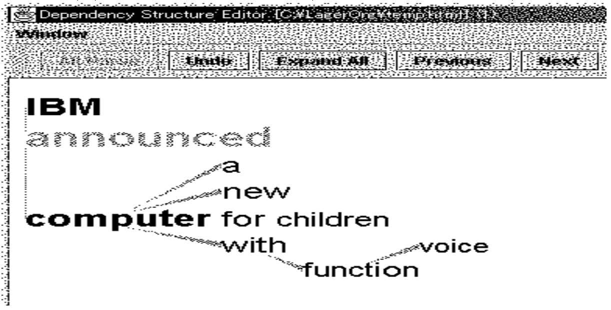
5 Experiment

We have conducted a small experiment for evaluating LAL annotation to our English-to-Japanese machine translation system[9]. We gathered about 60 sentences from Web pages in the computer domain, and added LAL annotation to these sentences with the LAL annotation editor. In this experiment, only word-to-word modifications were corrected. Due to severe parsing errors and glitches of the annotation editor, 53 of the 60 sentences were used in this experiment. The average sentence length for this test set was 21 words. Two evaluators assigned a quality evaluation ranging from 1 (worst) to 5 (best) for each translation, with and without use of annotation.

Translation results for 18 sentences (about 34%) were better for the annotated case than the non-annotated case. These better sentences were 1.16

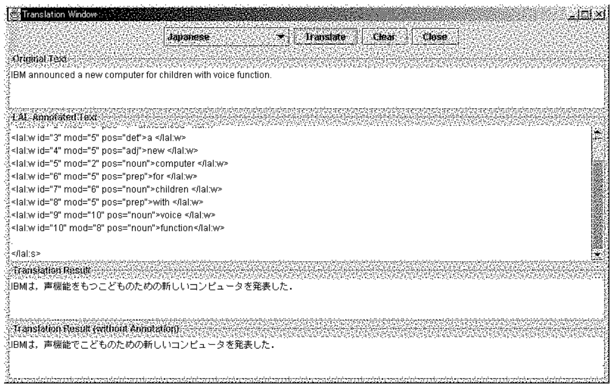


(a) Expanded View (before correction)



(b) Expanded View (after correction)

Figure 4: Screen Images of LAL Editor - Expanded View



IBM announced a new computer for children with voice function.

```
<lal:w id="3" mod="5" pos="det">a </lal:w>
<lal:w id="4" mod="5" pos="adj">new </lal:w>
<lal:w id="5" mod="2" pos="noun">computer </lal:w>
<lal:w id="6" mod="5" pos="prep">for </lal:w>
<lal:w id="7" mod="6" pos="noun">children </lal:w>
<lal:w id="8" mod="5" pos="prep">with </lal:w>
<lal:w id="9" mod="10" pos="noun">voice </lal:w>
<lal:w id="10" mod="8" pos="noun">function</lal:w>

</lal:s>
```

IBMは，声機能をもつこどものための新しいコンピュータを発表した．

IBMは，声機能でこどものための新しいコンピュータを発表した．

Figure 5: Translation test window of LAL Editor

points better (27% better in quality score). On the other hand, 26 sentences (about 49%) were not changed, and 9 sentences (about 17%) were worse. The main reason why these 9 sentences were worse was the structural mismatch between the output of the LAL Editor and the expected structure of EtoJ translation system, since the LAL Editor and the EtoJ MT system use different parsing systems. We have developed a structure conversion routine from LAL editor output to EtoJ input, but it does not yet cover all situations. This is the reason why these 9 sentences become worse.

Note that this experiment only uses word-to-word modification corrections, so there is room for producing better translations if we use other types of annotation such as part-of-speech, and word sense.

6 Discussion

There have been several efforts to define tags for describing language resources, such as TEI [10], OpenTag [8], CES [1], EAGLES [2], GDA [3]. The main focus of these efforts other than GDA has been to share linguistic resources by expressing them in a standard tag set, and therefore they define very detailed levels of tags for expressing linguistic details. GDA has almost the same purposes but it has also defined a very complex tag set. This complexity discourages people from using these tag sets when writing documents, and it also becomes difficult to make an annotation tool for these tags. LAL is not opposed to these previous efforts, but attempts to strike a useful balance between expressiveness and simplicity, so that annotation can be used widely.

As mentioned in the discussion of the experiment, there is an issue when the parsing system of LAL editor and the parsing system of a NLP tool which accepts the output of LAL editor are different. As mentioned before, we used the ESG parser for producing LAL-annotated English, and Japanese-to-English MT system for accepting LAL-annotated English. Since these systems have been independently developed based on different approaches by different developers, we found there are some structural differences. For instance, given a prepositional phrase Prep N, ESG's head word of the prepositional phrase is Prep, but EtoJ MT engine's head is N. In most cases, we can make systematic conversion routines for different structures. In fact, for most of sentences whose translation is worse when annotation is used, we can provide structural conversion routines for linguistic structures included in them. The basic idea of LAL-awareness for NLP tools is that an NLP tool uses LAL information as much as possible, but if LAL information produces a severe conflict with the internal processing, then such information should not be used. Our EtoJ MT program was basically implemented this way based on the algorithm described in [11], but we seem to need more research on this issue.

7 Conclusion

In this paper, we have proposed an XML-compliant tag set called Linguistic Annotation Language (or LAL), which helps NLP programs perform their tasks more correctly. LAL is designed to be as simple as possible so that humans can use it with minimal help from assisting tools. We have also developed a GUI-based LAL annotation editor, and have shown in an experiment that use of LAL annotation enhances translation quality. We hope that wide acceptance of LAL will make it possible to use more intelligent Internet tools and services.

References

[1] CES, "Corpus Encoding Standard (CES)," (http://www.cs.vassar.edu/CES/)

[2] EAGLES, "Expert Advisory Group on Language Engineering Standards," (http://www.ilc.pi.cnr.it/EAGLES/home.html)

[3] GDA, "Global Document Annotation," (http://www.etl.go.jp/etl/nl/gda/)

[4] Koichi Hashida, Katashi Nagao, et. al, "Progress and Prospect of Global Document Annotation," (in Japanese) Proc. of 4th Annual Meeting of the Association of Natural Language Processing, pp. 618–621, 1998

[5] McCord, M. C., "Slot Grammars," Computational Linguistics, Vol. 6, pp. 31–43, 1980.

[6] McCord, M. C., "Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars," in (ed) R. Studer, Natural Language and Logic: International Scientific Symposium, Lecture Notes in Computer Science, pp. 118–145, Springer Verlag, 1990.

[7] McCord, M. C., and Bernth, A., "The LMT Transformational System," Proc. of Proceedings of AMTA-98, pp. 344–355, 1998.

[8] OpenTag, "A Standard Extraction/Abstraction Text Format for Translation and NLP Tools," (http://www.opentag.org/)

[9] Takeda, K., "Pattern-Based Machine Translation," Proc. of 16th COLING, Vol. 2, pp. 1155–1158, August 1996.

[10] TEI, "Text Encoding Initiative (TEI)," (http://www.uic.edu:80/orgs/tei/)

[11] Watanabe, H., "A Method for Accelerating CFG-Parsing by Using Dependency Information," Proc. of 18th COLING, 2000.

[12] Watanabe, H., Nagao, K., McCord, M. C., and Bernth, A., "Improving Natural Language Processing by Linguistic Document Annotation," Proc. of COLING 2000 Workshop for Semantic Annotation and Intelligent Content, pp. 20–27, 2000.