# Processing Japanese Self-correction in Speech Dialog Systems

**Kotaro Funakoshi, Takenobu Tokunaga** and **Hozumi Tanaka**
Department of Computer Science
Tokyo Institute of Technology
{koh, take, tanaka}@cl.cs.titech.ac.jp

## Abstract

Speech dialog systems need to deal with various kinds of ill-formed speech inputs that appear in natural human-human dialog. Self-correction (or speech-repair) is a particularly problematic phenomenon. Although many ways of dealing with self-correction have been proposed, these have limitations in both detecting and correcting for this phenomenon. In this paper, we propose a method to overcome these problems in Japanese speech dialog. We evaluate the proposed method using our speech dialog corpus and discuss its limitations and the work that remains to be done.

## 1 Introduction

Self-correction, or speech repair, is a major source of the disfluencies that speech dialog systems have to resolve. Since Hindle (1983), there have been many proposals as to how speech dialog systems can deal with self-correction (Bear et al., 1992; Nakatani and Hirschberg, 1993; Den, 1997; Nakano and Shimazu, 1998; Core and Schubert, 1999).

Through self-monitoring, human speakers can instantly correct their mistakes during an utterance (Levelt, 1989). Therefore, we can detect self-correction with local models such as the Repair Interval Model (RIM) proposed by Nakatani and Hirschberg (1993). Most work has used the same model or models similar to the RIM. The RIM divides the self-correction into three intervals, reparandum (RPD), disfluency (DF), and repair (RP), and assumes that these three intervals appear in the order of " ... RPD DF RP ...." For example, "[I want]$_{\mathrm{RPD}}$ [uh]$_{\mathrm{DF}}$ [I want]$_{\mathrm{RP}}$ a window seat." However, human speakers do not always correct errors immediately. Sometimes a moment passes before we realize that we made an error. Or sometimes we change our intention, or decide to add more information during utterances. In such cases, in Japanese, we can correct our utterances at positions beyond more than one constituent (figure 1).

In figure 1, solid arrows show the dependency between words [1]. The dashed arrow shows the correspondence between two constituents that share the same semantic role with regard to the verb "*osite* (push)." These two constituents refer to the same object, but describe different information. In this utterance, the speaker corrects "*akai tama* (red ball)" (RPD) with "*ookii tama* (big ball)" (RP) beyond "*osite*". The RIM cannot capture this self-correction because "*osite*" is not a DF.

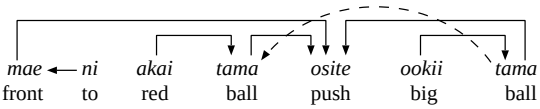


Figure 1: Long-distance self-correction in Japanese (Push the big red ball ahead)

After the detection of self-corrections, which cause ill-formedness, we have to restore the well-formedness of utterances. For this purpose, past work proposed the deletion of detected RPDs. This method works well in many cases, but sometimes removes too much or too little information. Core and Schubert (1999) pointed out this problem, but provided no procedure to solve it. To resolve this problem, we have to merge the RPD and the RP into one constituent.

In this paper, we propose a method to handle Japanese self-correction on our incremental dependency parser. We extend the model for Japanese self-correction and propose a method to merge RPD and RP to solve the problems mentioned above. We evaluate our method by using our quasi-dialog corpus and discuss its limitations and our future work. Section 2 describes our parser based on Japanese dependency analysis. Section 3 shows how to deal with the self-correction on the parser. Section 4 discusses the evaluation on the corpus. We conclude and look at our future research direction in section 5.

[1] "*ni* (to)" is a postposition. The usual Japanese dependency analysis makes "*mae* (front)" depend on "*ni*". In this paper, however, our analysis makes "*ni*" depend on "*mae*" because postpositions are often omitted in spoken Japanese.

## 2 Incremental dependency parser

We use an incremental dependency parser because incremental processing is requisite for current and future speech dialog systems.

### 2.1 Dependency parser

We can describe the Japanese syntactic structure in a regular expression as "(C F*)+", where C is a content word and F is a function word. We call the pattern "(C F*)" *bun-setsu* [2]. A function word depends on the preceding content word.

The parser creates a dependency tree on a stack, an element of which keeps a subtree of the structure. The parser maintains multiple stacks simultaneously, each of which corresponds to a different hypothesis.

After the parser receives a word sequence from the speech recognizer, it pushes words on the stack incrementally. When a content word is pushed into a stack, all the succeeding function words are attached to the content word [3]. If two consecutive function words are not allowed to adjoin, the parser considers the second function word as a correction of the first one, and replaces the first word with the second.

When more than one *bun-setsu* are created on the stack, the parser pops up the first two elements of the stack ($t_2$, $t_1$, respectively, in figure 2), then checks to see if a dependency between $rw_1$ and $rw_2$ is possible. Here, $rw_i$ denotes the root word of the subtree $t_i$. If a dependency is possible, the parser duplicates the stack. The parser restores the original stack by pushing down the two popped-up elements. In the new stack, the parser pushes down a new element containing the dependency of $rw_1$ and $rw_2$. To this new stack, the parser recursively applies the same procedure.

In the example in figure 2, when the verb "*osite*" is pushed onto a stack "[(*mae-ni*)|((*akai*) *tama*)>" [4] and no function word follows "*osite*", the parser generates three stacks:

[(*mae-ni*)|((*akai*) *tama*)|(*osite*)>
[(*mae-ni*)|(((*akai*) *tama*) *osite*)>
[((*mae-ni*)((*akai*) *tama*) *osite*)>

The parser assigns a score to each hypothesis and limits the number of hypotheses. (In this paper, we do not describe the score computation.)

### 2.2 Grammar and dictionary

When a content word $C_1$ depends on another content word $C_2$, we assume that $C_1$ takes a semantic role with regard to $C_2$. Possible semantic roles and constraints to assign the roles are described in the dictionary (figure 3).

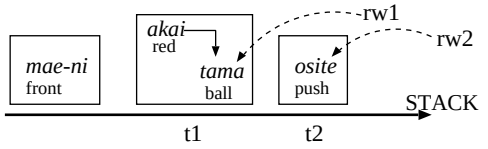


Figure 2: An example of stacks for "*mae ni akai tama osite*" (Push the red ball ahead)

```
osite VERB IMP+ PUSH+ DEGREE:#STD
    <OBJ>  1 *  NOUN wa|wo|mo|-  INSTANCE+
    <SBJ>  1 *  NOUN wa|ga|mo|-  ANIMATE+ INSTANCE+
    <TO>   1 *  NOUN ni|e|-      LOCATION+
    <FROM> 1 *  NOUN kara        LOCATION+
    <EXT>  1 *  ADV  -           DEGREE:*
```

Figure 3: Dictionary entry for "osite" (push)

The first line in figure 3 shows the features of a verb "*osite* (push)";

- a verb (`VERB`)
- imperative (`IMP+`)
- action PUSH (`PUSH`)
- degree of act (`DEGREE:#STD`).

The following lines show the roles of the verb and the constraints on each role. For example, the second line specifies the constraints on a word that will take the `<OBJ>` role;

- number of words which will take this role (`1`)
- direction of dependency, forward or backward ("`*`" means "both")
- part of speech (`NOUN`)
- postpositions which mark this role ("*wa*", "*wo*", "*mo*" and "– (unmarked)")
- semantic feature (`INSTANCE+`).

The parser assigns a role to every dependent according to the dictionary. By referring to these roles, a syntactic tree can be easily transformed into a semantic frame. Moreover, they help the restoration process for self-corrections.

## 3 Self-correction

When we applied past work to our corpus (QDC, see section 4 for details), we encountered two problems as mentioned in section 1. One was that a model like the RIM (Nakatani and Hirschberg, 1993) cannot detect long-distance self-corrections like that shown in figure 1. The other was that the restoration procedure was not sufficient.

[2] Bun-setsu means "sentence segment".

[3] We assume that speakers never interrupt their speech in the middle of a bun-setsu.

[4] "[" and ">" show the bottom and the top of the stack, respectively. "|" indicates a boundary of two elements. "( )" indicates a dependency.

In this section, we explain our self-correction classification, and then show how self-corrections can be detected and well-formedness restored with the parser introduced in section 2.

### 3.1 Self-correction classification

Figure 4 shows the self-correction classification. We classify self-correction into three classes: addition, repair, and restart.

**Addition** Addition includes simple repetition. There is an addition constraint that an RPD and an RP must be able to refer to the same object or the same manner. The RIM can detect only adjacent additions. Therefore, we introduce a new class – long-distance. Most work has treated both addition and repair in the same way. However, since there is a difference in usage, we distinguish between them.

**Repair** An explicit repair is accompanied by an *editing term*, such as "*gomen* (sorry)", "*tigau* (no)", and so on. An implicit repair is not.

A repair is further classified into adjacent and long-distance. However, implicit long-distance repair is generally not acceptable in Japanese.

**Restart** A restart occurs when a speaker interrupts the current utterance and begins a new one. There are explicit and implicit restarts. An implicit restart is generally difficult to detect without using prosodic information, gesture, etc. Furthermore, it is difficult to distinguish an explicit restart from an explicit repair. However, once a restart is detected, the ill-formedness can be easily corrected by simply removing the words before the editing term.

In this paper, we focus on addition and repair and do not discuss restart any further.

**Hesitation** Most of the methods that have been reported handle hesitation in the resolution of self-correction. They rely on partial matching between the word fragment and the following word to detect hesitations. However, we cannot reasonably assume that current speech recognizers can recognize a word fragment as a correct word. Moreover, this does not work if the following word is a different word regardless of recognition results. Therefore, we do not handle hesitations with self-corrections. Instead, we provide word skipping to deal with hesitation and speech recognizer misrecognition. This is done before each dependency analysis, but we will not give a detailed explanation here of word skipping.

### 3.2 Processing addition and repair

Some methods such as (Bear et al., 1992) process self-corrections sentence by sentence. However, such approaches are not suitable for use with incremental parsing (Nakano and Shimazu, 1998). To deal with self-corrections incrementally, we embed the self-correction process in the parser like (Hindle, 1983); that is, after each dependency analysis, self-correction is checked for and restoration is done if necessary. The parser considers the stack's top element $t_2$ as an RP, and the second element $t_1$ (or part of $t_1$) as an RPD (see figure 2).

1. **addition**
   (a) **adjacent** "[*akai tama wo*] (red ball) [*ookii tama wo*] (big ball) *osite* (push)"
   (b) **long-distance** "[*akai tama wo*] (red ball) *osite* (push) [*ookii tama wo*] (big ball)"
2. **repair**
   (a) **explicit**
      i. **adjacent** "[*akai tama wo*] (red ball) *gomen* (sorry) [*aoi tama wo*] (blue ball) *osite* (push)"
      ii. **long-distance** "[*akai tama wo*] (red ball) *osite* (push) *gomen* (sorry) [*aoi tama wo*] (blue ball)"
   (b) **implicit**
      i. **adjacent** "[*akai tama wo*] (red ball) [*aoi tama wo*] (blue ball) *osite* (push)"
      ii. ***long-distance** "[*akai tama wo*] (red ball) *osite* (push) [*aoi tama wo*] (blue ball)"
3. **restart**
   (a) **explicit** "[*akai tama wo*] (red ball) *gomen* (sorry) [*uma wa mae ni itte*] (Horse, go ahead)"
   (b) **implicit** "[*akai tama wo*] (red ball) [*uma wa mae ni itte*] (Horse, go ahead)"

Figure 4: Classification of Japanese self-correction

First, we explain how to handle long-distance self-corrections. From examination of our corpora and our linguistic introspection, we assume that long-distance self-corrections always take the following pattern:

... RPD ... verb DF RP ....

Also, the RPD necessarily depends on the verb. As a result, the RPD and the RP of long-distance self-corrections must be the pairs of noun phrases or adverbial phrases.

There are two ways to treat long-distance self-corrections. One is to assume that the RP backward depends on the verb (as in figure 1); that is, by assuming that a long-distance self-correction is a combination of inversion and self-correction. The other is to assume that the RP forward depends on the omitted verb; that is,

... RPD ... verb DF RP (verb) ....

In this case, supplementing the omitted verb reduces the above structure to the conventional RIM structure "... RPD DF RP ...." This interpretation has an advantage in that we can handle various self-corrections uniformly by supplementing omitted verbs. However, we use the former interpretation

1. Pop the element of the stack as $t_2$. Pop the next element as $t_1$. If $t_1$ is an editing term, pop the third element as $t_1$ and set the flag indicating explicitness. The root word of $t_1$ is $rw_1$ and that of $t_2$ is $rw_2$.
2. The case that $rw_1$ and $rw_2$ belong to the same part of speech (adjacent):
   (a) If $rw_1$ and $rw_2$ satisfy the replacement conditions, return that $rw_2$ is a candidate of self-correction to $rw_1$. Go to 4.
3. The case that $rw_1$ and $rw_2$ belong to a different part of speech (long-distance):
   (a) If $rw_1$ is not a verb, return that $rw_2$ is not a self-correction to $rw_1$ and go to 4.
   (b) If $rw_2$ is neither a noun nor an adverb, return that $rw_2$ is not a self-correction to $rw_1$ and go to 4.
   (c) If there is any content word $d_i^{rw_1}$ depending on $rw_1$ that satisfies the replacement conditions with $rw_2$, return that $rw_2$ is a self-correction candidate to $d_i^{rw_1}$. However, if it is not explicit, $rw_2$ and $d_i^{rw_1}$ have to indicate the same object or the same manner. Then go to 4.
4. Push back the popped elements and exit.

Figure 5: Detection algorithm of addition and repair

because supplementing verbs has a high processing cost and the former seems more natural.

**Detection** Figure 5 shows an algorithm to detect addition and repair in our parser that was described in section 2. This algorithm is based on the classification in figure 4.

The replacement conditions given in figure 5 are the constraints that $rw_1$ or $d_i^{rw_1}$ (a content word depending on $rw_1$) and $rw_2$ must satisfy. Figure 6 shows an example of the replacement conditions for nouns. This is almost the same as Class (I) of Classification (A) in (Nakano and Shimazu, 1998). We show how the replacement conditions work with the following utterance [5].

| *kamera-ha* | *mae-wo* | *mae-ni* | *itte* |
|---|---|---|---|
| (camera-topic) | (front-obj) | (front-to) | (go) |

The possibility of a self-correction between "*kamera-ha*" and "*mae-wo*" is not detected because it does not satisfy any condition listed in figure 6. However, the possibility of a self-correction between "*mae-wo*" and "*mae-ni*" is detected because it satisfies the second condition in figure 6. Here, "*ha*" is a postposition indicating the topic (in this case the subjective case as well) and "*wo*" indicates the objective case.

[5] "*kamera-ha*" shows that a postposition "*ha*" is attached to a content word "*kamera*".

- $F_1 = F_2$
- $N_1 = N_2$ and $F_2 \neq$ nil
- $N_1 \sim N_2$ and $F_1 =$ nil

$F_i$ is a function word and $N_i$ is a noun, and $F_i$ depends on $N_i$. "$N_1 = N_2$" means that $N_1$ and $N_2$ are the same word. "$N_1 \sim N_2$" means that $N_1$ and $N_2$ belong to the same semantic class. "$F_1 =$ nil" means that $N_1$ does not have any function word.

Figure 6: Replacement conditions for noun phrases

**Restoration** In the past, detected RPDs have simply been removed from hypotheses. Core and Schubert (1999) illustrated the problem of this approach with an example: "have the engine take the oranges to Elmira, um, I mean, take them to Corning." They claimed that removing the RPD lost the referent of "them." Their answer to this problem was to pass the RPDs with hypotheses to the semantic analysis. However, they did not provide any specific idea as to how the semantic analysis should handle the RPDs. Moreover, discarding the relation between an RPD and an RP at the syntactic analysis stage makes hypotheses difficult to score.

In our method, the parser processes a self-correction as soon as it is detected. Also, the parser does not identify the type of self-correction after restoration because a self-correction, especially at the beginning of an utterance, can be interpreted in several ways. For example, we can interpret the utterance, "I, I go there", as a repetition (addition), a restart, or a hesitation or misrecognition of the speech recognizer. However, every interpretation generates the same result: "I go there." Therefore, our parser keeps one hypothesis for several interpretations and does not keep redundant hypotheses which waste resources.

When subtree $t_1$ (RPD) is replaced by subtree $t_2$ (RP), the parser merges the two subtrees. More specifically, the parser transfers information from $t_1$ to $t_2$ by copying missing words in $t_2$, unless this causes a contradiction.

For example, from the two dependency trees $t_1$ and $t_2$ of the following utterance,

[*sono akai usiro no tama*]$_{t_1}$ *gomen* [*aoi tama*]$_{t_2}$ ....
that red back of ball sorry blue ball ....

| | | | | |
|---|---|---|---|---|
| $t_1$: | ((*sono*) | (*akai*) | (*usiro-no*) | *tama*) |
| | <IND> | <COL> | <LOC> | |
| $t_2$: | | ((*aoi*) | | *tama*) |
| | | <COL> | | |

the parser generates $t_2'$. <X> indicates a semantic role has been given to the word just above <X>.

$t_2'$: ((*sono*) (*aoi*) (*usiro-no*) *tama*)
<IND> <COL> <LOC>

The parser creates a correspondence between words in $t_1$ and $t_2$ by matching words and roles.

Let subtree $t_1$ with a root word $rw_1$ be replaced by subtree $t_2$ with a root word $rw_2$. Here, $d_1^{rw_1}, \ldots, d_m^{rw_1}$ are the $m$ words depending on $rw_1$, and $d_1^{rw_2}, \ldots, d_n^{rw_2}$ are the $n$ words depending on $rw_2$. In figure 7, we show the algorithm of the function dmerge($rw_1, rw_2$) which copies the missing words from $t_1$ to $t_2$. This algorithm is designed to satisfy three assumptions concerning human recognition of self-correction.

**Assumption 1** If there is a one-to-one correspondence between $d_i^{rw_1}$ and $d_j^{rw_2}$ with respect to their word form, then $d_i^{rw_1}$ can be recognized as replaceable by $d_j^{rw_2}$, even though the role of $d_i^{rw_1}$ with regard to $rw_1$ differs from the role of $d_j^{rw_2}$ with regard to $rw_2$.

**Assumption 2** If there is a one-to-one correspondence between $d_i^{rw_1}$ and $d_j^{rw_2}$ with respect to their semantic role, then $d_i^{rw_1}$ can be recognized as replaceable by $d_j^{rw_2}$, even though $d_i^{rw_1}$ and $d_j^{rw_2}$ are different words.

**Assumption 3** Only if there is no discrepancy between assumptions 1 and 2, $d_i^{rw_1}$ can be recognized as replaceable by $d_j^{rw_2}$.

Assumption 1 corresponds to step 1 in figure 7 and assumption 2 corresponds to step 2.

Because of assumption 1 and step 4 in figure 7, from $t_1$ and $t_2$ of the next utterance,

[*are migi kara osite*]$_{t_1}$ *gomen* [*migi ni osite*]$_{t_2}$
that right from push sorry right to push

$t_1$: ((*are*) (*migi-kara*) *osite*)
<OBJ> <FROM>
$t_2$: ((*migi-ni*) *osite*)
<TO>

we can make the correct subtree $t_2'$ instead of the wrong subtree $t_2''$.

$t_2'$: ((*are*) (*migi-ni*) *osite*)
<OBJ> <TO>
$t_2''$: ((*are*) (*migi-kara*) (*migi-ni*) *osite*)
<OBJ> <FROM> <TO>

Because of assumption 2 and step 4 in figure 7, from $t_1$ and $t_2$ of the following utterance,

[*tukue no mae ni osite*]$_{t_1}$ *gomen* [*usiro ni osite*]$_{t_2}$
desk of front to push sorry back to push

$t_1$: (((*tukue-no*) *mae-ni*) *osite*)
<GEN> <TO>
$t_2$: ((*usiro-ni*) *osite*)
<TO>

dmerge($rw_1, rw_2$): copies missing words from subtree $t_1$ with root word $rw_1$ to subtree $t_2$ with root word $rw_2$

1. Find the correspondence between $d_i^{rw_1}$ and $d_j^{rw_2}$ with respect to their word form, where $i = 1 \ldots m$ and $j = 1 \ldots n$.
2. Find the correspondence between $d_i^{rw_1}$ and $d_j^{rw_2}$ with respect to their role, where $i = 1 \ldots m$ and $j = 1 \ldots n$.
3. Pick up the pairs of one-to-one correspondence found in steps 1 and 2, and apply dmerge to them recursively. However, these correspondences have to be coherent between steps 1 and 2.
4. copy to $rw2$ any $d_i^{rw_1}$ for which no correspondence is found in steps 1 and 2.
5. If the semantic class of $rw_2$ subsumes that of $rw_1$, replace $rw_2$ with $rw_1$.

Figure 7: Merge algorithm

we can make the correct subtree $t_2'$.

$t_2'$: (((*tukue-no*) *usiro-ni*) *osite*)
<GEN> <TO>

The third assumption (and the condition of step 3 concerning coherency in figure 7) means that even a human cannot make a unique interpretation if the word correspondence between $t_1$ and $t_2$ is ambiguous. However, the third assumption seems too strong. This algorithm thus needs further investigation and refinement.

In this algorithm, we use roles instead of postpositions to create correspondence between words. Since postpositions are often omitted in spoken Japanese, our method is more robust than using postpositions directly. Moreover, in the case of constituents in which dependency is not explicitly marked by postpositions, such as adjective-noun, noun-noun constructions, our method has an advantage in that the parser does not need extra computation to find the correspondence.

The problem of pronouns in RPs, as pointed out by (Core and Schubert, 1999), is resolved in step 5 in figure 7.

## 4 Evaluation

The QDC (quasi-dialog corpus) [6] has 15 dialogs and 532 utterances between a user and three virtual agents simulated by humans. This corpus was collected in the following setting. A user could command two agents to arrange objects in a virtual world. The remaining agent was a camera which presented a world view to the user. The camera agent itself could not handle objects; it could only

[6] The QDC is open to the public at http://tanaka-www.cs.titech.ac.jp/pub/qdc/ .

change the user's view according to orders given by the user.

We manually applied our proposed method to the QDC. In the QDC, we found 71 self-corrections, excluding hesitations, and we judged that 16 of these were not properly solved by our method. Although our parser produces scored N-best hypotheses, we counted only the first hypotheses as the answer.

Because this corpus is small and we applied our method manually, we do not discuss quantitative evaluation. Instead, we will show examples [7] that our method could not handle properly.

**Type 1: Separated RPD and RP**

This type of error is classified into class (II) of classification (A) in (Nakano and Shimazu, 1998). This is another case where RPD and RP are separated. This occurs at the beginning of utterances.

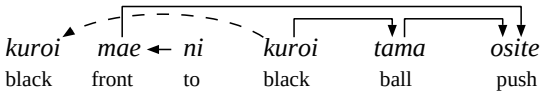


Figure 8: Example of a type 1 error

Since our model allows only a verb between the RPD and the RP (see section 3.2) and "*mae ni*" is not a verb, this example cannot be handled by our method. In particular, "*mae ni*" blocks the first "*kuroi*" from the second "*kuroi*" and they are never next to each other on the stack. Our method thus cannot detect this self-correction. Although we cannot find a correspondence between an RPD and an RP in such cases, we can usually obtain a proper hypothesis through word skipping. Moreover, if we can handle restart properly, many cases will be solved without problem.

**Type 2: Information lost through word replacement**

There was one instance of this type. In the method proposed in this paper, we save information by copying missing words from an RPD to an RP. However, this method loses important information in cases such as the following.

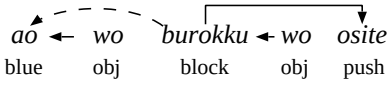


Figure 9: Example of a type 2 error

The speaker first mentioned a blue object by using a noun "*ao* (blue)", and referred to it again with a noun "*burokku* (block)." In most cases, no problem arises if a noun is replaced by a more specific noun. In this case, though, the user referred to the same object twice in terms of a different aspect – color and object type. Simple replacement of nouns in such cases causes the loss of important information. To deal with such a case, we need more semantic operations with ontological knowledge.

**Type 3: Requiring deep semantic analysis**

There were nine instances of this type.

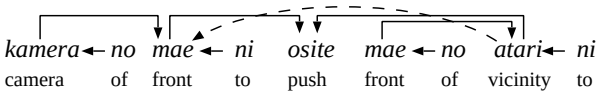


Figure 10: Example of a type 3 error (1)

In the example in figure 10, our method creates a correspondence between "*mae ni*" and "*atari ni*" and copies "*kamera no*" to "*atari ni.*" However, it should be attached to "*mae no*" instead of "*atari ni.*" Our restoration algorithm (figure 7) works word by word, but it should have worked on a larger unit in this case.

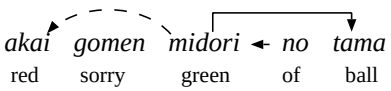


Figure 11: Example of a type 3 error (2)

In the example in figure 11, our method (as well as other methods that handle self-corrections at the surface level) fails to detect the self-correction concerning color because of the difference in the parts of speech between "*akai*" and "*midori no.*" In Japanese, there is an adjective corresponding to "red", but no one corresponding to "green". Instead we use the noun phrase "*midori no* (of green)."

**Type 4: Head omission**

Two examples of this type were found, and both were almost the same expressions used by the same person. In the example in figure 12, the last phrase "*kamera no* (of the camera)" would depend on the second phrase "*migi ni* (to the right of)." However, it cannot depend on "*migi*" because this would cause cross-dependency (the bold solid arrow in figure 12).

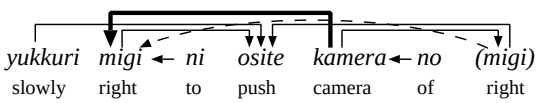


Figure 12: Example of a type 4 error

To resolve this self-correction as a long-distance addition, we have to complement the omitted head noun "*migi*" at the end of the utterance.

[7] These examples are simplified for explanation.

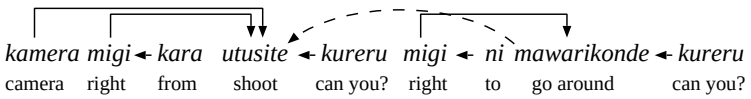


Figure 13: Example of a type 5 error

**Type 5: Superficial self-correction**
We found one instance of this type. In figure 13, the first utterance directs the camera to shoot the object mentioned before[8] from its right side. The second utterance directs the camera to go around to the right of the object. The first impression is that this is a simple self-correction of clauses. However, the second instruction is an elaboration of the first one; namely, it instructs the camera to go around to the right side of the object to shoot it. To distinguish such an utterance from self-corrections, we need to parallelize the intention understanding with the syntactic analysis.

## 5 Conclusion

In this paper, we have expanded the conventional model of self-correction and have proposed a method to restore well-formedness without information loss. A parser incorporating this method can handle a wider variety of expressions. While the parser itself might encounter difficulties if applied directly to other languages, such as English, the basic principles of the restoration process are applicable to languages other than Japanese.

We evaluated the proposed method when applied to our quasi-dialog corpus and identified unresolved problems. As future work, we intend to evaluate our method in an actual speech dialog system.

Our method uses only syntactic and semantic information to detect self-corrections. We can find most self-correction candidates with these types of information. However, we need more information to evaluate the validity of each candidate. In our method, the more words a hypothesis contains, the higher the score assigned to the hypothesis. This policy causes a problem. For example, if the parser receives the sentence in figure 14 containing a self-correction of "*akai*", it produces at least the following three hypotheses.

A: ((((*akai*) *isu-no*) *mae-no*) *tama*)
the ball at the front of the red chair

B: ((*akai*) ((*isu-no*) *mae-no*) *tama*)
the red ball at the front of the chair

C: ((*akai*) (((*akai*) *isu-no*) *mae-no*) *tama*)
the red ball at the front of the red chair

Although the parser should rank these hypothesis in the order of A, B, C, it actually ranks them in the order of C, A, B, because hypothesis C contains the most words. However, the policy that causes this problem is needed to prevent the parser from removing too much information. If we give preference to any self-correction candidate, the parser will rank the following (wrong) hypothesis above hypotheses A, B, and C.

((*akai*) (*mae-no*) *tama*)
the red ball at the front (of you)

To resolve this problem, we have to utilize acoustic-prosodic information, as in (Bear et al., 1992; Nakatani and Hirschberg, 1993), for scoring to boost the ranks of hypotheses where self-corrections are restored. This will be especially critical for restarts, because restarts usually disregard many words.

[8] This is a zero pronoun.

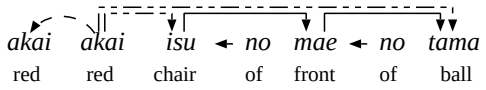


Figure 14: Example of a problematic sentence in scoring

## References

J. Bear, J. Downing, and E. Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of 30th Annual Meeting of ACL*, pages 56–63.

M. G. Core and L. K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of 37th Annual Meeting of ACL*, pages 413–420.

Y. Den. 1997. A uniform approach to spoken language analysis (in Japanese). *Journal of Natural Language Processing*, 4(1):23–40.

D. Hindle. 1983. Deterministic parsing of syntactic non-fluencies. In *Proceedings of 21st Annual Meeting of ACL*, pages 123–128.

W. J. M. Levelt. 1989. *Speaking*. The MIT Press.

M. Nakano and A. Shimazu. 1998. Parsing utterances including self-repairs (in Japanese). *IPSJ Journal*, 39(6):1935–1943.

C. Nakatani and J. Hirschberg. 1993. A speech-first model for repair identification and correction. In *Proceedings of 31th Annual Meeting of ACL*, pages 200–207.