Improving Natural Language Processing by Linguistic Document Annotation

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

* IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato,
Kanagawa 242-8502, Japan
watanabe@trl.ibm.co.jp, nagao@trl.ibm.co.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 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. It consists of linguistic information tags such as tags specifying word/phrasal boundaries, and task-dependent instruction tags such as tags defining the scope of translation for machine translation programs. We have also developed an LAL-annotation editor to facilitate users to annotate documents without seeing tags.

1 Introduction

The rapid expansion of the Internet has accelerated the proliferation of documents written in HTML and XML. Programs for performing natural language processing (or NLP) tasks such as keyword extraction, automatic text summarization, and machine translation have to be able to deal with such 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 in a source text, it greatly helps NLP programs to produce a better result. Consider the following situations. When you use a search engine, you are often returned a list of thousands of documents matching your query. Most of the current search engines just use superficial information such as keywords. If search engines used richer linguistic information such as syntactic structures, they would be able to provide a more appropriate ranking of retrieved documents.

When you generate a summary of an HTML page by using an automatic summary generation program, a copyright notice is sometimes included in the summary text. Most of the current automatic summary programs simply select important sentences on the basis of surface clues such as keywords and sentence location in a document. As a result, they sometimes select a copyright notice located at the end of a document, since sentences located at the ends of documents tend to be important. This problem can be avoided if the main part of document is explicitly declared.

HTML Source
I used the h3 tag to emphasize ⟨h3⟩this part⟨/h3⟩.
Rendering Image
I used the h3 tag to emphasize **this part** .

Figure 1: An example of wrong usage of HTML tag

Further, when you use a Web page translation program, you sometimes see wrong translations. Most of them are generated by the incompleteness of MT technology, but some are generated by problems involving HTML and XML tag usage. For instance, writers often misuse tags to obtain certain stylistic effects. For instance, some writers use a heading tag to obtain large font and bold style, as shown in Fig. 1. Most machine translation (MT) engines change the translation logic when a sentence is a title, so this wrong use of heading tags sometimes causes a wrong translation result. However, the likelihood of this will decrease if a style sheet mechanism is widely accepted by Web authors in the future.

Another example of HTML/XML problems is the recognition of a sentence. There are many cases in which a sentence is terminated not by a period, but merely by a ⟨br⟩ tag, for instance, in an HTML table environment. As shown in Fig. 2, a writer

```
⟨table⟩
⟨tr⟩
⟨td⟩
⟨a href="..."⟩Internet Shops⟨/a⟩⟨br⟩
⟨a href="..."⟩Cool Sites⟨/a⟩⟨br⟩
⟨a href="..."⟩What's New!⟨/a⟩
⟨/td⟩
⟨/tr⟩
⟨/table⟩
```

Figure 2: An example of using ⟨br⟩ tags in a table

sometimes intends each line in a cell of a table to express a sentence, even if there is no punctuation at the end of the line. The MT program cannot tell whether each line is a sentence or whether these three lines form one sentence.

In general, it is very helpful for machine translation programs to know boundaries in many levels (such as sentence, phrases, and words) and to know word-to-word dependency relations. For instance, in the following example, "St." has two possible meanings: "street" and "saint." Therefore, we cannot determine whether the following example consists of one or two sentences without parsing it.

> I went to New Ark St. Paul lived there
> in two years ago.

As another example, the following sentence is ambiguous so that there are two interpretations; one interpretation is that what he likes is people and the other interpretation is that what he likes is accommodating. If there are tags indicating the direct-object modifier of the word "like," then the correct interpretation is possible.

> He likes accommodating people.

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

2.1 Design Principle

Linguistic Annotation Language (or LAL) is an XML-compliant tag set. It was designed with the following considerations:

- Simplicity: Although we consider that LAL tags should be as simple as possible so that humans will want to try annotating documents manually, we must offer an assisting tool for annotation in practice. The simplicity is also important to make an easy-to-use annotation tool, since if we use a feature-rich tag set, a user must check many annotation items. Therefore, the main part of LAL consists of syntactic annotation tags for specifying boundaries at many levels, and limited semantic annotation tags for specifying limited semantic information. In practice, boundary specification with limited linguistic information can cover most NLP problems, so it is sufficiently effective for NLP programs in terms of increasing accuracy.
- Assistance with NLP Tasks: The main purpose of LAL is to help NLP programs to perform their tasks much better. Therefore, in addition to tags for linguistic information, it should contain task-dependent instruction tags such as a tag indicating translation scope.

LAL tags are usually expressed by using XML namespaces. Their XML namespace prefix is **lal**. Since linguistic information annotation inherently has different annotation directions, linguistic annotation tags may overlap with other HTML and XML tags. In this case, LAL tags are expressed in the form of the processing instructions.

2.2 LAL Tags

LAL tags are classified into linguistic information tags and task-dependent instruction tags. Linguistic information tags are further classified into syntactic and semantic tags. Each type of LAL tag is described below.

2.2.1 Syntactic Information Tags

This category has tags for sentences, words, and phrases. These tags are mainly used to specify a scope for each unit.

Sentence: The sentence tag **s** is used to specify a sentence scope.

```
⟨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.

Word: The word tag **w** is used to specify a word scope. 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 due to the space limitation.

⟨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, each word can have an ID value of its modifiee in a mod attribute. The ID value of a mod attribute must be an ID value of a word or a seg tag. For instance, the following example contains attributes showing that the word "with" modifies the word "saw," and which means 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 *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 a new ⟨lal:w id="w1"⟩car
⟨/lal:w⟩ yesterday.⟨/lal:s⟩
⟨lal:s⟩She was very surprised to learn
that ⟨lal:w ref="w1"⟩it⟨/lal:w⟩ was very
expensive.⟨/lal:s⟩

Phrase: The phrase tag **seg** is used to specify a phrase scope in any level. The following example specifies the scope of a noun phrase "a man ... a telescope," and this also implies that a prepositional phrase "with a telescope" modifies a noun phrase "a man."

She saw ⟨lal:seg⟩a man with a telescope⟨/lal:seg⟩.

In addition to boundary specification, you can specify syntactic category for a phrase by using an optional attribute *cat*. The value of the cat attribute is also dependent on languages and systems. The following example specifies that a phrase "a man with a telescope" is a noun phrase.

He saw ⟨lal:seg cat="np"⟩a man with a
telescope⟨/lal:seg⟩.

The attribute *para="yes"* means that this segment also means a scope of coordination. The following example shows that a word "software" and a word "hardware" are coordinated.

This company deals with ⟨lal:seg cat="np"
para="yes"⟩software and hardware⟨/lal:seg⟩
of computer.

2.2.2 Semtantic Information Tags

LAL has the following limited semantic tags which are selected since these expressions are often used.

The **proper** tag is used to specify a proper name, and it has the *type* attribute specifying a sub-class of a proper name, such as person, place, organization, or country.

⟨lal:proper type="country"⟩Luxembourg
⟨/lal:proper⟩

This information is effective for translation, for instance, to select an appropriate translation word of a verb which may be changed if a subject of the verb has a human property, etc.

You can also use **acronym** and **abbr** elements defined in HTML to specify an acronym and an abbreviation terms. They are a little bit extended to have the expan attribute to specify an expanded form of abbreviation or acronym like the abbr tag of TEI[1]

⟨lal:acronym expan="International Business Machines"⟩IBM⟨/lal:acronym⟩

The **date** tag is used to specify a date expression, whereas, the **time** tag is used to specify a time expression. The *value* attribute is used to specify a normalized form of a date or time defined by ISO 8601 [5].

⟨lal:date value="2000-01-01"⟩Jan. 1, 2000
⟨/lal:date⟩
⟨lal:time value="15:00"⟩3:00 PM⟨/lal:time⟩

The **num** tag is used to specify a number expression (e.g., two million and twenty-one). The *type* and *value* attributes are used to specify a normalized form of the number expression. Further, the **money** tag is used to specify money expression, in particular, to add monetary unit information.

[1] Some of LAL tags have the same name as those defined in previous efforts such as TEI, since we do not like to introduce new tag names, rather, would like to reuse existing names if the meaning is the same.

⟨lal:num type="cardinal" value="21"⟩twenty one⟨/lal:num⟩
⟨lal:money unit="usd"⟩ ⟨lal:num value="1000"⟩ one thousand ⟨/lal:num⟩ dollars ⟨/lal:money⟩

2.2.3 Task-Dependent Instruction Tags

Machine Translation: For machine translation of HTML or XML documents, we need unique algorithms to detect which segments are to be translated and which are not. In particular, XML can introduce new tags, whose semantics we generally do not know. Therefore, we need an instructional tag to inform a machine translation program whether or not a text segment is to be translated.

If an MT program encounters ⟨lal:tranStop/⟩, it passes over the subsequent text until it encounters ⟨lal:tranStart/⟩.

Text Summarization: Automatic text summarization programs have problem in handling HTML texts with the result that unimportant sentences are included in the summary texts. This problem occurs because the program extracts important sentences whose importance it calculates on the basis of the number of important keywords, the location in a text, and so on [16]. Thus, a summary program may select unimportant sentences if it does not know the main text area in a document. A typical HTML text has related information areas such as a list of related links, the name of the reporter, and a copyright notice, in the beginning and ending area, and these areas can cause a wrong summary to be generated. Therefore, we need a tag that specifies which segments should be processed in order to generate a summary of a document.

If a summary program encounters ⟨lal:smrycalcStop/⟩, it stops summary calculation until it encounters ⟨lal:smrycalcStart/⟩. Therefore, additional information parts such as a copyright notice, and a writer's signature, should not be included in this summary calculation scope.

3 LAL-aware NLP Programs

We have modified some NLP systems to be LAL-aware[2].

ESG [7, 8] is an English parsing system developed by IBM Watson Research Center, and updated to accept and generate LAL-annotated English. This LAL-aware version of ESG is used as a backend process to show users an interpretation of a system of a given English sentence in the LAL-annotation editor described in the next section.

[2]They support only syntactic information tags currently, and will support other tags later.

KNP [6] is a Japanese dependency parsing system developed by Kyoto University. We have developed a post-process routine to convert KNP parsing result into LAL format. This is also used as a backend process to show the initial interpretation of a given Japanese sentence in the LAL-annotation editor.

Further, we have modified IBM's English to German, French, Spanish, and Italian translation engines [8, 9, 10] and English to Japanese translation engine [13, 14, 17] to accept LAL-annotated English HTML input.

In addition, we have developed an algorithm for accelerating CFG-parsing process by using LAL tag information[3] [19], and this algorithm is implemented in the English-to-Japanese translation engine mentioned above.

4 LAL-Annotation Editor

Since inserting tags into documents manually is not generally an easy task for end users, it is important to provide a GUI-based annotation editor. In developing such an editor, we took into consideration the following points:

- Users should not have to see any tags.
- Users should not have to see internal representations expressing linguistic information.
- Users should be able to view and modify linguistic information such as feature values, but only if they want.

With respect to the above points, we have found that most of the errors made by NLP programs result from their failure to recognize the linguistic structures of sentences. Therefore, the LAL editor shows only a structural view of a given sentence; 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, Fig. 3 shows the dependency structure of the English sentence "IBM announced a new computer system for children with voice function." This dependency structure is not easy to understand for end users, partly because it is difficult to remind the original sentence quickly due to

[3]This does not depend on LAL, rather it is a general algorithm applicable for CFG-parsing when any dependency information is given.

not keeping the surface word order in a given sentence in this structure[4]. Therefore, the necessary property of a linguistic structural view is for users to easily reconstruct the original surface sentence string.

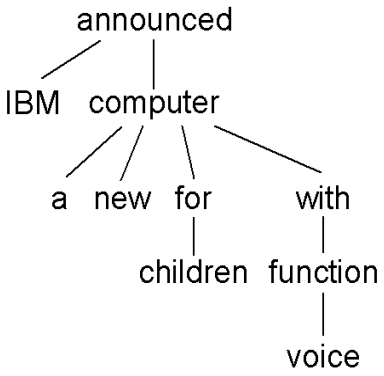


Figure 3: An example of tree structure of an English sentence

Considering this requirement, we have developed an algorithm to show linguistic dependency structure with keeping the surface word order which shows dependencies by indentations. Fig. 5 shows examples of linguistic structural view by this algorithm. In these examples, 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 at the same time.

The next important issue is that how easily a user can understand the overall linguistic structure. If a user is, at first, presented with detailed linguistic structure in 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.

To suffice this requirement, we have introduced 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 in different lines with keeping the surface order. Fig. 5 (a) shows an example of this reduced presentation view. In this view, since the obvious dependencies for native speakers (e.g. "a" and "computer") are not displayed explicitly, a user can concentrate on dependencies between key units (or phrases). If a user find 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. Fig. 5 (b) and (c) shows examples of this expanded view.

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

```
1  Locate the root at an appropriate position;
2  Add the root to node-list;
3  while node-list ≠ φ {
4      curunit ← remove-first-element(node-list);
5      curline ← the row of curunit;
6      Add pre-modifiers of curunit to mod-list and
       sort it by the distance with curunit in the as-
       cending order;
7      while mod-list ≠ φ {
8          mod ← remove-first-element(mod-list);
9          If the forward modification is major in the
           current language, mod is the nearest pre-
           modifier, and there is no words between
           mod and curunit, then {
10             Locate mod just before curunit;
11         } else {
12             Insert a new row just before the row of
               the curline, and make it curline;
13             Locate mod in curline at the column
               after that in which the last character
               of curunit is located.
14         }
15     }
16     curline ← the row of curunit;
17     Add post-modifiers of curunit to mod-list and
       sort it by the distance with curunit in the as-
       cending order;
18     while mod-list ≠ φ {
19         mod ← remove-first-element(mod-list);
20         If the backward modification is major in
           the current language, mod is the nearest
           post-modifier, and there is no words be-
           tween mod and curunit, then {
21             Locate mod just after curunit;
22         } else {
23             Insert a new row just after the row of
               the curline, and make it curline;
24             Locate mod in curline at the column
               after that in which the last character
               of curunit is located.
25         }
26     }
27 }
28 The root unit and its direct modifiers are adjusted
   to be located in the same column.
```

Figure 4: Algorithm for presenting linguistic structure

The algorithm for presenting linguistic structures we have developed is shown in Fig. 4. In this algorithm, please note that main verbs and its modifier clauses are used as presentation units (modifiees and modifiers) in the reduced view, and words are used as presentation units in the expanded view.

We have developed a GUI-based LAL-annotation editor that provides a structural views by using the above algorithm. Fig. 5 shows screen im-

ages of the editor. In the reduced view (as shown in (a)), an end user can easily grasp the overall structure so that "IBM" modify "announced," the phrase "a new computer" modifies (or is an direct object of) "announced," and the phrase "with voice recognition function" modifies "announced," etc. In this case, since the dependencies between "for" and "announced," and "with" and "announced" are wrong, a user changes the mode to the expanded view (as shown in (b)). In this view, a user can change dependencies by dragging a modifier to the correct modifiee using a mouse. The corrected dependency structure is shown in (c).

Fig. 6 shows the output of LAL editor for the above English sentence.

This algorithm is language-independent except for determining if forward modification or backward modification is major. Fig. 7 shows a screen image of the LAL editor for a Japanese sentence which is a translation of the above English sentence.

5 Discussion

There have been several efforts to define tags for describing language resources, such as the Text Encoding Initiative [15], OpenTag [11], Corpus Encoding Standard [1], the Expert Advisory Group on Language Engineering Standards [2], Global Document Annotation (or 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 very complex tag set. This complexity discourages people from using these tag sets when writing documents, and it becomes difficult to make an assisting tool for annotating the tags. However, LAL is not opposed to these previous efforts, but rather proposes a certain level of subset of the tags that can be used widely. In addition to this objective, as mentioned earlier, LAL's main objective is to help make NLP programs very accurate. Therefore, LAL includes task-specific annotations.

There has been some discussions about the merits of linguistic annotation tags for ordinary people. For instance, Hashida [4] stated that wide usage of such tags would greatly improve the results of NLP programs for applications such as machine translation, information retrieval, information extraction, summarization, question-answering system, example-based reasoning, and data mining, and that this would encourage ordinary people to use linguistic annotation tags. Some NLP researchers expect that since many users create HTML pages even without HTML editing tools, such users may therefore use linguistic annotation tags as well. However, it has also been observed that ordinary people write HTML pages because there is a direct advantage to them in being able to create attractive pages and an indirect advantage that the more attractive their pages, the more "hits" they will get. In contrast, linguistic annotation tags offer ordinary people only indirect advantages. Therefore, to popularize these tags, it is important to minimize the workload of adding linguistic annotation tags; that is to say, we must provide easy-to-use annotation tools. The key points in making such tools easy to use are, as mentioned earlier, minimum interaction and effective presentation. To satisfy these requirements, it is important to define a comprehensive, simple set of annotation tags.

6 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. 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] "Data elements and interchange formats – Information interchange – Representation of dates and times," ISO 8601:1988.

[6] Kurohasi, S., and Nagao, M., "A Syntactic Analysis Method of Long Japanese Sentences based on the Detection of Conjunctive Structures," Computational Linguistics, Vol. 20, No. 4, 1994.

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

[8] McCord, C. M., "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.

```
⟨?xml version="1.0" encoding="US-ASCII"?⟩
⟨lal⟩
⟨lal:s id="id1"⟩
⟨lal:w id="id1-1" mod="id1-2" pos="noun" lex="IBM" ftrs="sg,propn"⟩IBM ⟨/lal:w⟩
⟨lal:w id="id1-2" pos="verb" lex="announce"⟩announced ⟨/lal:w⟩
⟨lal:w id="id1-3" mod="id1-5" pos="det" lex="a" ftrs="sg"⟩a ⟨/lal:w⟩
⟨lal:w id="id1-4" mod="id1-5" pos="adj" lex="new"⟩new ⟨/lal:w⟩
⟨lal:w id="id1-5" mod="id1-2" pos="noun" lex="computer" ftrs="sg,cn"⟩computer ⟨/lal:w⟩
⟨lal:w id="id1-6" mod="id1-5" pos="prep" lex="for"⟩for ⟨/lal:w⟩
⟨lal:w id="id1-7" mod="id1-6" pos="noun" lex="child"⟩children ⟨/lal:w⟩
⟨lal:w id="id1-8" mod="id1-5" pos="prep" lex="with"⟩with⟨/lal:w⟩
⟨lal:w id="id1-9" mod="id1-10" pos="noun" lex="voice"⟩voice ⟨/lal:w⟩
⟨lal:w id="id1-10" mod="id1-11" pos="noun" lex="recognition"⟩recognition⟨/lal:w⟩
⟨lal:w id="id1-11" mod="id1-8" pos="noun" lex="function"⟩function⟨/lal:w⟩
.
</lal:s⟩
⟨/lal⟩
```

Figure 6: Example of LAL Annotation Output

[9] McCord, C. M., "Heuristics for Broad-Coverage Natural Language Parsing," Proc. of the ARPA Human Language Technology Workshop, 1993.

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

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

[12] SGML, "ISO/IEC 8879-1986 (E). Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML). First Edition – 1986-10-15.International Organization for Standardization," 1986.

[13] Takeda, K., "Pattern-Based Context-Free Grammars for Machine Translation," Proc. of 34th ACL, pp. 144–151, June 1996.

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

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

[16] Watanabe, H., "A Method for Abstracting Newspaper Articles by Using Surface Clues," Proc. of 16th International Conference of Computational Linguistics, pp. 974–979, Aug. 4-9, 1996.

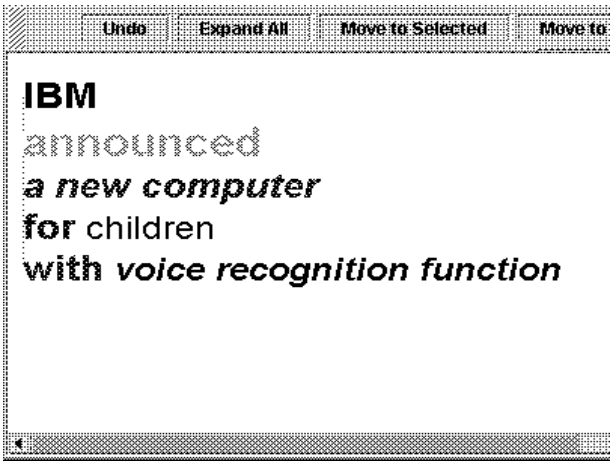
[17] Watanabe, H., and Takeda, K., "A Pattern-based Machine Translation System Extended by Example-based Processing," Proc. of the 36th ACL & 17th COLING, Vol. 2, pp. 1369-1373, 1998.

[18] Watanabe, H., "Linguistic Annotation Language – The Markup Language for Assisting NLP programs –," IBM Research Report RT0334, 1999.

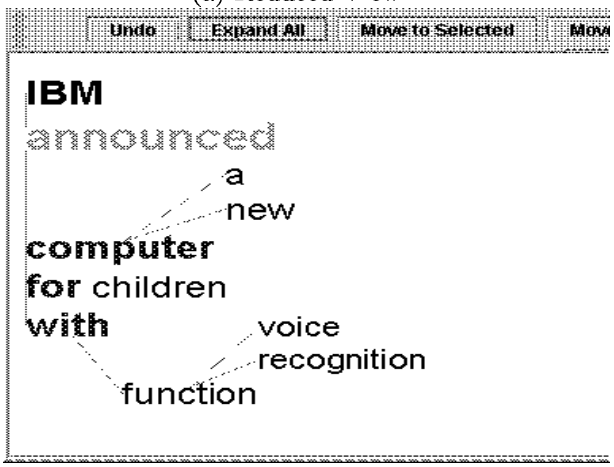
[19] Watanabe, H., "A Method for Accelerating CFG-Parsing by Using Dependency Information," Proc. of 18th COLING, 2000.

[20] XML, "Extensible Markup Language (XML)," (http://www.w3.org/TR/PR-xml-971208), World Wide Web Consortium, Dec. 8, 1997.

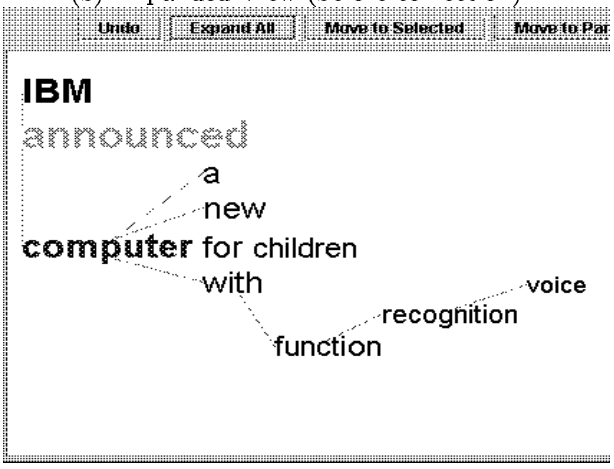
[21] XMLNS, "Namespaces in XML," (http://www.w3.org/TR/1998/WD-xml-names-19980327), World Wide Web Consortium, March 27, 1998.



(a) Reduced View



(b) Expanded View (before correction)



(c) Expanded View (after correction)

Figure 5: Screen Image of LAL Editor for English sentence

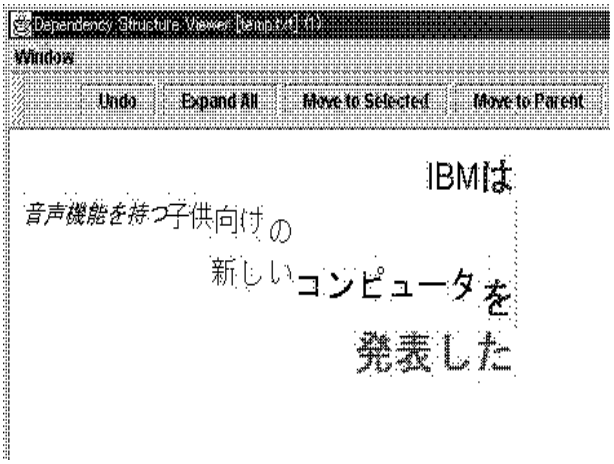


Figure 7: Screen Image of LAL Editor for Japanese sentence