Proceedings of DiSS'03, Disfluency in Spontaneous Speech Workshop, 5–8 September 2003, Göteborg University, Sweden.
Robert Eklund (ed.), Gothenburg Papers in Theoretical Linguistics 90, ISSN 0349–1021, pp. 55–58.

Evaluation of a robust parser for spoken Japanese

Kotaro Funakoshi & Takenobu Tokunaga

Department of Computer Science, Tokyo Institute of Technology

Abstract

We implemented a parser designed to handle ill-formedness in Japanese speech. The parser was evaluated by utilizing newly collected speech data, which was obtained from an experiment designed to produce ill-formed data effectively. Introducing the proposed method increased the number of correctly analyzed utterances from 171 to 322, from among 532 utterances in the corpus.

1. Introduction

Ill-formedness in speech is a major obstacle to designing effective speech dialogue systems. In Japanese, there are three major kinds of ill-formedness: postposition omission, inversion and self-correction. In this paper, we describe our implementation of the method previously proposed by our group [3] to handle ill-formedness. We evaluated this method by using newly collected speech data to demonstrate its effectiveness in speech dialogue systems.

In evaluating methods of dealing with ill-formedness, a major problem is to create a corpus that includes many ill-formed utterances. Although postposition omission, inversion, and self-correction are likely to occur more frequently than other, minor types of ill-formedness, their absolute frequencies of occurrence are not that large. Bear et al. [1] reported only 607 sentences containing self-corrections in a 10,000 sentence corpus (6%). According to Den [2], the ATR dialogue database has self-corrections in about 10% of its sentences. Nakano & Shimazu [6] found 704 self-corrections in their corpus of about 15,000 turns. In contrast, Heeman & Allen [4] reported 1,973 self-corrections in their Trains Corpus of 6,163 turns, and Levelt [5] reported a self-correction rate of 34% for human-human dialogue. Thus, the self-correction rate in a dialogue corpus seems be around 10% (5–30%), although it clearly varies according to the tasks involved in collection. Yamamoto et al. [8] found 171 postposition omissions in 4,063 noun phrases (4%) and 32 inversions in 1,818 utterances (1.8%). Both types of ill-formedness occur less often than self-correction.

We empirically expect that the more deliberately a speaker speaks, the less ill-formedness occurs. In contrast, ill-formedness should occur more frequently in a distractive situation in which a speaker can not concentrate on speaking. We thus designed an experiment to collect ill-formed utterances by creating such a distractive situation.

In section 2, we classify the various types of ill-formedness, and in section 3, we briefly describe our method of handling these phenomena. In section 4, we describe the procedure for collecting ill-formed data, examine the collected data, and give the results of our data analysis. We conclude the paper in section 5.

2. Ill-formedness in Japanese Speech

We consider four types of ill-formedness in Japanese speech: postposition omission, inversion, self-correction, and hesitation. In this paper, we refer to an instance of each type of ill-formedness as a *disfluency*.

2.1. Postposition Omission

In Japanese, the grammatical role of a noun phrase is marked by a postposition, and the order of postpositional phrases is relatively free. In Japanese dialogue, however, speakers often omit postpositions, and this causes difficulties in syntactic and semantic analysis. In addition, when we use automatic speech recognizers (ASRs) in dialogue systems, we have to deal with the misrecognition of postpositions. Because their acoustic power tends to be weak, postpositions tend to be misrecognized more than content words by ASRs.

Yamamoto et al. [8] reported that omission of the postpositions "*wa*", "*ga*", "*wo*", "*ni*" and "*e*" makes up about 80% of all postposition omission. In this paper, we consider seven postpositions (the above five, "*mo*" and "*no*").

2.2. Inversion

Since Japanese is a head-final language, sentences usually end with a predicate. In dialogue, however, speakers sometimes add several phrases after the predicate. We consider such cases as inversion, and we assume that these post-predicate phrases depend on the predicate.

2.3. Self-correction

Self-correction is also known as speech repair, or simply repair. In Japanese, self-correction can be combined with postposition omission and inversion [3]:

(1) "*akai tama-(wo) mae-(ni) osite migi-noyatsu-wo*"
red ball-(ACC) front-(GOAL) push right-GEN one-ACC
(Push the right red ball forward)

In example (1), the speaker corrected "*akai tama-(wo)*" ("*wo*" was omitted) by adding the inverted pronoun phrase, "*migi-no yatsu-wo*".

2.4. Hesitation

Hesitation occurs when a speaker is interrupted or fails to articulate, resulting in a word fragment in the utterance. In many cases, self-correction follows a hesitation, but not always. Moreover, it is hard for current ASRs to recognize fragments. Thus, we treat hesitation as a different phenomenon from self-correction.

3. Analysis Method

3.1. Parser and Dictionary

We adopt the dependency parser and method of handling ill-formedness proposed previously by our group [3]. The method handles all ill-formedness in the parser in parallel with syntactic analysis. We describe the parser and the dictionary it uses below.

3.1.1 Dependency Parser

We can describe a fragment of a Japanese syntactic structure in a regular expression as "(*C F**)+", where *C* is a content word and *F* is a function word. We call such a unit "(*C F**)" a phrase. The function word depends on the preceding content word.

The parser creates a dependency tree of phrases on a stack, in which each element stores a subtree of the structure. The parser maintains multiple stacks simultaneously, each corresponding to a different hypothesis (syntactic structure). After the parser receives a word sequence from the ASR, it incrementally pushes the words onto the stack.

Once a content word is pushed onto a stack, all the succeeding function words in the sequence are attached to the content word. If two consecutive function words are not allowed to adjoin, the parser considers the second one to be a correction of the first, and it replaces the first with the second. This process thus creates a phrase as one element at the top of the stack.

When more than one element is created in the stack, the parser pops the first two elements t_1 and t_2 (t_2 is at the top), then checks for the possibility of a dependency between rw_1 and rw_2. Here, rw_i denotes the root word of subtree t_i. If the dependency is possible, the parser duplicates the stack. It then restores the original stack by pushing the two popped elements back on. In the new stack, the parser pushes a new element containing the dependency of *rw1* and *rw2*. Finally, the parser recursively applies the same procedure to the new stack.

For example, suppose the verb "*osite* (push)" is pushed onto the following stack:[1]

[(*mae-ni*) | ((*aka*i) *tama*) >
forward red ball

Assuming no function word follows "*osite*", the parser generates three new 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 thus limits the number of hypotheses. In this paper, we do not describe the score calculation algorithm in detail. Briefly, the parser gives preference to dependencies between closer words and interpretations that include more words in an utterance.

3.1.2. Dictionary Description

As mentioned above, we adopted a dependency parser, which does not employ explicit grammar rules. Instead, it has hard-coded grammatical knowledge of phrase structures and utilizes dependency constraints described in the word entries of a dictionary. Here, we show how those constraints are defined.

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. The possible semantic roles and constraints on those roles are described in the dictionary as illustrated in Figure 1.

Osite	VERB	IMP+	PUSH+	
	<OBJ>	1 NOUN	wa|wo|mo	INST+
	<SBJ>	1 NOUN	wa|ga|mo	ANIM+ INST+
	<TO>	1 NOUN	ni|e	LOC+
	<FROM>	1 NOUN	wa|ga|mo	LOC+
	<EXT>	1 ADV	-	DEG:*

Figure 1: Dictionary entry for "*osite* (push)".

The first line in Figure 1 gives the features of the verb "*osite* (push)":

- part of speech (VERB)
- imperative mood (IMP+)
- action (PUSH+)

The following lines show the possible semantic roles of the verb and the constraints on each role. For example, the second line specifies the constraints on a word taking the role <OBJ>:

- number of words that can take this role (1)
- part of speech (NOUN)
- postpositions that can mark this role ("*wa*", "*wo*" and "*mo*")
- semantic feature(s) (INST+)

The parser assigns a semantic role to every dependency according to the dictionary. By referring to these roles, a syntactic tree can be easily transformed into a semantic frame. The semantic roles also help handle self-correction.

3.2. Analysis of Ill-formed Utterances

3.2.1. Postposition Omission and Inversion

We handle postposition omission and inversion by augmenting the parser and dictionary described in section 3.1. For postposition omission, we allow an unmarked (-) dependency for words that generally relate to one of the seven postpositions listed in section 2.1.

Inversion is handled by allowing not only forward dependency but also backward dependency. First, we specify the possible dependency directions for each role of every word in the dictionary by attaching one of the labels "B", "F", or "*". "F" and "B" allow only forward dependency and backward dependency, respectively, while "*" allows both. Finally, the parser can then handle dependencies in both directions.

3.2.2. Self-correction

The algorithm to handle self-correction is described in detail in our group's previous paper [3]. In this section, we explain it briefly.

The parser detects a possible self-correction by examining the two elements at the top of the stack. When the parser detects a self-correction, it duplicates the hypothesis stack, leaves one stack intact, and lets the other keep the restored data.

We give an example below:

(2) "*uma wa akai tama aoi tama osite*"
horse-TOP red ball-UM blue ball-UM push
(Horse, push the red ball blue ball)

Here, "UM" means "unmarked" due to postposition omission. This example translates as "Horse, push the red ball blue ball." By applying its encoded rules, the parser detects the possibility that *akai tama* has been corrected with *aoi tama*, after it creates a hypothesis stack:

α: [(*uma-wa*) | ((*akai*) *tama*) | ((*aoi*) *tama*) >.

Then, the parser removes the redundant part, and generates a new stack:

β: [(*uma-wa*) | ((*aoi*) *tama*) >.

Here, stack β can be extended to a correct interpretation:

γ: [((*uma-wa*) ((*aoi*) *tama*) *osite*) >.

Note that stack γ cannot be reached directly from stack α.

The method explained above cannot handle self-corrections between function words, including postpositions. All self-

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

corrections between function words are handled in creating phrases (section 3.1.1). However, speakers rarely correct only function words in a phrase. In most cases, they give a new phrase containing the correct function words. On the other hand, ASRs frequently insert incorrect function words after correctly recognized function words, due to fillers between phrases. Thus, we neglect the self-corrections between function words.

Lastly, when the parser encounters an editing term, it creates an empty stack for a restart.

3.2.3. Hesitation

It is difficult for ASRs to recognize word fragments resulting from hesitation. We handle hesitation (i.e., word fragments) by employing word skipping.

The parser skips words in parallel with the dependency analysis. We assume that misrecognized words tend to be isolated in their local contexts. Thus, to reduce ambiguity, the parser skips words that cannot depend on their neighboring words. When the parser finds such a word in a hypothesis stack, it duplicates the stack and removes the word from one of the two stacks.

4. Experiment

We collected ill-formed speech data and evaluated the analysis method described in section 3.

4.1. Data Collection

4.1.1. Domain of Collected Utterances

In this experiment, the subject's task was to arrange four colored balls in their prescribed positions by instructing four agents: Horse, Chicken, Snowman, and Camera. The agents other than the Camera could perform the following actions:

1. push an object;
2. turn to a certain direction;
3. move to a certain position or direction.

The Camera could turn and move but could not push an object. Instead, it could photograph a scene.

4.1.2. Collection Procedure

As mentioned in section 1, it is not easy to collect speech data that includes disfluencies. We thus designed our experiment to collect disfluencies as follows.

A supervisor and one subject participated in each session. We used five Japanese students (four male and one female) as subjects. The supervisor showed the subject a bird's-eye view of the current disposition of the agents and balls, as shown in Figure 2.

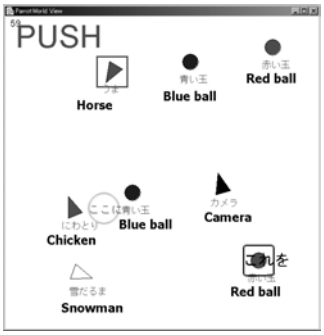

Figure 2: Revised dictionary entry for "*osite*".

The agents were denoted by isosceles triangles and oriented by the sharpest vertices, while the balls were denoted by circles and colored blue or red. The agents and balls were labeled with their names.

The stimulus consisted of a sequence of marks indicating an action and its case roles, which were agent, object, source, and destination. The marks were displayed one by one on the map at intervals of 0.5 second in random order. The stimulus thus corresponded to a command to an agent, and the subjects were instructed to express the command orally, in parallel with the sequential presentation of the marks. Since the subjects were asked to utter their commands in real time—that is, while in the process of constructing a sentence—we could expect many ill-formed utterances, containing complex self-corrections in particular, as well as simple repetitions.

In Figure 2, the action is shown at the upper left in English, the agent is marked by the square on the Horse, the object is marked by the square on one of the red balls, and the destination of the action "push" is marked by the circle between the Chicken and one of the blue balls. This stimulus could be expressed as the command: "*uma wa kamera no usiro no akai tama wo aoi tama no hidari ni osite* (Horse, push the red ball behind the Camera to the left of the blue ball)". However, the marks were not always presented in the same order as the standard surface order of the sentence. The keyword specifying the action could be shown before the other marks, so that, in such a case, an inversion would occur if the subject followed the order of the marks.

The subjects are instructed to repair their utterances freely if they thought their utterances were wrong or unnatural and they wanted to do so. However, they were not actually required to invert or correct their utterances.

4.1.3. Collected Data

We conducted two sessions with each subject and collected 536 utterances (about 50 utterances per session). The disposition of the agents and balls was changed for each session. The average length of the utterances was nine words. The collected data included 7 postposition omissions, 4 inversions, 153 self-corrections, and 49 hesitations. These disfluencies appeared in 139 utterances (26%).

We applied an ASR (AmiVoice, Advanced Media, Inc.) to the data, and it made 184 deletions, 55 insertions, and 300 substitutions. The grammar used by the ASR prescribed only the phrase structures. The vocabulary size was 120 words, including 11 fillers. The ASR recognized 203 utterances (38%) perfectly except for fillers. Of these, 168 were recognized perfectly and contained no disfluencies.

4.2. Evaluation

4.2.1. Evaluation Procedure

We implemented the parser described in section 3 and applied it to the collected data. For evaluation purposes, we used the best among the multiple interpretations produced by the parser. We then classified the syntactic analysis results into three categories.

[Correct] The resulting dependency tree matched the speaker's intention. The semantic roles assigned by the parser were also correct.

[Partially Correct] The resulting tree was a subtree of the correct tree.

[Wrong] Either the structure of the tree or the assigned semantic roles (or both) were inconsistent with the speaker's intention.

4.2.2. Results

First, we transcribed the speech data and parsed the transcribed text with the parser. Table 1 shows the parsing results, with each cell showing the number of utterances.

Table 1: Results of parsing the manual transcription.

Utterance Type	C	P	W	Total
w/ disfluency	106	2	31	139
w/o disfluency	393	1	3	396
Total	499	3	34	536

C: correct, P: partially correct, W: wrong

The parser correctly analyzed 76% of the utterances with disfluencies. For most of the incorrectly analyzed utterances with disfluencies, the parser preferred interpretations without self-corrections, because it was designed to give preference to interpretations covering more of the words in an utterance. As for the incorrectly analyzed utterances without disfluencies, the parser misinterpreted semantic roles.

Table 2 shows the parsing results for the ASR output.

Table 2: Results of parsing the automatic dictation.

Utterance Type	C	P	W	Total
w/ disfluency and misrecognition	28	17	67	112
w/ disfluency only	22	1	2	25
w/ misrecognition only	101	42	76	219
w/o disfluency and misrecognition	171	1	6	178
Total	322	61	153	536

In this case, the parser correctly analyzed 64.5% (322) of the 499 utterances that were correctly analyzed for the manual transcription. Incorporating our method of handling ill-formedness into the parser enabled it to correctly analyze 151 (= 28 + 22 + 101) of those 322 utterances. Two thirds of the recovered misrecognitions were deletions of one of the seven postpositions. The remaining one third occurred in reparanda that the speakers intended to correct.

5. Conclusion

In this paper, we reported our implementation and evaluation of a parser designed previously by our group [3] to handle ill-formedness in Japanese speech dialogue. Introducing a method of handling disfluencies into the parser enabled it to interpret 106 more utterances (about 20% of the collected data) if the ASR worked perfectly (Table 1). With the AmiVoice ASR, the number of sentences analyzed correctly was greatly improved, from 32% (171) to 60% (322).

We designed an experiment to obtain ill-formed data effectively. However, the collected data included only 7 postposition omissions and 4 inversions. This shows that the procedure described in section 4.1.2 was insufficient to produce large numbers of postposition omissions and inversions. We expect that free conversation would be more suitable than the restricted situations employed in our experiment for obtaining ill-formed data.

We also expected that the procedure would collect many complex self-corrections. However, there were only 6 self-corrections containing more than four words in the reparanda (4% of 153), not including restarts. This was slightly greater than the number reported in Ref. [7] but less than that reported in Ref. [1] and much lower than expected. Our group [3] previously pointed out that Japanese speakers can correct their utterances from distant locations within a sentence by combining inversion and self-correction, as in example (1), but we could find only 3 such self-corrections in the collected data.

6. References

[1] Bear, John, John Dowding & Elizabeth Shriberg. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. *Proc. of 30th Annual Meeting of ACL*, pp. 56–63.

[2] Den, Yasuharu. 1997. A uniform approach to spoken language analysis (in Japanese). *Journal of Natural Language Processing*, vol. 4, No. 1 pp. 23–40.

[3] Funakoshi, Kotaro, Takenobu Tokunaga & Hozumi Tanaka. 2002. Processing Japanese self-correction in speech dialog systems. *Proc. of COLING2002*, pp. 287–293.

[4] Heeman, Peter A. & James F. Allen. 1997. Intonational boundaries, speech repairs and discourse markers: Modeling spoken dialog. *Proc. of 35th Annual Meeting of ACL*, pp. 254–261.

[5] Levelt, Willem J. M. 1983. Monitoring and self-repairs in speech. *Cognition*, vol. 14, pp. 41–104.

[6] Nakano, Mikio & Akira Shimazu. 1998. Parsing utterances including self-repairs (in Japanese). *IPSJ Journal*, vol. 39, no. 6. pp. 1935–1943.

[7] Spilker, Jörg, Martin Klaner & Günther Görz. 2000. Processing self-corrections in a speech-to-speech system. Wolfgang Wahlster, editor, Verbmobil: Foundations of Speech-to-Speech Translation, Springer, pp. 131–140.

[8] Yamamoto, Mikio, Satoshi Kobayashi & Seiichi Nakagawa. 1992. An analysis and parsing method of the omission of post-position and inversion on Japanese spoken sentence in dialog (in Japanese). *IPSJ Journal*, vol. 33, no. 11, pp. 1322–1320.